ON A CLASS OF ACYCLIC DIRECTED GRAPHS

by

J. L. Szwarcfiter

Memorandum No. UCB/ERL M80/6

February 1980

ELECTRONICS RESEARCH LABORATORY

College of Engineering
University of California, Berkeley
94720

ON A CLASS OF ACYCLIC DIRECTED GRAPHS*

Jayme L. Szwarcfiter**
Universidade Federal do Rio de Janeiro
COPPE, I. Mat. e NCE
Caixa Postal 2324, CEP 20000
Rio de Janeiro, RJ
Brasil

February 1980

Key Words: algorithm, depth first search, directed graphs, graphs, isomorphism, minimal chain decomposition, partially ordered sets, reducible graphs, series parallel graphs, transitive closure, transitive reduction, trees.

CR Categories: 5.32

*This work has been supported by the Conselho Nacional de Desenvolvimento Cientifico e Tecnologico (CNP_q), Brasil, processo 574/78. The preparation of the manuscript has been supported by the National Science Foundation, grant MCS78-20054.

**Present Address: University of California, Computer Science Division-EECS, Berkeley, CA 94720, USA.

ABSTRACT

A special class of acyclic digraphs has been considered. It contains those acyclic digraphs whose transitive reduction is a directed rooted tree. Alternative characterizations have also been given, including one by forbidden subgraph containment of its transitive closure. For digraphs belonging to the mentioned class, linear time algorithms have been described for the following problems: recognition, transitive reduction and closure, isomorphism, minimal chain decomposition, dimension of the induced poset. The ideas of the isomorphism algorithm were further extended to include a larger class of digraphs, leading to an algorithm whose time bound is the same as that for recognizing reducible digraphs. Based on the characterizations of the considered class, the problem of verifying isomorphism of DFS of undirected graphs has also been solved in linear time.

1. Introduction

Looking for special classes of graphs possessing some particular characteristics is generally common in graph theory. In what concerns graph algorithms, we suppose that the interest in any such special class would increase depending on the following factors: (i) the importance and quantity of different general algorithmic problems that can be solved simpler and more efficiently, if the input instance is a graph of the class in consideration; (ii) the capacity of the class to generate its own "interesting" algorithmic problems; (iii) the capacity of the class to be used as an intermediate step in solving problems for larger or different classes and (iv) the size of the class.

Among others, we can mention the following classes of graphs, which have been introduced in recent years: chordal graphs (Gavril [6]), comparability graphs (Even, Pnueli and Lempel [4]), interval graphs (Booth and Lueker [2]), reducible digraphs (Hecht and Ullman [7]), series parallel digraphs (Valdes, Tarjan and Lawler [13]).

In the present paper, we present a special class of acyclic digraphs, named tree structured digraphs. This class is defined and characterized in some different ways, in the next section. The recognition of whether a graph belongs to that class is discussed in Section 3. Most of the following sections contain different applications of tree structured digraphs, i.e. a discussion of some general problems, when the input is restricted to the class in consideration. In Section 4 is the problem of transitive reduction and closure. An isomorphism algorithm is described in the following section. The problem of verifying whether or not two DFS of an undirected graph are isomorphic, is solved subsequently based on characterizations and isomorphism of tree structured

digraphs. Finding a minimal chain decomposition and the dimension of the induced poset are problems considered in Sections 7 and 8, respectively. The isomorphism of tree structured digraphs is extended in Section 9, to include also digraphs (possibly with cycles) that constitute a defined subclass of reducible digraphs. A proposal of a problem related to the class in consideration is presented in Section 10 and some additional remarks form the last section.

The algorithms described through the paper have all linear time bounds, except that of Section 9 which is almost linear, but not linear.

A graph (V,E) is a finite non-empty set V together with a set E of pairs of distinct elements of V. The elements of V and E are the vertices and edges of the graph, respectively. We denote $n = |V|$ and $m = |E|$. A graph can be undirected or directed (digraph) according to whether its edges are unordered or ordered pairs, respectively. A vertex v is adjacent to vertex w if $(v, w) \in E$. The adjacency list A(v) of vertex v is the set of vertices w of the graph, such that v is adjacent to w. A sequence of vertices $v_1, \ldots, v_k$ such that for $1 \leq i < k$ $(v_i, v_{i+1}) \in E$ is called a path from v_1 to v_k and v_1 is said to reach v_k. The length of such a path is defined as $k - 1$. A cycle is a path $v_1, \ldots, v_k$ with $v_1 = v_k$ and containing at least two different edges. A graph with no cycles is acyclic. A graph $G_1(V_1,E_1)$ is a subgraph of graph $G_2(V_2,E_2)$ when $V_1 \subseteq V_2$ and $E_1 \subseteq E_2$. If additionally $V_1 = V_2$ then G_1 is called a spanning subgraph of G_2. An undirected graph is connected if there is a path between every pair of its vertices. A tree is an undirected graph which is connected and acyclic. In a digraph the indegree and outdegree of a vertex v are the number of vertices w such that $(w,v) \in E$ and $(v,w) \in E$ respectively. If indegree (v) = 0 then

vertex v is called a source, if outdegree (v) = 0 then v is a sink. A digraph is rooted if there is a vertex called root of the digraph, that reaches all its vertices.

Let D be an acyclic digraph. Its minimum spanning subgraph and maximum spanning supergraph which preserve the reachability of D are the transitive reduction and transitive closure of D, respectively. The transitive closure of D is also called the partially ordered set (poset) induced by D. We also write $v < w$ to denote that v reaches w in D, for $v \neq w$. In this case we say that v, w are comparable. If neither $v \not< w$ nor $w \not< v$ then v, w are incomparable. Observe that since v is acyclic if $v < w \Rightarrow w \not< v$. A topological ordering of D is a sequence $v_1, v_2, \ldots, v_n$ of all its vertices, such that if $v_i < v_j \Rightarrow i < j$. Clearly, if an acyclic digraph D is rooted then there is one vertex which is both the unique root and unique source of D.

A directed rooted tree T is a (acyclic) rooted digraph such that all its vertices except the root, have indegree one. If $v < w$ in T then v is an ancestor of w and w a descendant of v. If additionally $(v,w) \in E$ then v is a father of w and w a son of v. It can be shown that there is a unique path from the root r of T to any of its vertices. The level of a vertex v in T is equal to one plus the length of the path from r to v (the level of r is therefore equal to 1). A forest is a set of directed rooted trees.

The vertices of a graph can be traversed according to predefined rules, such as those of depth first search (DFS). A DFS of an undirected graph divides its edges into two disjoint subsets, the tree edges and fronds. A DFS of a digraph divides its edges into four disjoint subsets, the tree edges, forward edges, back edges and cross edges. See [1] for instance, for a description of DFS.

