ELSEVIER

Discrete Applied Mathematics 59 (1995) 267–283

DISCRETE APPLIED MATHEMATICS

The arborescence-realization problem

R.P. Swaminathan[a,*], Donald K. Wagner[b,1]

[a]*Department of Computer Science, University of Cincinnati, Cincinnati, OH 45221, USA*
[b]*Mathematical Sciences Division, Office of Naval Research, Arlington, VA 22217, USA*

Received 8 April 1992; revised 25 June 1993

Abstract

A $\{0, 1\}$-matrix M is *arborescence graphic* if there exists an arborescence T such that the arcs of T are indexed on the rows of M and the columns of M are the incidence vectors of the arc sets of dipaths of T. If such a T exists, then T is an *arborescence realization* for M. This paper presents an almost-linear-time algorithm to determine whether a given $\{0, 1\}$-matrix is arborescence graphic and, if so, to construct an arborescence realization. The algorithm is then applied to recognize a subclass of the extended-Horn satisfiability problems introduced by Chandru and Hooker (1991).

1. Introduction

A $\{0, 1\}$-matrix M is *arborescence graphic* if there exists an arborescence T such that the arcs of T are indexed on the rows of M and the columns of M are the incidence vectors of the arc sets of dipaths of T. If such a T exists, then T is an *arborescence realization* for M. (Note that not every dipath of T need correspond to a column of M.) The *arborescence-realization problem* is to determine whether a given $\{0, 1\}$-matrix is arborescence graphic and, if so, to construct an arborescence realization. An algorithm is presented to solve the arborescence-realization problem; the complexity of the algorithm is $O(\alpha(n,r)n)$, where n is the number of nonzero entries and r is the number of rows of the given $\{0, 1\}$-matrix and $\alpha(r,n)$ is a functional inverse of Ackermann's function. The function $\alpha(n,r)$ is extremely slow growing, and for all "practical" values of n and r, $\alpha(n,r) \leqslant 4$; see Tarjan [17] for details.

The arborescence-realization problem has a number of applications. Ball et al. [1] describe a reliability covering problem that is NP-hard in general. They give a polynomial-time algorithm for solving the problem in the case that a certain matrix

*Corresponding author.
[1]Research partially supported by Office of Naval Research Grant N00014-86-K-0689 at Purdue University.

SSDI 0166-218X(93)E0170-4

associated with the problem is arborescence graphic. Thus, the algorithm of this paper can be used to recognize this special case of the reliability problem.

The arborescence-realization problem also arises in information-retrieval problems; see, for example, Lipski [14]. In this context, the rows of a $\{0, 1\}$-matrix M correspond to pieces of information, or records, and the columns of M are the incidence vectors of prescribed subsets of related records. The arborescence-realization problem corresponds to determining a tree-structured storage configuration in which the prescribed subsets of related records are stored contiguously. Contiguous storage allows for efficient retrieval of the related items.

A third application arises in propositional logic. Chandru and Hooker [6] describe a class of satisfiability problems that can be solved in linear time; in general, the satisfiability problem is NP-complete. This class, called *extended Horn*, generalizes the well-studied Horn class. In Section 6, a polynomial-time algorithm is given to recognize a subclass of extended Horn, which also generalizes Horn. This algorithm uses, as a subroutine, an algorithm for solving the arborescence-realization problem.

The arborescence-realization problem generalizes the consecutive-ones problem, which is that of determining whether the rows of a given $\{0, 1\}$-matrix can be permuted such that in the resulting matrix, the 1's in each column are consecutive. In particular, a matrix has the consecutive-ones property if and only if it is arborescence graphic and has an arborescence realization that is a dipath, which is true if and only if the given matrix, augmented with a column of all 1's, is arborescence graphic.

A vertex version of the arborescence-realization problem is posed by Truszczyński [19], who also provides a polynomial-time algorithm for the problem. In an unpublished paper, Dietz et al. [8] give a linear-time algorithm. In Section 5 of this paper, Truszczyński's problem is shown to be equivalent to the arborescence-realization problem. Thus, the Dietz–Furst–Hopcroft algorithm also solves the arborescence-realization problem. Comparisons between the Dietz–Furst–Hopcroft algorithm and the one presented in this paper are given in Section 5.

The remainder of the paper is organized as follows. Section 2 provides background material, Section 3 states the arborescence-realization algorithm, and Section 4 discusses some details of the algorithm. Section 5 describes the relationship of the arborescence-realization problem to its vertex analogue. Finally, Section 6 describes an algorithm for recognizing a subclass of extended-Horn problems.

2. Definitions

Undefined graph-theory terminology and notation is consistent with Bondy and Murty [4]. For convenience, cycles, paths, and trees of a graph are equated with their edge sets, and the arc set of a directed graph (or digraph) is equated with the edge set of its underlying (undirected) graph. All graphs and digraphs considered in this paper are loopless.

An *arborescence* is a digraph T, the underlying graph of which is a tree, that has exactly one vertex of indegree zero; this vertex is the *root* of T. Note that every vertex of T other than the root has indegree exactly one. If (u, v) is an arc of T, then u is the *parent* of v and v is a *child* of u. A digraph D is *strongly connected* if there exists a (u, v)-dipath for every pair of vertices u and v of D.

All matrices considered in this paper are assumed to have at least one nonzero entry in every row and column. Clearly this is not a restrictive assumption with respect to the arborescence-realization problem. For algorithmic purposes, matrices are assumed to be stored in *column-list form*, which means that for each column, the rows having nonzeros in that column are given in a linked list.

3. The main algorithm

Let M be an arborescence-graphic matrix, and let T be an arborescence realization of M. The matrix M can be represented by a digraph D constructed by adding an arc to T joining the ends of each dipath that corresponds to a column of M such that the dipath plus the arc form a dicycle. Then, T is spanning tree of D.

Given an arbitrary $\{0, 1\}$-matrix M, the algorithm below determines whether M is arborescence graphic by attempting to construct a digraph D having spanning tree T that is an arborescence realization of M and such that D and T are related as in the previous paragraph. The algorithm consists of three main steps, which are outlined after three definitions that facilitate the discussion.

Let G be a graph, and let T be a spanning tree of G. The pair (G, T) is called a *graph-tree pair* (abbreviated *gt-pair*). Each cycle of G that contains exactly one edge not in T is a *fundamental* cycle of (G, T). Each path of T of the form $C - \{e\}$, where C is a fundamental cycle of (G, T) and $e \notin T$ is a *fundamental* path of (G, T).

The first step of the algorithm constructs a gt-pair (G, T), if one exists, such that the columns of M correspond to the fundamental paths of (G, T). If M is arborescence graphic, then such a gt-pair necessarily exists.

