JOURNAL OF ALGORITHMS 16, 283–294 (1994)

An $O(n^2)$ Divide-and-Conquer Algorithm for the Prime Tree Decomposition of Two-Structures and Modular Decomposition of Graphs

ANDRZEJ EHRENFEUCHT, HAROLD N. GABOW,
ROSS M. MCCONNELL, AND STEPHEN J. SULLIVAN

Department of Computer Science, University of Colorado at Boulder, Boulder, Colorado 80309

Received May 1992; accepted May 6, 1993

This paper presents a simple divide-and-conquer algorithm for computing the prime tree decomposition of a two-structure. The algorithm runs in $O(n^2)$ time, when n is the number of nodes of the two-structure. A directed or undirected graph is a special case of a two-structure, and the restriction of the decomposition to graphs is known as the *modular decomposition* or *substitution decomposition*. The decomposition has applications in solving certain scheduling problems and a number of problems on graphs and partial orders. Two algorithms with a comparable time bound have previously been published for undirected graphs, but they make use of elaborate data structures that limit their practical use, and they have no easy generalization to directed graphs or two-structures.

1. INTRODUCTION

The adjacency array representation of a graph is a coloring of the set $\{(x, y)$: x and y are nodes of G and $x \neq y\}$ with two colors, 0 and 1. A two-structure [1, 2] is the generalization that allows an arbitrary coloring of this set. A two-structure may thus be viewed as an edge-colored complete directed graph. Graphs are a special case of two-structures, so all theorems and algorithms presented in this paper for two-structures are immediately applicable to graphs.

If g is a two-structure, $\mathrm{dom}(g)$ denotes its nodes. The *substructure induced in g by* $X \subseteq \mathrm{dom}(g)$, denoted $g|X$, is the subgraph of g induced by X, where the edges in $g|X$ have the same color as they do in g. A node x *distinguishes* nodes y and z if (x, y) and (x, z) are different colors, or if (y, x) and (z, x) are different colors. A *clan* is a set $X \subseteq \mathrm{dom}(g)$ such

0196-6774/94 $6.00

that no $x \in \text{dom}(g) - X$ distinguishes any two members of X. It is easily seen that if two clans are disjoint, all edges from one of the clans to the other are the same color. If $\mathbf{R}$ is a partition of the nodes of g, a *system of representatives* from $\mathbf{R}$ is a set consisting of one node from each member of $\mathbf{R}$. If each member of $\mathbf{R}$ is a clan of g, then every system of representatives induces the same substructure. This substructure is denoted $g/\mathbf{R}$, and completely specifies the colors of all edges that are not internal to a member of $\mathbf{R}$. The operation may be performed recursively on each member of $\mathbf{R}$ by partitioning it into still smaller clans. The result is a compact, hierarchical decomposition of the entire two-structure.

The *prime tree family* of a two-structure is such a hierarchical decomposition, and it is unique. It is given by the family of *prime clans*: $\{X : X$ is a nonempty clan of g, and for any clan Y, either $Y \subseteq X$, $X \subseteq Y$, or $X \cap Y = \emptyset\}$. If $|X| > 1$, the maximal-cardinality members of the prime tree family that are proper subsets of $|X|$, denoted $\text{children}_g(X)$, are a partition of X. Thus $(g|X)/\text{children}_g(X)$, given for all X in the prime tree family, completely represents the original two-structure. As is described below, this family represents in a simple way all clans in the two-structure, not just those that are members of the family. The number of clans may be exponential in n, but this representation takes space that is linear in n.

The prime tree family for the case of graphs was first described by Gallai [3] and has since been rediscovered independently by different researchers. Kelly [4] gives a survey of the history of the idea. The prime tree family in the case of graphs is also known as the *substitution decomposition* [5], the *modular decomposition* [6], and the *X-join decomposition* [7], while the clans in the case of graphs are also known as *closed sets* [3], *modules* [6], *autonomous sets* [5], *partitive sets* [8], *clumps* [9], and *stable sets* [10]. The series-parallel decomposition of general series-parallel partial orders [11] and the cotree decomposition of cographs [12] are special cases of this decomposition. The decomposition is also closely related to the *split decomposition* of Cunningham and Edmonds [13].

There is a large number of combinatorial problems on graphs and partial orders whose solution may be facilitated by the prime tree family of the graph or partial order. Examples include finding maximum-weight cliques and maximum-weight matchings, minimum node colorings [14], finding the dimension of a partial order [15], constructing perfect graphs [14], finding whether a graph is a comparability graph [16], and solving certain scheduling problems [17]. Mohring [5] gives a review.