2. Characterization

We say that a directed acyclic graph is tree structured (TS) when its transitive reduction is a directed rooted tree. Figure 1 (a) is an example of such a digraph. Its transitive reduction is the tree of figure of 1 (b).

The following lemma presents some alternative ways of characterizing the class of tree structured digraphs.

Lemma 2.1

Let D (V,E) be an acyclic rooted digraph. The following statements are then equivalent:

(1) The transitive reduction of D is a directed rooted tree.

(2) There exists a spanning directed rooted tree T of D, such that for every edge $(v,w) \in E$, v is an ancestor of w in T.

(3) There exists a DFS of D with no cross edges.

(4) For any pair of edges (v,w), $(z,w) \in E$, v and z are comparable $(v \neq z)$.

(5) For any pair of vertices $v, w \in V$, with $v < w$, the vertices of any path from v to w are a subset of the vertices of the longest from v to w.

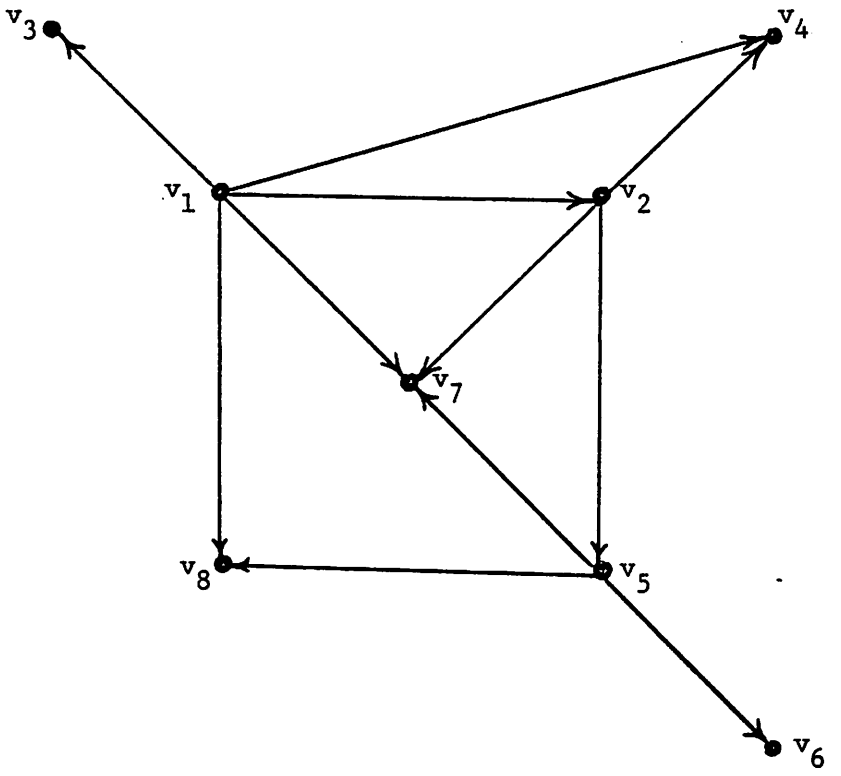
Proof:

(1) => (2):

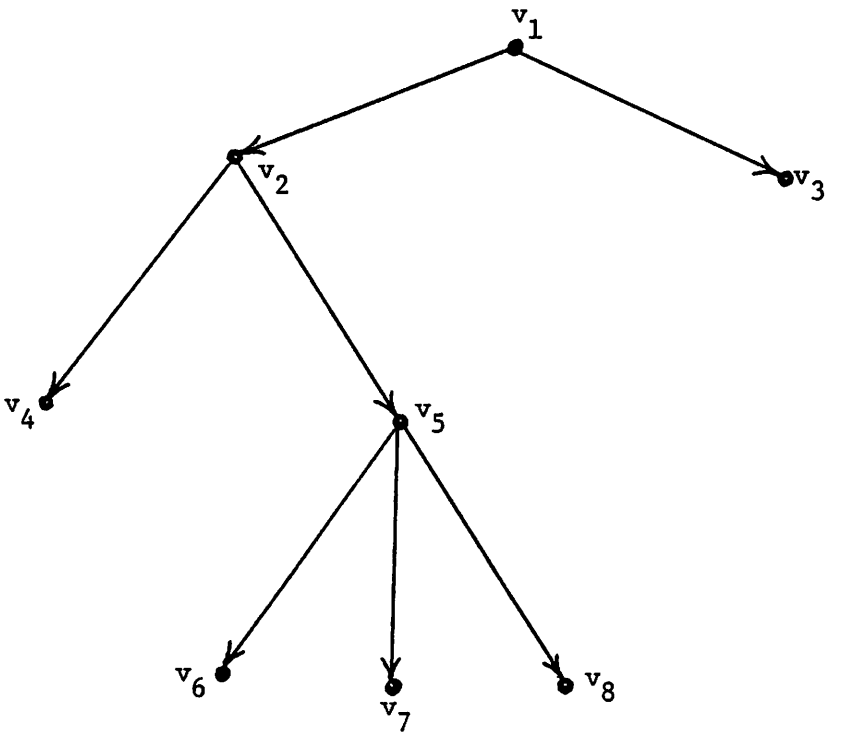
Let T' be the transitive reduction of D, which according to (1) is a directed rooted tree. Let (v,w) be an edge of D. If v is not an ancestor of w in T', this means that $v \not< w$ in D, a contradiction with (v,w) being an edge of D. Therefore, T = T'.

(2) => (3):

Consider the spanning tree T. Order the adjacency lists of D in



(a)



(b)

Fig. 1: A tree structured digraph and its transitive reduction

such a way that for every vertex v of D, if w, z $\in$ A(v) but (v,w) is in T and (v,z) is not in T, then w precedes z in A(v). Let r be the root of T. Starting with r, perform a DFS of D. Because of the considered ordering, it can be shown by an inductive argument that the DFS forest so obtained is exactly tree T. Finally, we also conclude that no cross edge (v,w) can exist in the DFS, otherwise v would not be an ancestor of w in T, a contradiction.

(3) => (4):

Consider a DFS of D, with no cross edges. Let (v,w) and (z,w) be edges of D. If w is reached before v in the mentioned DFS, since w $\not<$ z edge (v,w) would be a cross edge, a contradiction. Therefore, v is reached before w in the DFS. Similarly we conclude that z is reached before w. Therefore these vertices are reached in the order v,z,w or z,v,w. In the first case, there exists a path from v to z. In the second, from z to v. Hence v and z are comparable.