If M is arborescence graphic, then there also exists a digraph D having T as a spanning tree such that the columns of M correspond to the fundamental paths of (D, T) and such that every fundamental cycle of (D, T) is a dicycle. The second step of the algorithm attempts to find such a digraph by appropriately orienting G. It is proved that if M is arborescence graphic, then G has such as orientation.

If (D, T) is the output of the second step and T is an arborescence of D, then T is an arborescence realization of M. If not, the third step attempts to modify (D, T) so as to turn T into an arborescence. Any such modification must preserve the fundamental dicycles of (D, T) which suggests the relevance of the notions of 2-isomorphism introduced by Whitney [21] for graphs and Thomassen [18] for digraphs.

Let $D = (V, A)$ be a 2-connected digraph. Let $F \subseteq A$ be such that $|V(D[F]) \cap V(D[A - F])| = 2$, and let u and v be the vertices common to $D[F]$ and

$D[A - F]$. Define D' to be the digraph obtained from D by interchanging the incidences of u and v in $D[F]$ and reversing the orientations of the arcs in F. Then, D' is obtained from D by the *reversal* of $D[F]$. Observe that D and D' have the same set of dicycles. Also observe that any spanning tree of D is also a spanning tree of D'. In general, a digraph D'' is *2-isomorphic* to a digraph D if D'' is obtained from D by a sequence of reversals. For (undirected) graphs, *reversal* and *2-isomorphism* are defined as above, but without any reference to the orientation of the arcs.

A final observation is needed before stating the algorithm. If the rows and columns of the matrix M can be permuted so that it admits a *block decomposition* as shown in Fig. 1, then the arborescence-realization problem can be decomposed. In particular, it is easy to verify that M is arborescence graphic if and only if its blocks $M_1, \ldots, M_t$ are arborescence graphic. Moreover, the blocks can be computed in time linear in the number of nonzeros of M; see, for example, Bixby and Wagner [2]. Therefore, it can be assumed that M has no block decomposition into two or more blocks; such a matrix is *connected*. A consequence of this assumption is that if (G, T) is a gt-pair such that the columns of M correspond to the fundamental paths of (G, T), then G is 2-connected; again, see Bixby and Wagner [2].

Algorithm ARBORESCENCE REALIZATION.
Input: A connected $r \times c$ $\{0,1\}$-matrix M given in column-list form.
Output: An arborescence realization T of M or the conclusion that M is not arborescence graphic.
Step 1: Construct a gt-pair (G, T), if one exists, such that the columns of M are the incidence vectors of the fundamental paths of (G, T); if no such gt-pair exists, conclude that M is not arborescence graphic and stop.
Step 2: Construct an orientation D of G, if one exists, such that each fundamental cycle of (D, T) is a dicycle; if no such orientation exists, conclude that M is not arborescence graphic and stop.
Step 3: Construct a digraph D' 2-isomorphic to D, if one exists, in which T is an arborescence; if no such digraph exists, conclude that M is not arborescence graphic and stop; otherwise, output the arborescence T and stop.

From the previous discussion, it is clear that if the algorithm outputs an arborescence T, then T is an arborescence realization of M. What needs to be shown is that if

$$M = \begin{pmatrix} M_1 & 0 & \cdots & & 0 \\ 0 & \cdot & & & \\ \vdots & & \cdot & & \vdots \\ & & & \cdot & 0 \\ 0 & & \cdots & 0 & M_t \end{pmatrix}$$

Fig. 1. A block decomposition of M.

the algorithm declares that M is not arborescence graphic, then this is indeed the case. The proof of the algorithm's correctness uses results of Whitney [21] and Thomassen [18] on 2-isomorphism, which appear as Theorems 1 and 2 below.

Theorem 1. *Let G and G' be 2-connected graphs on the same edge set. Then, G and G' have the same set of cycles if and only if they are 2-isomorphic.*

The *cycle space* of a digraph is the vector space generated by the incidence vectors of the cycles of the underlying graph using modulo-2 arithmetic. For simplicity, in this context, cycles are equated with their incidence vectors.

Theorem 2. *Let D and D' be strongly connected 2-connected digraphs on the same arc set that have the same cycle space. If there exists a set of dicycles common to D and D' that generates this space, then D and D' are 2-isomorphic.*

Lemma 3. *Let D be a 2-connected digraph, and let T be a spanning tree of D. If every fundamental cycle of (D,T) is a dicycle, then D is strongly connected.*

Proof. If D is not strongly connected, then there exists vertices u and v for which no (u,v)-dipath exists. Let U be the set of vertices that are reachable from u. Since D is connected, there exists an arc e that has its head, but not its tail, in U. Since D is 2-connected, e is in some fundamental cycle C of (D,T). Since C is a dicycle, it has an arc that has its tail, but not its head, in U, a contradiction. □

Theorem 4. *Algorithm* ARBORESCENCE REALIZATION *is correct.*

Proof. As observed earlier, if M is arborescence graphic, then there exists gt-pair (G,T) such that the columns of M correspond to the fundamental paths of (G,T). Thus, the stopping criterion in Step 1 is correct.

Suppose that the algorithm incorrectly stops in Step 2, and let T be an arborescence realization of M. Construct a digraph D' by adding an arc joining the ends of each dipath of T that corresponds to a column of M such that the dipath plus the arc form a dicycle. Let G' be the underlying graph of D'. Then, (G,T) and (G',T) have the same set of fundamental cycles (assuming an appropriate choice of edge names). Since a set of fundamental cycles determines all of the graph's cycles, Theorem 1 implies that G can be obtained from G' by a sequence of reversals. Any reversal in a sequence taking G' to G corresponds to a reversal in a sequence taking D' to some digraph, say D, that is an orientation of G. Moreover, since reversals preserve dicycles, every fundamental cycle of (D,T) is a dicycle, a contradiction.

Finally, suppose that the algorithm incorrectly stops in Step 3 with the conclusion that M is not arborescence graphic, and again let T be an arborescence realization of M. As in the previous paragraph, construct the pair (D',T). Then, (D,T) (the output of Step 2) and (D',T) have the same fundamental cycles. Thus, D and D' have the same cycle space. Moreover, by construction, every fundamental cycle is a dicycle of both D and D'. By Lemma 3, D and D' are strongly connected. Therefore, by Theorem 2, D and D' are 2-isomorphic, a contradiction. □

The problem described in Step 1 of the algorithm is sometimes called the *graph-realization* problem. Several algorithms exist for its solution, the most efficient of which are those of Bixby and Wagner [2] and Fujishige [10]. Each of these algorithms requires $O(\alpha(n,r)n)$ time, where n is the number of nonzeros of M, r is the number of rows of M, and $\alpha(n,r)$ is a functional inverse of Ackermann's function.