James, Stanton, and Cowan [18] give an early $O(n^4)$ algorithm for computing the prime tree family for the case of undirected graphs. Various $O(n^3)$ algorithms to construct the prime tree family for the cases of partial orders and undirected graphs have since been given [7, 19, 20].

Muller and Spinrad [6] give an $O(n^2)$ algorithm for undirected graphs, and Spinrad [21] gives an $O(n + m\alpha(m, n))$ algorithm for undirected graphs. Both of these algorithms use elaborate data structures and apply only to undirected graphs. Maurer [22] gives an $O(n^4)$ algorithm for the more general case of directed graphs. Ehrenfeucht, Harju, and Rozenberg [23] give an algorithm for arbitrary two-structures that runs in $O(n^3)$ time. The algorithm presented here is quite simple, and gives the result for arbitrary two-structures in $O(n^2)$ time.

2. Preliminaries

In this section, we establish notation and give results from previous papers that will be referred to in this paper. Let G be a directed graph. The *component graph* for G [24] has one node for each strongly connected component. If X and Y are two strongly connected components of G, then (X, Y) is an edge in the component graph if there exist $x \in X$ and $y \in Y$ such that (x, y) is an edge of G.

Let X and Y be finite sets. X and Y are *overlapping* if and only if $X - Y$, $X \cap Y$, and $Y - X$ are all nonempty. A *two-edge* over X is an ordered pair (x, y) such that $x \neq y$ and $x, y \in X$. In this paper, the term *edge* refers to a two-edge. The set $E_2(X)$ is the set of all possible edges over X, and a two-structure on domain X is a coloring of $E_2(X)$. An edge (x, y) is *symmetric* if (y, x) is the same color. A two-structure is symmetric if all of its edges are symmetric. X is a *singleton* set if $|X| = 1$. The nodes of a two-structure g are known as its *domain*, denoted $\mathrm{dom}(g)$, and g is a coloring of $E_2(\mathrm{dom}(g))$. Obviously, $\mathrm{dom}(g)$, $\varnothing$, and the singleton subsets of $\mathrm{dom}(g)$ are clans of g; these are the *trivial* clans of g. In this paper, clans will be assumed to be nonempty, except when otherwise stated.

Let g be a two-structure, and let $\mathbf{R}$ be a partition of $\mathrm{dom}(g)$ such that every member of $\mathbf{R}$ is a clan of g. Let g' be the two-structure $g/\mathbf{R}$. If $X \subseteq \mathrm{dom}(g')$, its *inverse image* in g is the union of the members of $\mathbf{R}$ that correspond to members of X. X is the *image* in g' of its inverse image.

Theorem 2.1 [1]. *Let g be a two-structure, and let X and Y be overlapping clans of g. Then $X \cap Y$, $X \cup Y$, and $X - Y$ are clans of g.*

The following is a restricted variant of Theorem 4.17 of [1]. See also [25] for an earlier proof on graphs.

Theorem 2.2. *Let g be a two-structure and let $\mathbf{R}$ be a partition of* $\mathrm{dom}(g)$ *into clans.*

1. *If X is a clan in g that has an image Y in $g/\mathbf{R}$, then Y is a clan in $g/\mathbf{R}$. Moreover, if X is a prime clan in g, then Y is a prime clan in $g/\mathbf{R}$.*

2. *If Y is a clan in $g/\mathbf{R}$, then its inverse image, X, is a clan in g. Moreover, if the members of $\mathbf{R}$ are prime clans in g and Y is a prime clan in $g/\mathbf{R}$, X is a prime clan in g.*

The following is an immediate consequence of Lemma 4.10 in [1]. See also [25] for an earlier proof on graphs.

LEMMA 2.3. *Let g be a two-structure, and let X be a clan of g. Then the clans of $g|X$ are those clans of g that are subsets of X. Moreover, the prime clans of $g|X$ that are proper subsets of X are the prime clans of g that are proper subsets of X.*

DEFINITION 2.4 [1]. Let g be a two-structure.

1. g is *primitive* if and only if it contains at least three nodes and all clans in g are trivial.

2. g is *complete* if and only if all of its edges are the same color.