(4) => (5):

Let $v = z_1, \ldots, z_j, z_{j+1}, \ldots, z_p = w$ be the longest path P from v to w, in D. Suppose that there exists another path from v to w that contains a vertex t, t $\notin$ P. Clearly, $z_1 < t < z_p$. Let z_j be the rightmost vertex of P such that t $\not< z_j$ (such a vertex exists because t $\not< z_1$). Because $t < z_{j+1}$, there exists a path from t to z_{j+1}. This means that we can locate a vertex s, s $\notin$ P such that t < s and $(s,z_{j+1}) \in E$. Applying (4) to the pair of edges (z_j,z_{j+1}) and (s,z_{j+1}), we conclude that z_j and s are comparable. If $s < z_j$ then by transitivity $t < z_j$, a contradiction. Therefore, $z_j < s$, which means that there exists a path P' $z_j, y_1, \ldots, y_k$, s from z_j to s. P' cannot contain vertices of P except z_j, otherwise D would contain a cycle. Hence, the path

$z_1, \ldots z_j, y_1, \ldots, y_k, s, z_{j+1}, \ldots, z_p$ from v to w, is longer than P, a contradiction. Thus there can be no path from v to w containing vertices not belonging to P.

(5) => (1):

Let us consider the subset S of edges of D which are part of some longest path from v to w, for all pairs of vertices v, w v < w. Because of (5), we can delete all edges of E-S, without altering the transitive reduction of D. Therefore, the digraph D'(V,S) is the transitive reduction of D. It follows that D' is a directed rooted tree.

It is also possible to formulate a characterization of the present class of digraphs, by means of a specific forbidden subgraph of its transitive closure. The following is a direct consequence of lemma 2.1.

Lemma 2.2

An acyclic digraph D is tree structured if and only if its transitive closure does not contain the digraph of figure (2), as an induced subgraph.

We can also verify the existence of a relationship between the class of tree structured digraphs and a class of acyclic digraphs known as series parallel digraphs. The latter has been shown useful when solving some problems of scheduling under constraints. If the constraints correspond to a digraph of this class, it can be shown that more efficient algorithms can be devised (Lawler [11], for instance). Some graph theoretic properties of series parallel digraphs have been presented by Valdes [14] and Valdes, Tarjan and Lawler [15].

A minimal series parallel digraph (MSP) is defined recursively by

(i) a digraph with one vertex is MSP

(ii) if $D_1(V_1,E_1)$ and $D_2(V_2,E_2)$ are MSP digraphs, so are the digraphs $(V_1 \cup V_2, E_1 \cup E_2)$ and $(V_1 \cup V_2, E_1 \cup E_2 \cup (A_I \times B_2))$, where A_I, B_2 are the set of sinks of D_1 and sources of D_2, respectively.

A general series parallel digraph (GSP) is then defined as being a digraph whose transitive reduction is MSP. GSP digraphs can also be characterized by the fact that a digraph is GSP if its transitive closure does not contain figure (3) as an induced subgraph ([14], [15]). The following lemma relates the classes TS and GSP.

Lemma 2.3

Let D be a TS digraph. Then D is GSP.

Proof:

It follows immediately from the forbidden subgraphs of figures 2 and 3.

An alternative simple argument for the above lemma is the fact that if D is TS then its transitive reduction is a directed rooted tree, which is a MSP digraph. Then D is also GSP. The next lemma states that no such similar relation exists between the classes TS and MSP.

Lemma 2.4

The classes of tree structured and minimal series parallel digraphs are not contained one in the other.

Proof:

The digraph of figure 2 is MSP, but not TS. The digraph of figure 4 is TS, but not MSP.

Finally, it is also possible to relate the class of tree structured digraphs and the class of general undirected graphs. This can be done

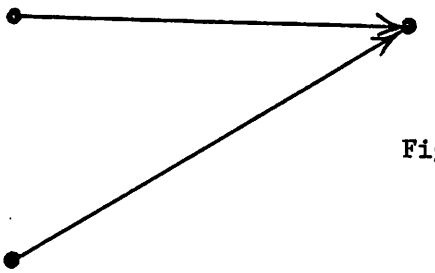


Fig. 2: The forbidden subgraph for the transitive closure of TS digraphs

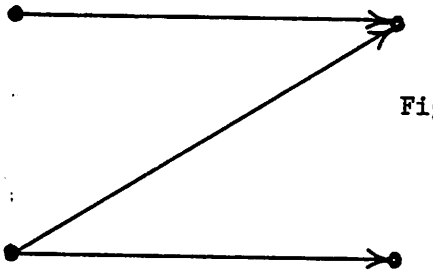


Fig. 3: The forbidden subgraph for the transitive closure of GSP digraphs

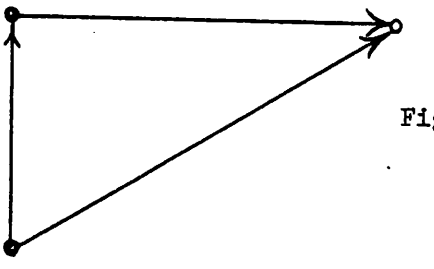


Fig. 4: A digraph which is TS but not MSP.

through the concept of depth first search, as follows. Let G be an undirected graph, in which a DFS has been performed. The DFS-number for a vertex v of G is simply the number corresponding to the order in which v has been first reached in the DFS. We can now formulate the mentioned relationship.

Lemma 2.5

Let a DFS be performed in an undirected connected graph G(V,E). Let D be the directed graph obtained from G by directing each edge of G from lower to higher DFS-numbers of its vertices. Then D is a tree structured digraph.

Proof:

Because all edges go from lower to higher numbers of its vertices, D is necessarily acyclic. Because a DFS of an undirected graph has no cross edges, all edges in D are from ancestors to descendants in the DFS spanning tree.

3. Recognition

Clearly, when handling special classes of graphs, a basic problem is to recognize whether or not a given graph is one that belongs to the class in consideration. The recognition of tree structured digraphs is a simple problem that can be solved efficiently using the following lemma.

Lemma 3.1

Let D(V,E) be a rooted digraph such that for every $v \in V$, A(v) has been topologically sorted. Then a DFS starting from the root of D has no cross edges if and only if D is tree structured.