The orientation desired in Step 2, if it exists, can be found as follows. Compute a spanning tree S of the bipartite graph B associated with M. (B has a vertex for each row and column of M and an edge for each nonzero entry.) Note that the vertex set of B coincides with the edge set of G. Arbitrarily choose a vertex s of B. Partition the remaining vertices into levels: vertex u is at *level* i if the (s,u)-path in S has exactly i edges. Now arbitrarily assign an orientation to the edge s of G. Assuming the desired orientation of G exists, the orientation assigned to the edge s uniquely determines an orientation for each edge at level 1. More generally, if orientations at level k have been assigned, then the orientations at level $k+1$ are uniquely determined. Thus, by proceeding level by level, either the desired orientation of G is found or it is concluded no such orientation exists. Construction of S and its levels requires $O(n)$ time using breadth-first search. Assigning orientations requires examination of each fundamental cycle of (G, T) exactly once, and so also takes $O(n)$ time.

Step 3, which is more involved, is covered in the next section.

4. Step 3 of the main algorithm

The main tool used to solve the problem described in Step 3 is a graph decomposition first introduced by Tutte [20] and studied further by Cunningham and Edmonds [7] and Hopcroft and Tarjan [12]. This decomposition is a convenient way to "display" all graphs 2-isomorphic to a given graph. The first part of this section describes a modification of this decomposition.

The decomposition is based on the 2-separations of a graph. Let $G = (V, E)$ be a 2-connected graph. A *2-separation* of G is a partition $\{E_1, E_2\}$ of E such that $|E_1| \geqslant 2 \leqslant |E_2|$ and $|V(G[E_1]) \cap V(G[E_2])| = 2$. If G has no 2-separation, then it is *3-connected*. (This definition of 3-connected differs slightly from that of Bondy and Murty [4] in that loops and parallel edges are not allowed.)

Let D be a 2-connected digraph, and let T be a spanning tree such that each fundamental cycle of (D, T) is a dicycle; such a pair is an *oriented gt-pair*. Let $\{E_1, E_2\}$ be a 2-separation of D, and let $\{u, v\} := V(D[E_1]) \cap V(D[E_2])$. Then, either $T \cap E_1$ is a spanning tree of $D[E_1]$ or $T \cap E_2$ is a spanning tree of $D[E_2]$; assume the former and set $T_1 := T \cap E_1$. Define D_1 by adding a new arc e to $D[E_1]$ joining u and v such that

(D_1, T_1) is an oriented gt-pair. (It is not hard to check that this is always possible.) Define D_2 by adding the same arc e to $D[E_2]$ such that its orientation in D_2 is opposite that in D_1, and set $T_2 := (T \cap E_2) \cup \{e\}$. Then, it is not difficult to verify that (D_2, T_2) is also an oriented gt-pair. Define the pair $\{(D_1, T_1), (D_2, T_2)\}$ to be the *simple decomposition* associated with the 2-separation $\{E_1, E_2\}$. The arc e is a *marker* arc of D_1 and D_2. A *decomposition* of (D, T) is inductively either the set $\{(D, T)\}$ or the set obtained from a decomposition of (D, T) by replacing some member S of the decomposition by the members of a simple decomposition of S.

Let $\mathscr{D} = \{(D_1, T_1), \ldots, (D_t, T_t)\}$ be a decomposition of (D, T), and let $G_1, \ldots, G_t$ and G be the respective underlying graphs of $D_1, \ldots, D_t$ and D. Then, $\mathscr{G} := \{G_1, \ldots, G_t\}$ is a *decomposition* of G. Of particular interest is when every member of $\mathscr{G}$ is either 3-connected, a bond (a connected loopless graph on two vertices), or a polygon (a connected graph, every vertex of which has degree two) and there does not exist any two bond or any two polygon members of $\mathscr{G}$ that have a marker edge in common. In this case, $\mathscr{G}$ is a *3-decomposition* of G and $\mathscr{D}$ is a *3-decomposition* of (D, T). Cunningham and Edmonds [7] proved that every graph has a unique 3-decomposition up to the choice of the names of the marker edges.

Associated with a decomposition $\mathscr{D}$ of an oriented gt-pair is a *decomposition tree* $\mathscr{T}$; the vertices of $\mathscr{T}$ are the members of $\mathscr{D}$ and two vertices are adjacent if and only if they have a marker arc in common. Observe that a subtree $\mathscr{S}$ of $\mathscr{D}$ is a decomposition tree of some decomposition, say $\mathscr{D}'$, of an oriented gt-pair, which is obtained by recursively *merging* adjacent members of $\mathscr{S}$. To be precise, the merging is done by first reversing the orientation of the common marker arc in one of the two members, then identifying the marker arc in the two members (i.e., head to head, tail to tail), and finally deleting this arc. Note that the order in which the members of $\mathscr{D}'$ are merged does not matter. The resulting oriented gt-pair is denoted $m(\mathscr{D}')$.

Now consider Step 3 of the algorithm, and suppose that D is 2-isomorphic to a digraph D' in which T is an arborescence. Suppose that a 3-decomposition $\mathscr{D}$ of (D, T) has been computed. Then, as shown below (Lemma 6), a 3-decomposition of (D', T) can be obtained by replacing the members of $\mathscr{D}$ with appropriate 2-isomorphic copies. Thus, in determining whether such a D' exists, it is sufficient to compute $\mathscr{D}$, replace the members of $\mathscr{D}$ by appropriate 2-isomorphic copies if possible, and then merge the resulting 3-decomposition to obtain (D', T). Computing the 3-decomposition of (D, T) is done by adapting the Hopcroft–Tarjan [12] algorithm for computing the 3-decomposition of the underlying graph of D and is discussed below. Replacing the members of $\mathscr{D}$ by 2-isomorphic copies is done by a one-time pass through the decomposition tree and is discussed in detail below. Finally, the merging of the resulting 3-decomposition is straightforward and is not discussed further.

Hopcroft and Tarjan [12] give a linear-time algorithm for computing the 3-decomposition of a graph. This algorithm is used to compute decomposition of an oriented gt-pair (D, T) as follows. First, compute the 3-decomposition $\mathscr{G}$ of G, the underlying graph of D. Choose $G_1 \in \mathscr{G}$ such that G_1 has exactly one marker edge, say e.