3. g is *linear* if and only if there exists a linear order $(x_1, \ldots, x_{|\mathrm{dom}(g)|})$ of the elements of $\mathrm{dom}(g)$ such that the edges $\{(x_i, x_j): i < j\}$ are the same color, the edges $\{(x_j, x_i): i < j\}$ are the same color, and the colors of these two sets are different.

It is clear that if g is complete, its clans are all subsets of $\mathrm{dom}(g)$, and if it is linear, then its clans are the consecutive sets (sets of the form $\{x_i, x_{i+1}, \ldots, x_j\}$) in the linear order on its nodes.

THEOREM 2.5 [2, 25].

(1) *If X is a prime clan of g and $|X| > 1$, $\mathrm{children}_g(X)$ is a partition of X;*

(2) *Each clan of a two-structure g is a union of siblings in the prime tree family of g;*

(3) *If X is a prime clan of g such that $|X| > 1$, X is of one of the following types:*

primitive, *no union of more than one and less than all of its children is a clan of g;*
complete, *every union of a subfamily of its children is a clan of g;*
linear, *there exists a linear ordering on its children such that the union of a subfamily is a clan of g if and only if the subfamily is consecutive in that ordering.*

A direct proof of Theorem 2.5 appears in [2], but the theorem is a special case of earlier results, given in Section III.4 of [25], which apply to set families satisfying certain algebraic properties that follow from Theorem 2.1 and the fact that $\mathrm{dom}(g)$ and its singleton subsets are clans. Such

families arise in a number of other contexts. See also [26] for an earlier study of such families, which are called *point-partitive hypergraphs*.

COROLLARY [2]. *If X is a member of the prime tree family, then $(g|X)/\text{children}_g(X)$ is either primitive, linear, or complete.*

By Theorem 2.5, we may represent all clans of g as follows. Create one node to represent each prime clan. If x represents a prime clan X, establish an edge from x to y whenever y represents a member of $\text{children}_g(X)$. We will call this data structure $\text{ptf}(g)$. In $\text{ptf}(g)$, the leaf descendants of a node x give the prime clan X it represents. In addition, if $|X| > 1$, one may label x as primitive, linear, or complete, and supply the appropriate ordering of its children if it is linear. If x is complete, its *color* is the color of the edges connecting its children. If x is linear, its *colors* are the two colors of edges connecting its children.

We seek to compute $\text{ptf}(g)$. For ease of notation, when x is a node of $\text{ptf}(g)$ that represents a prime clan X, we will view x and X as synonymous.

3. THE ALGORITHM

In this section, we give an algorithm that computes $\text{ptf}(g)$ for symmetric two-structures. Since undirected graphs are symmetric two-structures, they are covered. Later, we will give the minor modification needed when the two-structure may have asymmetric edges. We will assume that the two-structure is given as an adjacency array, and that if the two-structure has k edge colors, they are given in the adjacency array as integers from zero to $k - 1$.

DEFINITION 3.1. Let v be a node of $\text{dom}(g)$; $G(g, v)$ denotes a graph whose nodes are given by $\text{dom}(g) - \{v\}$. There is an edge in $G(g, v)$ from node x to node y if x distinguishes y and v in g. $\mathbf{M}(g, v)$ denotes the family of maximal clans of g that do not contain v. That is, $X \in \mathbf{M}(g, v)$ iff X is a clan, and for every clan Y such that $X \subset Y$, Y contains v.

$G(g, v)$ is trivially computed in $O(n^2)$ time. $\mathbf{M}(g, v)$ is a partition of $\text{dom}(g) - \{v\}$, and it may be computed with Algorithm 3.1, variants of which have appeared repeatedly in related contexts, for example, [16]. The prime tree family of an arbitrary two-structure is then computed with Algorithm 3.2.

ALGORITHM 3.1. *Compute* $\mathbf{M}(g, v)$.

Maintain a family $\mathbf{L}$ of partition classes, and for each partition class, S, maintain a set $Z(S)$ of "unprocessed outsiders." Initially, there is one

partition class $S = \text{dom}(g) - \{v\}$ in **L**, with $Z(S) = \{v\}$

While **L** contains a class S such that $Z(S)$ is nonempty
    Remove S from **L**
    Let w be an arbitrary member of $Z(S)$.
    Partition S into the maximal subsets that are not distinguished by w
    For each resulting subset W
        Make W a member of **L**
        Let $Z(W) = (S - W) \cup Z(S) - \{w\}$