Proof:

If the DFS has no cross edges then according to lemma 2.1, D is TS. Conversely, suppose D is TS and assume that a cross edge (v,w) has been detected. This means that during the DFS, vertex w has been reached before v. Let z be the lowest common ancestor of v and w, in the DFS (vertex z exists necessarily because the DFS is from the root of D). Let $z, z_1, \ldots, w$ and $z, z'_1, \ldots, v$ be the paths from z to w and v, respectively explored during the DFS. Because w has been reached before v, path $z, z_1, \ldots w$ has been explored before path $z, z'_1, \ldots v$. Also, $v < w$, $z_1 \leq w$ and therefore, we conclude that v and z are comparable, otherwise the forbidden subgraph of lemma 2.2 would have been found. Because z is the lowest common ancestor of v and w, we have $z \neq v$. Hence, $v < z_1$ which implies $z'_1 < z_1$. Since A(z) is topologically sorted, z'_1 must preceed z_1 in A(z), which contradicts the fact that path $z, z_1, \ldots, w$ has been explored before $z, z'_1, \ldots, v$. Hence no cross edges can be found in the DFS.

It should be noted that lemma 2.1 stated solely that if a digraph is tree structured, then there exists a DFS in which no cross edges exist, while lemma 3.1 specifies precisely which one is that DFS.

As a consequence of the above lemma, the recognition of tree structured digraphs can be done straightforwardly. Given digraph D(V,E) rooted at vertex r, we first topologically sort its adjacency lists. Afterwards, starting from vertex r, we perform a DFS of D. Digraph D is in class TS if the DFS has no cross edges. Deciding whether or not a given edge is a cross edge within a DFS can be done in constant time. Therefore, it is immediate to verify that the whole above process is completed in $O(n+m)$ time.

4. Transitive Reduction and Closure

Given a digraph D(V,E) supposedly belonging to class TS we wish to obtain its transitive reduction and closure. It can easily be verified that if D is indeed a tree structured digraph then the DFS tree obtained from lemma 3.1 is precisely the transitive reduction of D. This means that this problem can also be solved in $O(n+m)$ time. For finding the transitive closure digraph of D, we need just to observe that for each $v \in V$ the adjacency list A(v) of the transitive closure of D is formed exactly by the set of descendants of v, in the transitive reduction (directed rooted tree) of D. Consequently, the transitive closure can be obtained in time proportional to its size.

It should be noted that the computation of the transitive reduction and closure of GSP digraphs can be done also in linear time [15]. In particular the method by Valdes, Lawler and Tarjan would find the transitive reduction and closure of TS digraphs obviously in linear time. It might be mentioned however, that for tree structured digraphs the above described method is perhaps conceptually simpler.

5. Isomorphism

Determining the complexity of the general graph isomorphism problem is a well-known classical open problem. Polynomially bounded algorithms have been found for the isomorphism of some specific classes of graphs, as planar graphs [10], interval graphs [2], MSP digraphs [14,15]. The isomorphism of general series parallel digraphs however has been shown to be polynomially equivalent to the general graph isomorphism problem [14,15].

In the present section we present a solution to the isomorphism of a tree structured digraphs. The proposed method is based on the following lemma.

Lemma 5.1

Let $D_1(V_1,E_1)$ and $D_2(V_2,E_2)$ be TS digraphs, $T_1(V_1,E_{T_1})$ and $T_2(V_2,E_{T_2})$ their transitive reduction trees, respectively. For $i = 1, 2$ let $h(w_i)$ represent the level of vertex w_i in tree T_i. Suppose that a set $s(w_i)$ of labels is assigned to each vertex w_i of T_i, such that:

if w_i is the root of T_i then $s(w_i) = \phi$, otherwise
$s(w_i) = \{h(v_i) \mid (v_i,w_i) \in E_i\}$

Then D_1 and D_2 are isomorphic iff T_1 and T_2 are isomorphic labelled rooted trees.

Proof:

Suppose D_1 and D_2 are isomorphic. Then T_1 and T_2 are necessarily isomorphic directed trees. It remains to show that they are isomorphic also as labelled trees. The isomorphism of D_1, D_2 means that there exists a bijection $f: V_1 \simeq V_2$ s.t. $(v_1,w_1) \in E_1$ iff $(f(v_1), f(w_1)) \in E_2$. Consider now a fixed vertex $w_1 \in V_1$. Then $h(w_1) = h(f(w_1))$ and $h(v_1) = h(f(v_1))$ for all $v_1 \in V_1$ s.t. $(v_1,w_1) \in E_1$. Because D_1 is TS, v_1 is an ancestor of w_1 in T_1. Hence v_1 is uniquely determined by $h(v_1)$. We conclude then that $s(w_1) = s(f(w_1))$ and the same is true for all $w_1 \in V_1$.

From this last equality, we conclude that for each $w_1 \in V_1$, the levels in T_1 of the vertices v_1 s.t. $(v_1,w_1) \in E_1$ are respectively the same as the levels in T_2 of the vertices v_2 s.t. $(v_2,f(w_1)) \in E_2$. Since D_1 and D_2 are TS digraphs, v_1 and v_2 are uniquely determined by their levels in T_1, T_2 and by w_1, w_2 respectively. We conclude that

necessarily $v_2 = f(v_1)$. Hence $(v_1,w_1) \in E_1$ iff $(f(v_1),f(w_1)) \in E_2$ and D_1, D_2 are isomorphic.

From the above lemma, we then specify the actual algorithm. Given two tree structured digraphs $D_1(V_1,E_1)$ and $D_2(V_2,E_2)$ we first obtain their transitive reduction trees T_1 and T_2, respectively. Next for each $w_1 \in V_1$ we find the set $s(w_1)$, as indicated in lemma 5.1. Similarly, we find the label sets for all w_2 in T_2. Observe that each label set $s(w_i)$ is composed by elements $h(v_i)$ (levels), within the range $1 \leq h(v_i) < n$. Next we apply for instance an algorithm by Hopcroft and Tarjan [9] for labelled tree isomorphism and decide whether T_1, T_2 are isomorphic labelled rooted trees. From lemma 5.1 we know that D_1 and D_2 are isomorphic iff are T_1 and T_2.

An example of the labelled tree corresponding to the digraph of figure 1(a) is shown in figure 5. Obtaining the running time of the above method is also straightforward. From sections 3 and 4, we know that T_1 and T_2 can be constructed in $O(n+m)$ time. Computing all sets of labels would require also $O(n+m)$ operations. The algorithm by Hopcroft and Tarjan for labelled tree isomorphism is bounded by $O(n+L)$, where n is the number of vertices of each tree and L is the sum of the lengths of all label sets. Clearly, $L = m$ for each tree. Therefore, the entire process is bounded by $O(n+m)$.

As a consequence of the presented method for isomorphism, we can verify that any TS digraph is uniquely determined by its transitive reduction tree, together with the sets s(w) of labels, one set of labels per vertex of the tree, as defined by lemma 5.1.