(Such a member always exists.) Either $T \cap E(G_1)$ or $(T \cap E(G_1)) \cup \{e\}$ is a spanning tree of G_1; determining which requires $O(|E(G_1)|)$ time. In either case, define T_1 to be this spanning tree. Now, the edges of G_1 are appropriately oriented to obtain a digraph D_1. For an edge other than e, its orientation is that induced by D. The orientation of e is that which makes (D_1, T_1) an oriented gt-pair. Such an orientation of e exists (and is unique) and it is obtained by examining any fundamental cycle of (G_1, T_1) that contains e. This requires $O(|E(G_1)|)$ time. By choice of G_1, $\mathscr{G} - \{G_1\}$ is a decomposition, and so the above construction can be applied recursively to yield a 3-decomposition of (D, T). Note that the orientation of e in the member of $\mathscr{G}$ other than G_1 is now determined; it is opposite of that in D_1. The total time required is $O(\sum_{i=1}^{t} |E(G_i)|)$, which is $O(|A(D)|)$ by a result of Hopcroft and Tarjan [12].

The above construction also proves the following result.

Lemma 5. *Let (D,T) be an oriented gt-pair, and let G be the underlying graph of D. Let $\{G_1, \ldots, G_t\}$ be a decomposition of G. Then, there exists a decomposition of (D,T) such that G_i is the underlying graph of D_i, for $1 \leqslant i \leqslant t$.*

It is convenient to now assume that the decomposition tree associated with a decomposition is in fact an arborescence. This is done by specifying one member of the decomposition to be the root. For a member other than the root, define its *parent marker* to be the unique marker arc of the member that is in common with its parent.

The following notation is fixed for the remainder of the section. Let $\mathscr{D}$ be the 3-decomposition of (D, T), the input to Step 3, and let (D_0, T_0) be a fixed member of $\mathscr{D}$. Let t be the number of children of (D_0, T_0), and let $\mathscr{D}_i$ be the subdecomposition of $\mathscr{D}$ consisting the ith child together with all of its descendants. Define $(D_i, T_i) := m(\mathscr{D}_i)$. Then, $(D_1, T_1), \ldots, (D_t, T_t)$ are the *complete children* of (D_0, T_0). Define $(R, T_R) := m(\{(D_0, T_0), \ldots, (D_t, T_t)\})$. (Thus, if (D_0, T_0) is the root of $\mathscr{D}$, then $(R, T_R) = (D, T)$.) Let p_i be the parent marker of (D_i, T_i). If (D_0, T_0) is not the root of $\mathscr{D}$, then let p be the parent marker of (D_0, T_0); otherwise choose p to be any nonmarker arc of D_0, and define it to be the parent marker of (D_0, T_0).

Lemma 6 below is an adaptation of a similar result of Bixby and Wagner [2].

Lemma 6. *Let R' be 2-isomorphic to R. Then, there exists a decomposition of (R', T_R) consisting of oriented gt-pairs $(D'_0, T_0), \ldots, (D'_t, T_t)$ such that $D'_0, \ldots, D'_t$ are 2-isomorphic to $D_0, \ldots, D_t$, respectively.*

Proof. Let $H, H', H_0, \ldots, H_t$ be the respective underlying graphs of $R, R', D_0, \ldots, D_t$. Then, H and H' are 2-isomorphic. Now, by Bixby and Wagner [2, Theorem 4.1], there exists a decomposition $\mathscr{H}'$ of H' consisting of graphs $H'_0, \ldots, H'_t$ 2-isomorphic to $H_0, \ldots, H_t$, respectively. By Lemma 5, there exists a decomposition of (R', T) consisting of oriented gt-pairs $(D'_0, T_0), \ldots, (D'_t, T_t)$ such that H'_i is the underlying graph of D'_i, for $0 \leqslant i \leqslant t$. Since H_i and H'_i are 2-isomorphic, (D_i, T_i) and (D'_i, T_i) have the same fundamental cycles, for $0 \leqslant i \leqslant t$. Thus, by Lemma 3 and Theorem 2, D_i and D_i' are 2-isomorphic, for $0 \leqslant i \leqslant t$. □

Recursive application of Lemma 6 implies that (D', T) can be obtained from $\mathscr{D}$ by replacing some of its members by 2-isomorphic copies and then merging. Note that the converse is also true. That is, if $\mathscr{D}'$ is obtained from $\mathscr{D}$ by replacing some of its members by 2-isomorphic copies, then $m(\mathscr{D}')$ and (D, T) are 2-isomorphic.

Determining whether it is possible to replace some of the members of $\mathscr{D}$ by 2-isomorphic copies in such a way that merging the resulting 3-decomposition gives an oriented gt-pair in which T is an arborescence is done by a one-timė "bottom-to-top" pass through $\mathscr{D}$. Since each member of $\mathscr{D}$ is either 3-connected, a bond or a polygon, the set of digraphs 2-isomorphic to a given member is easy to describe.

As motivation for what follows, suppose that the 3-decomposition $\mathscr{D}$ of (D, T) has just two members, say (D_0, T_0) and (D_1, T_1), with (D_0, T_0) as the parent of (D_1, T_1). Now it is easy to see that if T is an arborescence in some digraph D' 2-isomorphic to D, then both T_0 and T_1 are arborescences in their respective members of any 3-decomposition of (D', T). Consider two 2-isomorphic copies of D_1, say D_1' and D_1'', such that T_1 is an arborescence in both D_1' and D_1'' and such that the root of T_1 is not an end of p_1 in D_1', but is in D_1''. A key observation is that if T is an arborescence in the oriented gt-pair obtained by merging (D_1', T_1) with some 2-isomorphic copy of (D_0, T_0), then it is also an arborescence in the oriented gt-pair obtained by merging (D_1'', T_1) with the same 2-isomorphic copy of (D_0, T_0). On the other hand, there exist examples where T is an arborescence if (D_1'', T_1) is merged with some 2-isomorphic copy of (D_0, T_0), but it is not an arborescence if (D_1', T_1) is merged with any 2-isomorphic copy of (D_0, T_0). For example, this happens when the root T_0 is not an end of p_1 in any 2-isomorphic copy of (D_0, T_0). These observations lead to the conclusion that the preferred choice of a 2-isomorphic copy of (D_1, T_1) is one in which T_1 is an arborescence, the root of which is incident to p_1.

The following classification scheme for oriented gt-pairs is introduced for the purpose of carrying out the above preference idea. Let (Q, S) be an oriented gt-pair, and let m be a distinguished arc of Q. Then, the *arrangement* of (Q, S, m), denoted $a(Q, S, m)$, is equal to 1 if T is an arborescence of Q and the root of T is an end of m; is equal to 2 if T is an arborescence of Q and the root of T is not an end of m; and is equal to 3 otherwise. The *type* of (Q, S, m), denoted $t(Q, S, m)$, is $\min\{a(Q', S, m) \mid Q'$ is 2-isomorphic to $Q\}$. The triple (Q, S, m) is *good* if $t(Q, S, m) = a(Q, S, m) < 3$.