ALGORITHM 3.2. *Compute the prime tree family for a symmetric two-structure, g.*

ptf(g)
    Select a node v of g and compute $\mathbf{M}(g, v)$ using Algorithm 3.1
    Let $g' = g/(\mathbf{M}(g, v) \cup \{\{v\}\})$,
    Let $\{v'\}$ be the image of $\{v\}$ in g'
    Let $G' = G(g', v')$
    Let G'' be the component graph of G'
    Create a tree node t
    $u := t$
    While G'' is not empty
        Create a tree node w and make it a child of u
        Remove a sink from G''; Let **F** be the corresponding members of $\mathbf{M}(g, v)$
        If $|\mathbf{F}| > 1$, u is primitive, else u is complete
        For each member X of **F**
            compute ptf($g|X$) recursively
            If u and the root of ptf($g|X$) are both complete and the same color then
                make the children of ptf($g|X$) be children of u;
            else make ptf($g|X$) be a child of u
        $u := w$
    return (t)

4. CORRECTNESS AND TIME BOUND FOR ALGORITHM 3.1

LEMMA 4.1. *Let g be a two-structure, and let $v \in \text{dom}(g)$. Then $\mathbf{M}(g, v) \cup \{\{v\}\}$ is a partition of* dom(g).

Proof. If a node $w \neq v$ is not in any member of $\mathbf{M}(g, v)$, then there is no clan containing w but excluding v, contradicting the fact that $\{w\}$ is such a clan. Suppose w is in two members, X and Y, of $\mathbf{M}(g, v)$. The union of X and Y is a clan that does not contain v, since one contains the other or else they overlap, in which case their union is a clan by Theorem

2.1. They cannot both be maximal clans not containing v, contradicting membership of both of them in $\mathbf{M}(g, v)$. Thus, each node of $\mathrm{dom}(g) - \{v\}$ is a member of exactly one set in $\mathbf{M}(g, v)$. $\mathbf{M}(g, v)$ is a partition of $\mathrm{dom}(g) - \{v\}$, and the lemma follows. □

LEMMA 4.2. *Algorithm* 3.1 *computes* $\mathbf{M}(g, v)$.

Proof. The algorithm clearly maintains the following invariant: For each partition class S, a member x of $\mathrm{dom}(g) - S$ may distinguish members of S only if it is a member of the outside list for S. The algorithm terminates when every outsider list is empty, hence, when each partition class is a clan. Conversely, if a clan of g does not contain v, it is a subset of the initial member of $\mathbf{L}$, and its members cannot be split into different partition classes by any outsider. Thus, its members are all in the same final partition class. Since each final partition class is a clan that does not contain v, and each maximal clan that does not contain v is a final partition class, the final partition classes must be the maximal clans that do not contain v, by Lemma 4.1. □

DEFINITION 4.3. Let g be a two-structure, and let $\mathbf{R}$ be a partition of $\mathrm{dom}(g)$. The edges that are *exposed* by $\mathbf{R}$ are the set $\{(u, v)\colon u, v \in \mathrm{dom}(g)$ and u and v are not in the same partition class of $\mathbf{R}\}$.

LEMMA 4.4. *Let* g *be a two-structure, let* $v \in \mathrm{dom}(g)$, *and let* k *be the number of edges exposed by* $M(g, v) \cup \{\{v\}\}$. *The number of operations required by Algorithm* 3.1 *is* $O(k)$.

Proof. Maintain all sets $\mathbf{L}$ and their outsider sets as linked lists. A set S may be partitioned with an outsider w in $O(|S|)$ time as follows. Use a two-phase bucket sort. In the first phase, bucket sort the members of S according to the color of the edges from w to those members. In the second phase, bucket sort each of the resulting nonempty buckets according to the colors of the edges from those members to w. Charge the cost of the partition to the edges connecting w and the members of S. Charge the cost of having originally inserted w on the outsider set for S to one of these edges. Since w is not on any of the new outsider sets, these edges do not receive any further charges in later iterations, so they are each charged constant cost over all executions. Only exposed edges receive charges. Thus, the costs of performing all partitions and maintaining all lists is $O(k)$. □

5. CORRECTNESS AND TIME BOUND FOR ALGORITHM 3.2

LEMMA 5.1. *Let* g *be a two-structure, and let* $v \in \mathrm{dom}(g)$, *where* $|\mathrm{dom}(g)| > 1$. *Let* $U \neq \{v\}$ *be a proper ancestor of* $\{v\}$ *in* $\mathrm{ptf}(g)$, *and let* W

be U's child that contains v:

1. *If* $(g|U)/\text{children}_g(U)$ *is primitive, each of U's children, other than W, is a member of* $\mathbf{M}(g, v)$.