Also for each vertex w of the transitive reduction tree T, $w \neq$ root of T, we can eliminate from s(w) its largest element h(v). Note that

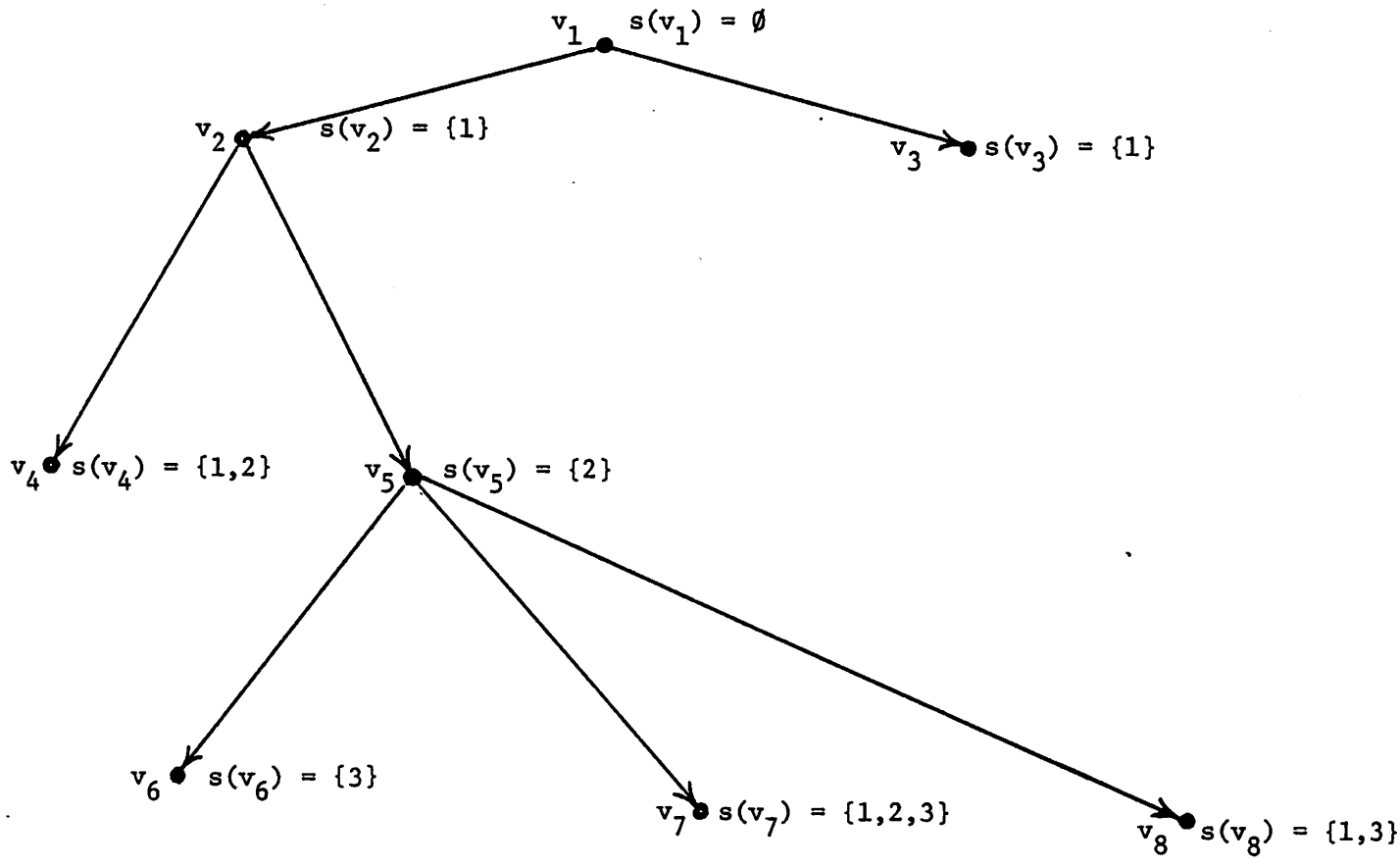


Fig. 5: A labelled tree representing the digraph of Fig. 1(a)

$$\max \{h(v) \mid h(v) \in s(w)\} = h(w) - 1, \text{ for } w \neq \text{root of } T.$$

6. Isomorphism of DFS

Suppose we perform a depth first search S_1 of an undirected graph G, starting from vertex r_1. Next, after perhaps changing the order in which the vertices of G appear in the adjacency lists and perhaps changing the start vertex to r_2, we perform another depth first search S_2 of G. The question that we would like to consider is whether or not S_1 and S_2 correspond to "similar" searches. More formally, we would say that "similar" DFS are isomporphic DFS, which are defined as below.

Two depth first searches S_1 and S_2 starting respectively from vertices r_1 and r_2 of an undirected graph G(V,E) are <u>isomorphic</u> when there exists a permutation f such that

(i) $r_2 = f(r_1)$ and

(ii) for every edge $(v,w) \in E$:

(v,w) is a tree edge in S_1 iff $(f(v),f(w))$ is a tree edge in S_2,

(v,w) is a frond edge in S_1 iff $(f(v),f(w))$ is a frond edge in S_2.

Clearly, tree edges of S_1 are mapped into tree edges of S_2 and fronds of S_1 are mapped into fronds of S_2. If two DFS are isomorphic their corresponding DFS trees are also, but this condition is not sufficient.

An example is shown in figure 6. Figure 6(a) is an undirected graph G, 6(b), 6(c) and 6(d) are three distinct depth first searches of G, with the tree edges corresponding to straight lines and fronds to curved ones. Clearly, the DFS of 6(b) and 6(c) are not isomorphic, while those of 6(b) and 6(d) are.

A solution to the problem of finding isomorphism of DFS can be based on the following lemma.