For simplification purposes, $a(R, T_R, p)$ is abbreviated to $a(R)$, $t(R, T_R, p)$ to $t(R)$, $a(D_i, T_i, p_i)$ to $a(D_i)$, and $t(D_i, T_i, p_i)$ to $t(D_i)$. This simplification is also used with digraphs 2-isomorphic to R or D_i. The term *good* means $a(R) = t(R) < 3$, etc.

If D_0 is 3-connected or a bond, then the only digraph 2-isomorphic to it is its *converse*, i.e., the digraph obtained by reversing the orientation on each arc. Polygons, on the other hand, have several 2-isomorphic copies. In the case that D_0 is a polygon, the following procedure is used to choose an appropriate 2-isomorphic copy during the pass through $\mathscr{D}$. To *relink* a polygon is to construct a 2-isomorphic copy of it.

Procedure RELINK(D_0).
If $p \notin T_0$ and the type of D_1 (say) is different from 1, then relink D_0 so that the head of p is incident to p_1.
If $p \in T_0$ and $t(D_i) \neq 2$ for $1 \leqslant i \leqslant t$, then relink D_0 so that the head of the unique arc not in T_0 is incident to p.
If $\{p, p_1\} \subseteq T_0$ and the type of D_1 (say) is 2, then relink D_0 so that the head of the unique arc not in T_0 is incident to p_1.

The algorithm for solving the problem of Step 3 can now be stated.

Algorithm 2-ISOMORPHISM.
Input: An oriented gt-pair (D, T).
Output: An oriented gt-pair (D', T) 2-isomorphic to (D, T) in which T is an arborescence or the conclusion that no such oriented gt-pair exists.
Step 1: Compute a 3-decomposition $\mathscr{D}$ of (D, T), and choose as the root of $\mathscr{D}$ a member that has an arc that is not a marker arc; define this arc to be the parent marker of the root. Let h be the height of the arborescence, and for $1 \leqslant i \leqslant h$, define P_i to be the subset of members of $\mathscr{D}$ that are at distance i from the root. Set $j \leftarrow h$.
Comment: The 3-decomposition $\mathscr{D}$ might be modified below; for convenience, the resulting 3-decomposition is still called $\mathscr{D}$.
Step 2: Choose an oriented gt-pair $(D_0, T_0) \in P_j$, and set $P_j \leftarrow P_j - \{(D_0, T_0)\}$. Let $(D_1, T_1), \ldots, (D_t, T_t)$ be the complete children of (D_0, T_0).
If D_0 is 3-connected or a bond, then define D_0' to be the converse of D_0. Set $(R, T_R) := m(\{(D_0, T_0), \ldots, (D_t, T_t)\})$ and $(R', T_R) := m(\{(D_0', T_0), (D_1, T_1), \ldots, (D_t, T_t)\})$. Set $(R'', T_R) := (R, T_R)$ if $a(R) < a(R')$ and $(R'', T_R) := (R', T_R)$ otherwise.
If D_0 is a polygon, then apply RELINK(D_0) and set $(R'', T_R) := m(\{(D_0, T_0), \ldots, (D_t, T_t)\})$.
If $a(R'') = 3$, then stop with the conclusion that (D, T) is not 2-isomorphic to an oriented gt-pair in which T is an arborescence. If $P_j \neq \emptyset$, then go to Step 2.
Step 3: If $j = 0$, then output $(D, T) := m(\mathscr{D})$ and stop; otherwise set $j \leftarrow j - 1$ and go to Step 2.

Lemma 7. *Let D_1' be 2-isomorphic to D_1, and let $(R', T_R) := m(\{(D_0, T_0), (D_1', T_1), (D_2, T_2), \ldots, (D_t, T_t)\})$. If $a(D_1') \leqslant a(D_1)$, then $a(R') \leqslant a(R)$.*

Proof. If $a(D_1) = 3$, then there exists a vertex of D_1 that has indegree at least two in T_1. This implies that there exists a vertex of R that has indegree at least two in T_R. Thus, $a(R) = 3$, and the result follows. Therefore, it can be assumed that T_1 is an arborescence in D_1 and thus D_1'.

Since T_1 is an arborescence of D_1', each vertex of T_1 in R' that is not an end of m_1 has indegree at most one in T_R. The indegree in T_R of every other vertex of R' is less than or equal to that of the corresponding vertex in R. The results follows. □

The following two theorems are the main theorems for justifying the algorithm.

Theorem 8. *Let D_0 be either 3-connected or a bond, and let D_0' be the converse of D_0. Let $(R', T_R) := m\{(D_0', T_0), (D_1, T_1), \ldots, (D_t, T_t)\}$. If D_i is good for $1 \leqslant i \leqslant t$, then either R or R' is good, or $t(R) = 3$.*

Proof. By Lemma 6, there exist digraphs $D_0'', \ldots, D_t''$ that are 2-isomorphic to $D_0, \ldots, D_t$, respectively, such that if (R'', T_R) is defined to be $m(\{(D_0'', T_0), (D_1'', T_1), \ldots, (D_t'', T_t)\})$, then $a(R'') = t(R)$. Since, for $1 \leqslant i \leqslant t$, D_i is good, $a(D_i) \leqslant a(D_i'')$. Since D_0 is 3-connected or a bond, either $D_0'' = D_0$ or $D_0'' = D_0'$. By Lemma 7, either $a(R) \leqslant a(R'')$ or $a(R') \leqslant a(R'')$. □

Theorem 9. *Let D_0 be a polygon, and suppose* RELINK(D_0) *has been applied. If D_i is good for $1 \leqslant i \leqslant t$, then either R is good or $t(R) = 3$.*

Proof. First suppose that $p \notin T_0$. If $t(D_i) = 1$, for $1 \leqslant i \leqslant t$, then $a(R) = 1$, and the result follows. Suppose D_1 (say) is type 2 and $t(D_2) = \cdots = t(D_t) = 1$. By RELINK$(D_0)$, the head of p is incident to p_1. It follows that $a(R) = 2$. Now Lemma 6 implies that in any digraph 2-isomorphic to R, there exists a vertex that is not an end of p and has an indegree of zero in T_R, which implies that $t(R) \geqslant 2$; the result follows. Similarly, if D_1 and D_2 (say) are both of type 2, then $t(R) = 3$.

Now suppose that $p \in T_0$. Let e be the unique arc of the D_0 not in T_0.