2. *If* $(g|U)/\text{children}_g(U)$ *is complete, the union of all of U's children except W is a member of* $\mathbf{M}(g, v)$.

3. *If* $(g|U)/\text{children}_g(U)$ *is linear, the union of children of U that are before W in the linear order is a member of* $\mathbf{M}(g, v)$ *or empty. The same is true of any children after W.*

There are no members of $\mathbf{M}(g, v)$ *other than those given by the above rule when it is applied to all ancestors of v.*

Proof. Follows from Theorem 2.5. □

THEOREM 5.2. *Let* $g' = g/(\mathbf{M}(g, v) \cup \{\{v\}\})$. *If g is symmetric, then the clans of g' that contain the image of* $\{v\}$ *are prime in g', and their inverse images in g give the ancestors of* $\{v\}$ *in the prime tree family of g.*

Proof. Suppose U is a proper ancestor of $\{v\}$ in the prime tree family of g and W is its child that contains v. By Lemma 5.1, U and W are the inverse images of sets of nodes of g'. By Theorem 2.2, their images in g' are prime in g'. If g is symmetric, U is complete or primitive, so by Theorem 2.2 and Lemma 5.1, the image of U is the smallest clan of g' that contains the image of W. Applying this argument to all ancestors U of $\{v\}$ shows that all clans of g' that contain the image of $\{v\}$ are the images of the ancestors of $\{v\}$ in the prime tree family of g. □

LEMMA 5.3. *Let g be a two-structure, and let X be the set of nodes reachable from node x in* $G(g, v)$. *The set* $\text{dom}(g) - X$ *is the largest clan of g that contains v and excludes x.*

Proof. If there is an edge from x to y in $G(g, v)$ then (x, y) and (x, v) are different colors or (y, x) and (v, x) are different colors, which means that any clan containing v and excluding x must also exclude y. If y is excluded from the clan, then the same argument shows that any node reachable from y on a single edge must also be excluded from the clan. Transitively, every node reachable from x on any path must be excluded from the clan. To see that $\text{dom}(g) - X$ is, in fact, a clan, we observe that the nodes in $\text{dom}(g) - X$ are not reachable in $G(g, v)$ from any node in X, which means that for any node z in X and any node u in $\text{dom}(g) - X$, (z, u) and (z, v) are the same color and (u, z) and (v, z) are the same color. Transitively, for any two nodes u and w in $\text{dom}(g) - X$, (z, u) and (z, w) are the same color and (u, z) and (w, z) are the same color. Thus, $\text{dom}(g) - X$ is a clan. □

COROLLARY. *Let X be a set corresponding to a sink in the component graph of $G(g, v)$. Then* $\mathrm{dom}(g) - X$ *is a maximal-cardinality clan that contains v and that is not equal to* $\mathrm{dom}(g)$.

PROPOSITION 5.4. *Let g be a two-structure, let $v \in \mathrm{dom}(g)$, and let $X \subseteq \mathrm{dom}(g) - \{v\}$. Let $g' = g \mid (\mathrm{dom}(g) - X)$. Then $G(g', v)$ is the subgraph induced in $G(g, v)$ by* $\mathrm{dom}(g) - (X \cup \{v\})$.

THEOREM 5.5 [2]. *Let g be a two-structure, and let W be a child of U in* $\mathrm{ptf}(g)$. *If U and W are both complete, the color of the edges connecting children of W is different from the color of the edges connecting children of U. If U and W are both linear, the pair of colors of edges connecting children of W is different from the pair of colors connecting children of U.*