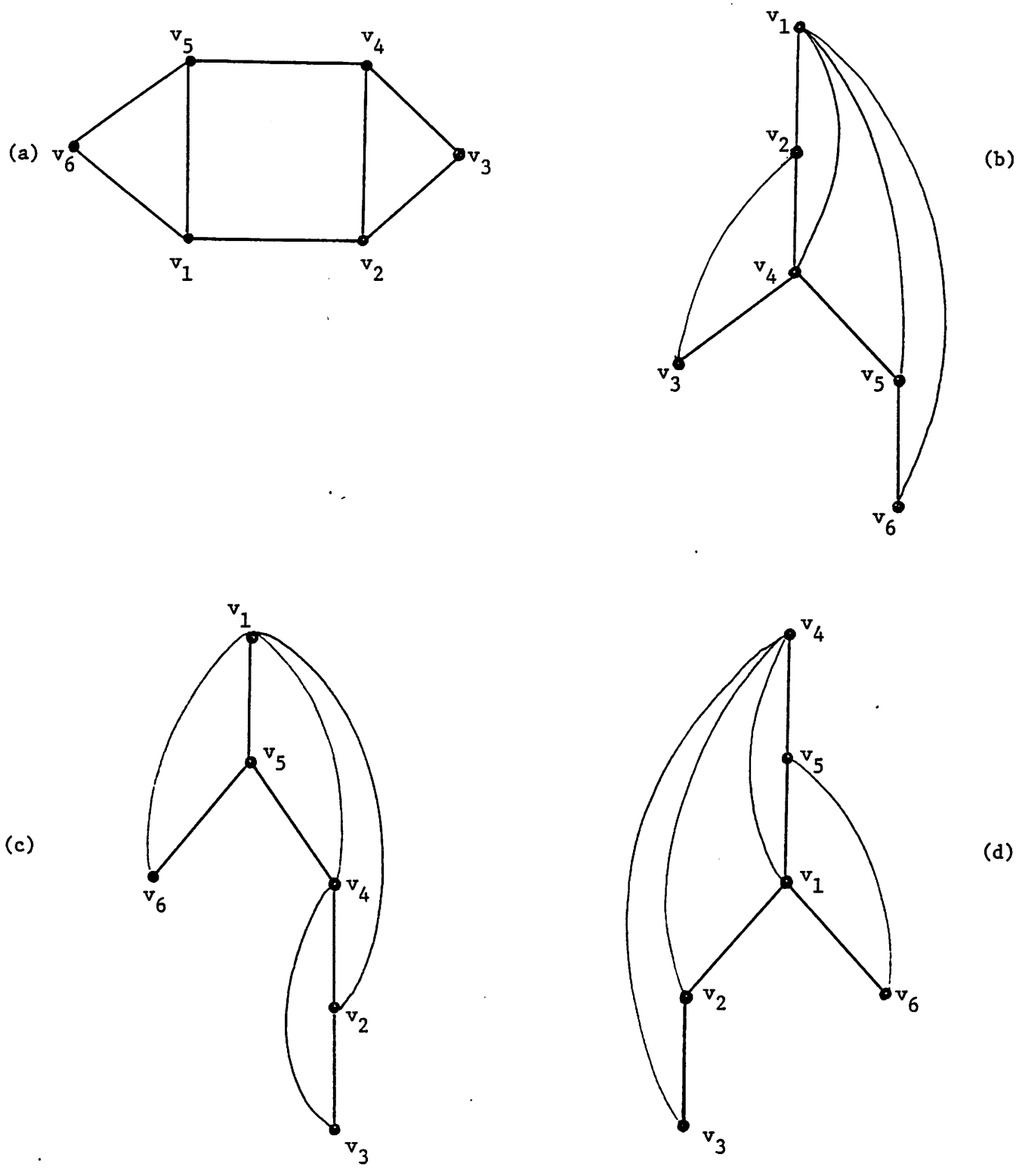


Fig. 6: An undirected graph and three possible DFS

Lemma 6.1

Let G be an undirected graph, S_1 and S_2 two depth first searches performed in G. Let D_1 and D_2 be the TS digraphs obtained by directing each edge of G from lower to higher DFS-numbers of its vertices, respectively. Then S_1 and S_2 are isomorphic DFS of G iff D_1 and D_2 are isomorphic (TS) digraphs.

The proof of the above lemma is straightforward and the actual algorithm is just an implementation of it. Given G and the two DFS S_1 and S_2, obtain the tree structured digraphs D_1 and D_2 as above, by directing each edge of G from low to high DFS-numbers, respectively. It should be noted that in this way, tree edges of S_1 are mapped into edges that belong to the transitive reduction of D_1, while fronds of S_1 are mapped into edges that are redundant under transitive reduction. This means that the DFS tree of S_1 corresponds precisely to the transitive reduction of D_1. The same of course, applies to S_2 and D_2, respectively. After obtaining D_1 and D_2 apply the method of section 5 to decide the isomorphism of these digraphs. Finally, S_1 and S_2 are isomorphic iff D_1 and D_2 are so. The running time of the algorithm can be clearly shown also to be bounded by $O(n+m)$.

As a consequence of the above fact, we can establish a further relation between undirected graphs and tree structured digraphs, as follows. Let α be the set of all (pairwise non-isomorphoric) undirected graphs. For each $G \in \alpha$ perform all possible DFS. Consider now a relation R such that for every pair S_1, S_2 of DFS of G, $S_1 R S_2$ iff S_1, S_2 are isomorphic. Clearly R is an equivalence relation and its equivalence classes have the following property:

Lemma 6.2

There exists a one-to-one correspondence between tree structured digraphs and the equivalence classes of relation R.

Now we say that an undirected graph G generates an equivalence class C of relation R, when there exists a DFS of G which is a member of C. We can then conclude that the number of distinct equivalence classes C of R generated by a fixed graph G equals the number of distinct tree structured digraphs whose underlying undirected graphs are all isomorphic to G.

7. Minimal Chain Decomposition

Let D(V,E) be an acyclic digraph. A chain of D is a sequence of vertices which is a path in the transitive closure of D. The minimal chain decomposition problem consists of finding a minimal set of chains which covers the vertices of D. It is known that this problem can be solved by reducing it to a network flow problem. The basis of a solution for it is the well-known theorem by Dilworth [3], which states that the minimum number of chains that cover D equals the maximum number of incomparable vertices of D.

We now restrict our attention to cases in which D is a tree structured digraph.

Lemma 7.1

Let D be a TS digraph. Then the maximum number of incomparable vertices of D equals the number of sinks of D.

The proof is obvious. For actually finding a minimal chain decomposition set $\{c_1, \ldots, c_k\}$ of D we initially determine the transitive

reduction tree T of D. The first chain c_1 in the set is a path from the root of T to any of its sinks. Next we delete from T the vertices of c_1. This may turn $T-c_1$ into a forest. Choose any tree T' of $T-c_1$ and chain c_2 is a path from the root of T' to any of its sinks. Delete c_2 from $T-c_1$ and apply the same operation to $(T-c_1)-c_2$. Proceed so until the forest becomes empty. Since the deletion of those chains cannot create new sinks, the total number of chains obtained by the above process equals the number of sinks of D, which means that $\{c_1, \ldots, c_k\}$ is indeed minimal.

It is also immediate to verify that the set of chains can be obtained from T in $O(n)$ time and therefore the entire process is bounded by $O(n+m)$.

It should be also observed that the chains obtained by the above method are in fact paths of D. This means that for TS digraphs D the problem of obtaining a minimal set of chains that covers the vertices of D is equivalent to the problem of finding a minimal set of disjoint paths whose vertices cover D. These two problems are clearly the same when D is its own transitive closure, but otherwise may have different solutions.