If $t(D_i) \neq 2$, for $1 \leqslant i \leqslant t$, then by RELINK$(D_0)$, the head of e is incident to p. Since D_i is good, for $1 \leqslant i \leqslant t$, $a(D_i) = 1$, which implies $a(R) = 1$; the result follows.

If $t(D_1) = 2$ and $p_1 \in T_0$, then, by RELINK(D_0), the head of e is incident to p_1. If $t(D_1) = 2$ and $p_1 \notin T_0$, then $p_1 = e$. In either case, if $t(D_i) = 1$ for $2 \leqslant i \leqslant t$, then $a(R) = 2$. Lemma 6 implies that in any digraph 2-isomorphic to R, there exists a vertex that is not an end of p, and has indegree zero in T_R. This implies that $t(R) \geqslant 2$, and the result follows. If D_2 (say) is type 2, then Lemma 6 implies that in any digraph 2-isomorphic to R, there exists at least two vertices of indegree zero in T_R. This implies that $t(R) = 3$, and the result follows. □

Theorem 10. *Algorithm* 2-ISOMORPHISM *is correct and has time complexity* $O(|A(D)|)$.

Proof. As observed earlier, a 3-decomposition of (D, T) can be computed in $O(|A(D)|)$ time. Moreover, the procedure is easily adapted to compute the sets $P_1, \ldots, P_h$ needed in Step 1 within the same time complexity.

In Step 2, applying RELINK(D_0) to a polygon D_0 requires $O(|A(D_0)|)$ time. Computing the converse of D_0 can also be done in $O(|A(D_0)|)$ time. Determining the arrangement of R (or R' or R'') can be done without actually computing R. In particular, one can replace each complete child (D_i, T_i) of (D_0, T_0) by a "small" oriented gt-pair (F_i, S_i) having the same arrangement and then compute the arrangement of $m\{(D_0, T_0), (F_1, S_1), \ldots, (F_t, S_t)\}$. This can be done in $O(|A(D_0)|)$ time.

Throughout the algorithm, Step 2 might examine every member of $\mathscr{D}$. Thus, Step 2 requires $O(\sum(|A(D_i)|: (D_i, T_i) \in \mathscr{D}))$ time, which is $O(|A(D)|)$ time by a theorem of Hopcroft and Tarjan [12]. Computing $m(\mathscr{D})$ in Step 3 requires the same amount of time. Thus, Algorithm 2-ISOMORPHISM requires $O(|A(D)|)$ time.

If $a(R'') = 3$, then Theorems 8 and 9 imply that $t(R) = 3$. Thus, in any graph 2-isomorphic to R, there exists a vertex that has indegree at least two in T_R. Lemma 6 then implies in any graph 2-isomorphic to D, there exists a vertex of indegree at least two in T. Therefore, the stopping criterion in Step 2 is correct.

If Step 3 is executed, then the algorithm did not stop in Step 2 when D_0 was the root of the 3-decomposition. Thus, in Step 2, $a(R'') < 3$, from which it follows that T is an arborescence of D'. □

Combining the results of the previous two sections yields the following result.

Theorem 11. *Algorithm* ARBORESCENCE REALIZATION *has time complexity* $O(\alpha(n,r)n)$.

5. Related results

A $\{0,1\}$-matrix M is *vertex-arborescence graphic* if there exists an arborescence T such that the vertices of T are indexed on the rows of M and the columns of M are the incidence vectors of the vertex sets of dipaths of T. If such a T exists, then T is a *vertex-arborescence realization* for M. The *vertex-arborescence-realization problem* is to determine whether a given $\{0,1\}$-matrix is vertex-arborescence graphic and, if so, to construct a vertex-arborescence realization. The first polynomial-time algorithm for this problem was developed by Truszczyński [19]. In an unpublished paper, Dietz et al. [8] gave an $O(n)$ algorithm. Theorem 13 below shows that the arborescence-realization problem and the vertex-arborescence-realization problem are equivalent.

Lemma 12. *Let (D,T) be an oriented gt-pair in which D is 2-connected and T is an arborescence. Then, the root of T has degree one in T.*

Proof. Suppose that there are two arcs of T incident to the root. Since D is 2-connected, there exists a fundamental cycle C of (D, T) that contains both arcs. Since (D, T) is an oriented gt-pair, C is a dicycle, implying that the root has indegree greater than zero, a contradiction. □

Theorem 13. *A connected $\{0,1\}$-matrix M is arborescence graphic if and only if it is vertex-arborescence graphic.*

Proof. The proof is by construction. Suppose M is arborescence graphic, and let T be an arborescence realization for M. Construct an arborescence T' from T as follows. Delete the root and its unique (by Lemma 12) incident arc, and for every arc e of T, change the name of the head of e to e. Then, T' is a vertex-arborescence realization for M. Reversing the above construction shows that if M is vertex-arborescence graphic, then it is arborescence graphic. $\square$

By Theorem 13 and its proof, the Dietz–Furst–Hopcroft algorithm yields an $O(n)$ algorithm for the arborescence-realization problem. This bound is theoretically better than the $O(\alpha(n, r)n)$ bound presented here, but from a practical view point, they are the same. The algorithms differ further in that the present algorithm is graph theoretic, requiring only simple data structures (assuming that the Bixby–Wagner algorithm [2] is used for Step 1), whereas the time bound for Dietz–First–Hopcroft algorithm is obtained by employing a sophisticated data structure called a PQR-tree, which is an extension of the PQ-tree developed by Booth and Lueker [5].

One important feature of the present algorithm is that it can be adapted to recognizing a subclass of extended-Horn problems; this is discussed in the next section. It is not clear how to use the Dietz–Furst–Hopcroft algorithm in this way.

The algorithm presented here is also related to some structural results first obtained by Bland and Ko [3] and independently by Swaminathan and Wagner [15, 16]. Bland and Ko characterized when a spanning tree of a connected graph is one that can be obtained by applying depth-first search. Call a gt-pair (G, T) a *dfs-pair* if T is a tree that can be obtained by applying depth-first search to G. Bland and Ko's result is of the form that a gt-pair (G, T) is a dfs-pair if and only if it does not have a "minor" that is in some specified list of gt-pairs. Note that (G, T) is a dfs-pair if and only if G has an orientation D such that (D, T) is an oriented gt-pair and T is an arborescence of D. The Bland–Ko result is actually comprised of two results. The first result characterizes those gt-pairs that underlie oriented gt-pairs, and the second result characterizes the subset of these that are dfs-pairs.

Finally, Ko [13] gave an algorithm that provides an alternative to Steps 2 and 3 of the main algorithm. Ko's algorithm is similar to the approach presented here in that it uses the same decomposition and achieves the same complexity.

6. Simple extended-Horn sets

The well-known satisfiability problem is the quintessential NP-complete problem; see Garey and Johnson [11]. Restricted versions of the satisfiability problem are known to be solvable in polynomial time. Among the more well-known restrictions is the satisfiability problem defined over Horn sets. A propositional clause is *Horn* if it contains at most one positive literal; a set of propositional clauses is *Horn* if each of its clauses is Horn. The satisfiability problem defined over Horn sets can be solved in linear time; see Dowling and Gallier [9]. Recognizing whether an instance is Horn can obviously be done in linear time.

Chandru and Hooker [6] introduced a generalization of Horn sets, called *extended-Horn*, and showed that the satisfiability problem for extended-Horn sets can also be solved in linear time. The definition of extended-Horn sets is rather complicated. No polynomial-time algorithm is known for recognizing whether an instance of the satisfiability problem is extended-Horn. This section presents a recognition algorithm for a subclass of extended-Horn, called *simple extended-Horn*. The recognition algorithm solves a sequence of arborescence-realization problems.

An instance of the satisfiability problem is specified by a collection of $\{0, \pm 1\}$-vectors indexed on a set of variables U. Each vector represents a clause; a $+1$ represents a positive literal and a -1 a negative literal.

Let C be a clause on a variable set U, and let T be an arborescence on arc set U with root s. Then, C is *extended-Horn with respect to* T if the set of arcs having a $+1$ in C is a dipath P of T and the set of arcs having -1 in C is an arc-disjoint union of dipaths $Q_1, \dots, Q_t$ of T such that one of the following holds:

1. $Q_1, \dots, Q_t$ start at the root s.
2. $Q_1, \dots, Q_{t-1}$, (say), start at the root s, and Q_t and P start at a vertex $q \neq s$.

The clause C is *simple extended-Horn with respect to* T if it is extended-Horn with respect to T and condition 1 is satisfied. A set of clauses is *extended-Horn* (respectively, *simple extended-Horn*) if there exists an arborescence T such that each clause in the set is extended-Horn (respectively, simple extended-Horn) with respect to T.

Simple extended-Horn sets include Horn sets since a Horn set is simple extended-Horn with respect to the arborescence in which all the arcs are incident to the root.

Let $\mathscr{C} = \{C_1, \dots, C_k\}$ be a set of clauses on a set of variables U. Let P_i and N_i, for $1 \leqslant i \leqslant k$, denote the subsets of U corresponding to the set of positive and negative literals of C_i, respectively. Let $\mathscr{P} := \bigcup_{i=1}^{k} P_i$ and $\mathscr{N} := \bigcup_{i=1}^{k} N_i$. If $u \in \mathscr{N} - \mathscr{P}$, then u can be deleted from each clause because the set after deletion is simple extended-Horn if and only if the original set is. In particular, if the set after deletion is simple extended-Horn with respect to an arborescence T, then adding an arc to T corresponding to u incident to the root of T results in an arborescence with respect to which the original set is simple extended-Horn. Thus, assume $\mathscr{N} \subseteq \mathscr{P} = U$.

If $\mathscr{C}$ is simple extended-Horn, then there exists an arborescence in which $P_1, \dots, P_k$ are dipaths. In other words, the matrix M, the columns of which are the incidence vectors of $P_1, \dots, P_k$, is arborescence graphic. The arborescence graphicness of M is necessary for $\mathscr{C}$ to be simple extended-Horn; in general, it is not sufficient.

Let $M_1, \dots, M_t$ be the blocks of M, and let $\mathscr{P}_1, \dots, \mathscr{P}_t$ be the partition of $\mathscr{P}$ (= the row set of M) induced by $M_1, \dots, M_t$. Observe that by the definition of blocks, either $P_i \subseteq \mathscr{P}_j$ or $P_i \cap \mathscr{P}_j = \emptyset$, for all $1 \leqslant i \leqslant k$ and $1 \leqslant j \leqslant t$. This partition of $\mathscr{P}$ also induces a partition of each N_i. Namely, for $1 \leqslant i \leqslant k$ and $1 \leqslant j \leqslant t$, define $N_{ij} := N_i \cap \mathscr{P}_j$. As will be seen shortly, the partition $N_{i1}, \dots, N_{it}$ of N_i corresponds to that required in the definition of simple extended-Horn.

For $1 \leqslant j \leqslant t$, define S_j to be the matrix obtained from M_j by adding columns that are the incidence vectors of members of $\{N_{ij} | 1 \leqslant i \leqslant k, N_{ij} \neq \emptyset\}$. Define I_j to be the intersection over all the members of the set $\{N_{ij} | 1 \leqslant i \leqslant k, N_{ij} \neq \emptyset\}$.

The following theorem serves as the basis for the recognition algorithm.

Theorem 14. *The set $\mathscr{C}$ is simple extended-Horn if and only if $S_1, \ldots, S_t$ are arborescence graphic and have respective arborescence realizations $T_1, \ldots, T_t$ in which some member of I_j is incident to the root of T_j, for $1 \leqslant j \leqslant t$.*

Proof. Suppose that $S_1, \ldots, S_t$ are arborescence graphic with respective realizations $T_1, \ldots, T_t$ as described above. For $1 \leqslant j \leqslant t$, let (D_j, T_j) be the corresponding oriented gt-pair. Since M_j is connected, S_j is connected. Thus, D_j is 2-connected. By Lemma 12, the root of T_j has degree one in T_j. Since some member of I_j is incident to the root, N_{ij} is a dipath of T_j that starts at the root for $1 \leqslant i \leqslant k$. Moreover, each P_i that is contained in $\mathscr{P}_j$ is a dipath of T_j. It follows that $\mathscr{C}$ is simple extended-Horn with respect to the arborescence obtained by identifying the roots of $T_1, \ldots, T_t$.

Now suppose that $\mathscr{C}$ is simple extended-Horn with respect to some arborescence, say T. Then, $P_1, \ldots, P_k$ are dipaths of T. Moreover, since $\mathscr{N} \subseteq \mathscr{P}$, every arc of T is in at least one P_i. Thus, T is an arborescence realization of the matrix M. Convert T to an oriented gt-pair, say (D, T), by adding an arc joining the ends of each P_i. Let $D_1, \ldots, D_t$ be the blocks of D, and let $(D_1, T_1), \ldots, (D_t, T_t)$ be the corresponding oriented gt-pairs. Then, T_j is an arborescence realization of block M_j, for $1 \leqslant j \leqslant t$.