As an example, we can see that a solution of the minimal chain decomposition problem obtained by the described process for the digraph of figure 1(a) is

$$\{(v_1,v_2,v_4),\ (v_3),\ (v_5,v_6),\ (v_7),\ (v_8)\}.$$

8. The Dimension of the Induced Poset

Let D(V,E) be an acyclic digraph. We can characterize the poset induced by D (i.e. the transitive closure of D) through a set of k topological orderings of D, such that $v < w$ iff v precedes w in all of

the k orderings, for all v, w $\in$ V. In other words, v and w are not comparable in D iff there are two orderings t_1, t_2 such that v precedes w in t_1 and w precedes v in t_2. The minimum value of such k is the dimension of the poset. It is an open problem determining the complexity of finding the dimension of a poset [5]. In this section we consider the problem of finding a minimum set of topological orderings which define a poset induced by a TS digraph.

Let T be a rooted tree. In a preorder traversal of T we visit the root of T and afterwards, recursively visit the sons of T. If those sons are visited in order left to right, we call it a left preorder; if the visits are from right to left, we have a right preorder.

Lemma 8.1

Let D(V,E) be a TS digraph and T its transitive reduction tree. If p_1, p_2 are the left and right preorder traversals of T respectively, then $\{p_1,p_2\}$ is a set of topological orderings that completely characterizes the poset induced by D.

Proof

Clearly p_1, p_2 are topological orderings of D, since any preorder traversal of T has this property. Now let v, w $\in$ V. If v < w, then v is an ancestor of w in T and therefore v precedes w in both p_1, p_2. If v, w are incomparable, a simple inductive argument shows that v, w appear in different relative positions in p_1 and p_2.

A consequence of the above lemma is the fact that a poset specified by a TS digraph is necessarily two-dimensional (except of course, the case $p_1 = p_2$ which corresponds to the poset being a total ordering).

And the actual topological orderings $\{p_1,p_2\}$ can be obtained in linear time. However, similar results have been obtained also for the more general class of GSP digraphs [15]. As for the previous case of transitive reduction and closure (Section 4), we mention that the above solution restricted to TS digraphs is perhaps simpler than that for the GSP class.

9. An Extension

In the present section, we show that it is possible to enlarge the class of digraphs for which the isomorphism problem may be solved using essentially the ideas of Section 5. Some introductory definitions are needed.

Let D be a digraph (possibly with cycles) rooted at r. Let S be a DFS of D, starting from r. Denote by B_S the set of back edges obtained from S. Digraph D is then called reducible when B_S is always the same, independent of S. Reducible digraphs have been characterized by Hecht and Ullman [7, 8] and they can be recognized in time slightly more than linear (Tarjan [12, 13]). Now denote by D(S) the acyclic digraph obtained from D by deleting the edges of B_S, i.e. D(S) is the digraph $(V,E-B_S)$. A reducible digraph D will then be called tree reducible when D(S) is a tree structured digraph. An example of a tree reducible digraph is shown in figure 7.

If D is an acyclic digraph then clearly $B = \phi$ and $D(S) = D$, for any DFS S. Therefore, the class of reducible digraphs contains all acyclic digraphs. This means that there is no hope in solving the isomorphism problem for reducible digraphs, without solving also isomorphism of general graphs. However, when D is tree reducible, there is a simple way of verifying it.

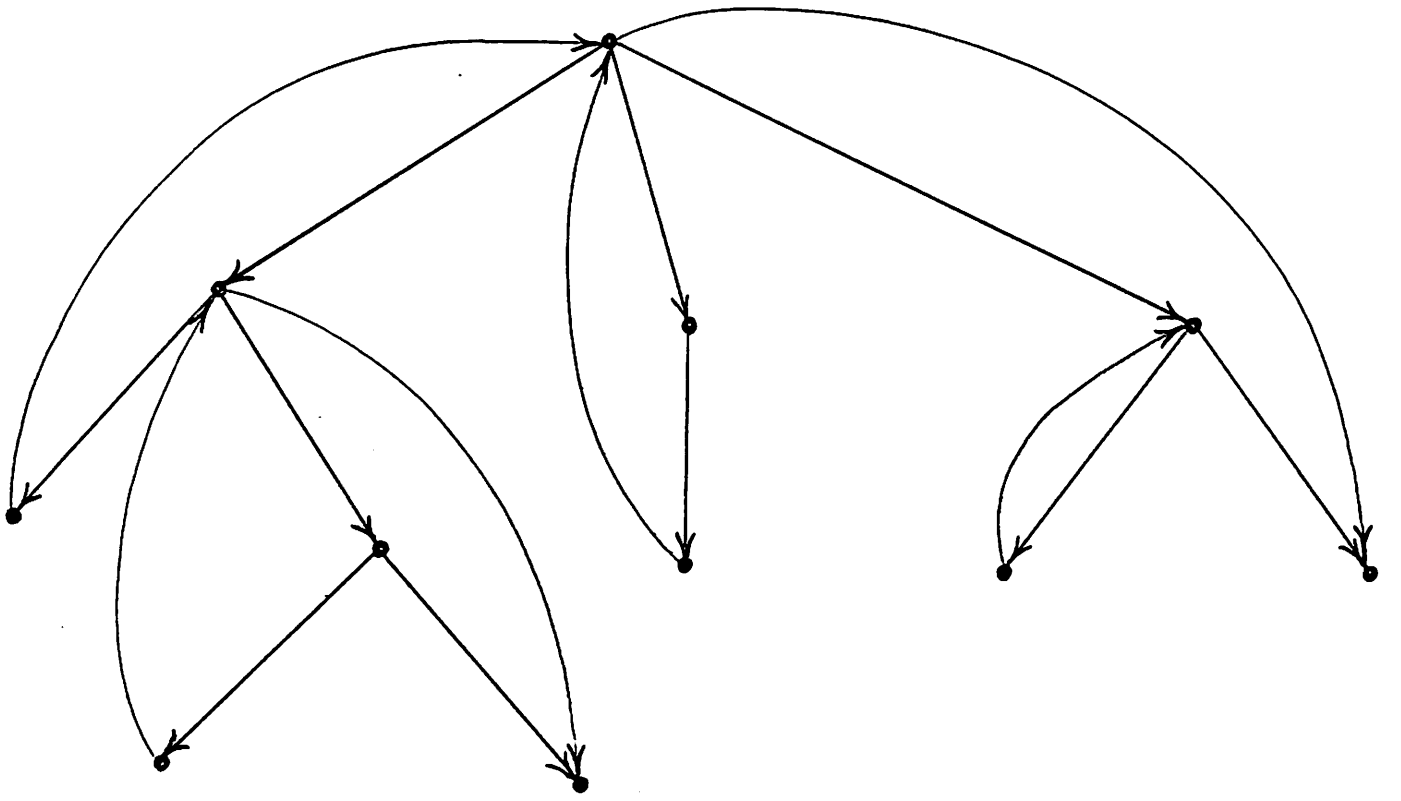


Fig. 7: A tree reducible digraph

We recall that the isomorphism of TS digraphs described in Section 5, was based essentially in two ideas:

(i) if D is a TS digraph, there is an efficient way of uniquely finding a special spanning tree T (the transitive reduction of D).

(ii) all edges (v,w) of D are such that w is a descendant of v in T.

Now, if D is tree reducible, we can also find unique and efficiently a special spanning tree T (the transitive reduction of D(S), for some DFS S). And all edges (v,w) of D are such that one between v, w is a descendant of the other. In any case, for every edge (v,w) of D if we fix vertex v, then w can be uniquely determined by its level in T and by the position of v in T. These ideas lead to the following algorithm for isomorphism of tree reducible digraphs.

Given digraphs $D_1(V_1,E_1)$ and $D_2(V_2,E_2)$ rooted at r_1 and r_2, respectively:

1. Recognize if D_1, D_2 are in fact both reducible digraphs [13]. Stop in case of negative answer.
2. Perform a DFS S_1 for D_1, rooted at r_1. Perform a DFS S_2 for D_2, rooted at r_2.
3. Recognize if D_1 (S_1), $D_2(S_2)$ are in fact both TS digraphs. Stop in case of negative answer.
4. Find the transitive reduction tree T_1 of $D_1(S_1)$ and T_2 of $D_2(S_2)$, respectively. For $i = 1, 2$ let $h(w_i)$ represent the level of vertex w_i of T_i. To each vertex w_i of T_i assign a set of labels $s(w_i)$ defined by

$$s(w_i) = \{h(v_i) \mid (v_i,w_i) \in E_i\}$$

5. Verify whether T_1, T_2 are isomorphic as labelled rooted trees [9]. D_1, D_2 are isomorphic iff T_1, T_2 so are.

Clearly if an application of the algorithm stops at 1 or 3, then D_1, D_2 are not both tree reducible digraphs. Otherwise, it will produce the correct answer. The time bound of step 1 is slightly more than linear. The remaining steps can be performed in linear time.

Finally, we remark that all sets $s(w_i)$ of lemma 5.1 contained only integers (levels) smaller than $h(w_i)$, the level of w_i in T_i. In the present case, the integers of $s(w_i)$ may be smaller or greater than $h(w_i)$. Ther greater ones correspond to back edges entering w_i.

10. A Problem

The following problem is related to the class of tree structured digraphs:

"Given an acyclic digraph D(V,E) rooted at r and an integer k, is there a DFS of D having at most k cross edges?"

For the two particular cases below, we can find efficient polynomial time solutions:

(i) if k = 0 then the problem is equivalent to the recognition of TS digraphs (Section 3).

(ii) if D is a transitive reduction digraph (i.e. if $v < w$ and $w < z \Rightarrow (v,z) \notin E$), then the minimum value of k such that it is possible to perform a DFS of D with exactly k cross edges is

$$k = \sum_{v \neq r} (\text{indegree } (v) - 1)$$

Observe that the problem in consideration can be also described as "find a maximal spanning subgraph of D, which is tree structured".

11. Conclusions

The class of tree structured digraphs has been presented. It is formed by all acyclic digraphs whose transitive reduction is a directed rooted tree. Alternative characterizations have also been discussed and relationships between the mentioned class and DFS (of both directed and undirected graphs) have been emphasized, in some different aspects.

Although the isomorphism problem for tree structured digraphs has been solved in a simple and efficient way, subgraph isomorphism has not been considered. It would be interesting to know whether remains NP-complete, the problem of verifying if a digraph D_1 contains a subgraph isomorphic to digraph D_2, in case where D_1 is a TS digraph and D_2 a directed rooted tree.

REFERENCES

1. Aho, A.V., Hopcroft, J.E. and Ullman, J.D., The Design and Analysis of Computer Algorithms, Addison-Wesley, Reading, MA, 1974.

2. Booth, K.S. and Lueker, G.S., Linear Algorithms to Recognize Interval Graphs and Test for the Consecutive Ones Property, Proc. 7th Ann. ACM Symp. on Theory of Computing, New York, N.Y. 1975, pp. 225-265.

3. Dilworth, R.P., A Decomposition Theorem for Partially Ordered Sets, Ann. Math., 51 (1950), pp. 161-166.

4. Even, S., Pnueli, A. and Lempel, A., Permutation Graphs and Transitive Graphs, J. ACM, 19 (1972), pp. 400-410.

5. Garey, M.R. and Johnson, D.S., Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.

6. Gavril, F., Algorithms for Minimum Coloring, Maximum Clique, Minimum Covering by Cliques and Maximum Independent Set of a Chordal Graph, SIAM J. Comp., 1 (1972), pp. 180-187.

7. Hecht, M.S. and Ullman, J.D., Flow Graph Reducibility, SIAM J. Comp., 1 (1972), pp. 188-202.

8. Hecht, M.S. and Ullman, J.D., Characterizations of Reducible Flow Graphs, J. ACM, 21 (1974). pp. 367-375.

9. Hopcroft, J.E. and Tarjan, R.E., Isomorphism of Planar Graphs, in Complexity of Computer Computations, R.E. Miller and J.W. Thatcher, eds., Plenum Press, New York, NY, 1972, pp. 131-152.

10. Hopcroft, J.E. and Wong, J.K., Linear Time Algorithm for Isomorphism of Planar Graphs, Proc. 6th Ann. ACM Symp. of Theory of Computing, New York, NY, 1974, pp. 172-184.

11. Lawler, E.L., Sequencing Jobs to Minimize Total Weighted Completion Time Subject to Precedence Constraints, Ann. Discrete Math., 2 (1978), pp. 75 -90.

12. Tarjan, R.E., Efficiency of a Good but not Linear Set Union Algorithm, J. ACM, 22 (1975), pp. 215-225.

13. Tarjan, R.E., Testing Flow Graph Reducibility, J. Comp. System Sciences, 9 (1974), pp. 355-365.

14. Valdes, J., Parsing Flowcharts and Series Parallel Graphs, Tech. Rep. STAN-CS-78-682, Comp. Science Dept., Stanford Univ., Stanford, CA, 1978.

15. Valdes, J., Tarjan, R.E. and Lawler, E.L., The Recognition of Series Parallel Digraphs, Proc. 11th Ann. ACM Symp. on Theory of Computing, Atlanta, Georgia, 1979.