Now, consider N_i, for some $i \in \{1, \ldots, k\}$. It is the disjoint union of dipaths of T, each of which starts at the root of T. Observe that $N_{ij} = N_i \cap T_j$, for $1 \leqslant j \leqslant t$. It follows that if N_{ij} is nonempty for some j, then N_{ij} is a dipath in T_j starting at the root of T_j. In other words, T_j is the required realization of the matrix S_j. □

Theorem 14 can be used to determine whether a given set $\mathscr{C} = \{C_1, \ldots, C_k\}$ of clauses is simple extended-Horn as follows. The first step is to construct the matrices $S_1, \ldots, S_t$. This is done as follows. First scan each clause to determine the sets $P_1, \ldots, P_k$ and $N_1, \ldots, N_k$, and then construct the matrix M defined above. Given that the number of variables in U is m, this can be done in $O(mk)$ time. Next, find the blocks $M_1, \ldots, M_t$ of M; this can also be done in $O(mk)$ time, as observed in Section 2. Using the partition of the row set of M given by $M_1, \ldots, M_t$, the partition of each N_i into the set $N_{i1}, \ldots, N_{it}$ can be found in $O(m)$ time. Thus, the matrices $S_1, \ldots, S_t$ can be found in $O(mk)$ time. Observe that the total number of nonzeros in the matrices $S_1, \ldots, S_t$ is bounded by mk. Thus, by Theorem 16, determining whether $S_1, \ldots, S_t$ are arborescence graphic can be done in time $O(\alpha(mk, m)mk)$.

Suppose that $S_1, \ldots, S_t$ are found to be arborescence graphic. Let $T_1, \ldots, T_t$ be the respective realizations. The next step is to determine whether each S_j has an arborescence realization of the desired type. Consider S_1. First, construct the set I_1 by scanning each row of S_1. Second, construct the oriented gt-pair, say (D_1, T_1), corresponding to the realization T_1. Choose $e \in I_1$, and consider applying Algorithm 2-ISOMORPHISM to (D_1, T_1) with the following refinements of Step 1. After computing a 3-decomposition $\mathscr{D}$ of (D_1, T_1), choose as the root of $\mathscr{D}$, the member that contains e, and choose as the parent marker of the root, the arc e. Since S_1 is arborescence graphic, the output of Algorithm 2-ISOMORPHISM is an oriented gt-pair (D_1', T_1) in which T_1 is an arborescence. Moreover, by Step 2 and Theorems

8 and 9, the triple (D_1', T_1, e) is good. The implication of (D_1', T_1, e) being good is that if there exists a digraph 2-isomorphic to D_1 in which T_1 is an arborescence and e is incident to the root of T_1, then D_1' is such a digraph. It follows that by running Algorithm 2-ISOMORPHISM for each arc e in I_1, it can be determined whether there exists an arborescence realization of S_1 in which some arc of I_1 is incident to the root.

By repeating the analysis for $S_2, \ldots, S_t$, it can be determined whether $\mathscr{C}$ is simple extended-Horn. In addition to the $O(\alpha(mk, m)mk)$ time for determining whether $S_1, \ldots, S_t$ are arborescence graphic, the complexity is dominated by executing Algorithm 2-ISOMORPHISM once for each edge in $I_1 \cup \cdots \cup I_t$. Note the size of this union is bounded by m. Thus, the following result has been proved.

Theorem 15. *Determining whether a set of k clauses on m variables is simple extended-Horn can be done in* $O(m^2k)$ *time.* □

References

[1] M.O. Ball, J.S. Provan and D.R. Shier, Reliability covering problems, Networks 21 (1991) 345–357.

[2] R.E. Bixby and D. K. Wagner, An almost-linear time algorithm for graph realization, Math. Oper. Res. 13 (1988) 99–123.

[3] R.G. Bland and C.W. Ko, Characterizations of Camion trees and depth-first search trees by excluded configurations, Technical Report 909, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY, (1990).

[4] J.A. Bondy and U.S.R. Murty, Graph Theory with Applications (North-Holland, New York, 1976).

[5] K.S. Booth and G.S. Lueker, Testing for the consecutive ones property, interval graphs and graph planarity using PQ-tree algorithms, J. Comput. System Sci. 13 (1976) 335–379.

[6] V. Chandru and J. Hooker, Extended-Horn sets in propositional logic, J. ACM 38 (1991) 205–221.

[7] W. H. Cunningham and J. Edmonds, A combinatorial decomposition theory, Canad. J. Math. 22 (1980) 734–765.

[8] P. Dietz, M. Furst and J.E. Hopcroft, A linear-time algorithm for the generalized consecutive retrieval problem, Technical Report TR-79-386, Department of Computer Science, Cornell University, Ithaca, NY (1979).

[9] W.F. Dowling and J.H. Gallier, Linear-time algorithms for testing the satisfiability of Horn formulae, J. Logic Programming 3 (1984) 267–284.

[10] S. Fujishige, An efficient PQ-graph algorithm for solving the graph-realization problem, J. Comput. System Sci. 21 (1980) 63–86.

[11] M.R. Garey and D.S. Johnson, Computers and Intractability: A Guide to Theory of NP-Completeness (Freeman, New York, 1979).

[12] J. Hopcroft and R.E. Tarjan, Dividing a graph into triconnected components, SIAM J. Comput. 2 (1973) 135–158.

[13] C.W. Ko, An algorithm to find a 2-isomorphic depth-first search image of a tree, Technical Report 927, School of Operations Research and Industrial Engineering, Cornell University, Ithaca, NY (1990).

[14] W. Lipski, Information storage and retrieval-mathematical foundations II (combinatorial problems), Theoret. Comput. Sci. 3 (1976) 183–211.

[15] R.P. Swaminathan and D.K. Wagner, A forbidden-minor characterization of orientable graph-tree pairs, Technical Report CC-89-14, Institute for Interdisciplinary Studies, Purdue University, West Lafayette, IN (1989).

[16] R.P. Swaminathan and D.K. Wagner, A forbidden-minor characterization of depth-first-search trees, Technical Report CC-91-1, Institute for Interdisciplinary Studies, Purdue University, West Lafayette, IN (1991).

[17] R.E. Tarjan, Data Structures and Network Algorithms (SIAM, Philadelphia, PA, 1983).
[18] C. Thomassen, Whitney's 2-switching theorem, cycle spaces, and arc mappings of directed graphs, J. Combin. Theory Ser. B 46 (1989) 259–291.
[19] M. Truszczyński, An algorithm of finding an acyclic f-graph for a family of sets, Fund. Inform. 3 (1980) 379–396.
[20] W.T. Tutte, Connectivity in Graphs (University of Toronto Press, Toronto, Ont., 1968).
[21] H. Whitney, 2-isomorphic graphs, Amer. J. Math. 55 (1933) 245–254.