The correctness of Algorithm 3.2 now follows. Algorithm 3.2 clearly returns the correct result whenever $|\mathrm{dom}(g)| = 1$. Let the inductive hypothesis be that it returns the correct result whenever $|\mathrm{dom}(g)| < k$. Suppose that $|\mathrm{dom}(g)| = k$. Let W be the child of $\mathrm{dom}(g)$ in $\mathrm{ptf}(g)$ that contains v, and let $g' = g/(\mathbf{M}(g, v) \cup \{\{v\}\})$. By Theorem 5.2, there is a unique maximal clan of g' that is a proper subset of $\mathrm{dom}(g')$ and that contains the image of $\{v\}$. Thus, by the corollary to Lemma 5.3, there is a unique sink in the component graph for $G(g', v')$. By Theorem 5.2, $W = \mathrm{dom}(g') - \bigcup \mathbf{F}$. The characterization of U as primitive or complete is correct, by Lemma 5.1. By the inductive hypothesis, the recursive call produces the correct tree for $g|X$, so by Lemma 2.3 and Theorem 5.5, the main routine correctly attaches all siblings of W and their descendants as children of $\mathrm{dom}(g)$. By Proposition 5.4 and Lemma 2.3, subsequent iterations of the main loop of Algorithm 3.1 are computationally equivalent to a recursive call to Algorithm 3.1 on $g|W$. By the inductive hypothesis, they produce the subtree of $\mathrm{ptf}(g)$ rooted at W. Thus, Algorithm 3.1 produces the correct result when $|\mathrm{dom}(g)| \leq k$. Inductively, it produces the correct result for all symmetric two-structures. □

For the time bound, let k be the number of edges exposed by $\mathbf{M}(g, v) \cup \{\{v\}\}$. If the two-structure has only one node, we may charge the cost of returning the trivial decomposition to the node. Otherwise, by Lemma 4.4, computing $\mathbf{M}(g, v)$ requires $O(k)$ time, so we may charge this cost to the exposed edges. Let $g' = g/(\mathbf{M}(g, v) \cup \{\{v\}\})$. The number of nodes of g' is $O(k^{1/2})$. Computing $G(g', v')$ and the component graph of $G(g', v')$ takes $O(k)$ time, as does the cost of identifying and removing sinks from the component graph. These costs may be charged to the exposed edges at constant cost per edge. In recursive calls, we use the same charging scheme on edges exposed in those calls. Each recursive call generated from the main procedure occurs on the substructure induced by a single member of $\mathbf{M}(g, v)$, and thus, the edges charged in one recursive call are

disjoint from those charged in either the main procedure or any other recursive call. It follows that all costs are charged to the edges at constant time per edge, giving the $O(n^2)$ time bound on the algorithm.

6. The Generalization of Algorithm 3.2 for Arbitrary Two-Structures

Algorithm 6.1 gives the generalization of Algorithm 3.2 for arbitrary two-structures. For the correctness, note that when asymmetric edges are allowed, case 3 of Lemma 5.1 can no longer be excluded. Because of this, Theorem 5.2 is no longer true. Instead, we make use of the following.

LEMMA 6.1. *Let g be a two-structure, let v be a node of g, and let W be the child of* $\mathrm{dom}(g)$ *in* $\mathrm{ptf}(g)$ *that contains v. Let $g' = g/(\mathbf{M}(g, v) \cup \{\{v\}\})$, let v' be the image of v in g', and let W' be the image of W. Let G'' be the component graph of $G(g', v')$, and let X' be the nodes of g' that correspond to the sinks of G''. Then $W' = \mathrm{dom}(g') - X'$.*

Proof. By Lemma 5.1, W has an image in g', and by Theorem 2.2, W' is prime in g'. By Theorem 2.2, if the root of $\mathrm{ptf}(g)$ is primitive, W' is the unique maximal clan of g' that contains v' and that is not equal to $\mathrm{dom}(g')$. If the root of $\mathrm{ptf}(g)$ is complete, then by Theorem 2.2 and Lemma 5.1, W' has only one sibling in $\mathrm{ptf}(g')$, so it is again the unique maximal clan of g' that contains v' and that is not equal to $\mathrm{dom}(g')$. In either case, Lemma 6.1 is true by the corollary to Lemma 5.3. If the root of $\mathrm{ptf}(g)$ is linear, then by Theorem 2.2 and Lemma 5.1, W' has one or two siblings in $\mathrm{ptf}(g')$. If it has one sibling, it is the unique maximal clan of g' that contains v' and that is not equal to $\mathrm{dom}(g')$, so again, the lemma is true. Otherwise, the union of W' and either of these siblings is thus a maximal clan of g' that contains v and is not equal to $\mathrm{dom}(g')$. Thus, each of these siblings is a sink of G'', by the corollary to Lemma 5.3. These are the only sinks of G'', since W' is prime in g'. Thus, the lemma is true in this case also. □

The proof of correctness of Algorithm 6.1 is similar to the one for Algorithm 3.2: Algorithm 6.1 clearly returns the correct result whenever $|\mathrm{dom}(g)| = 1$. Let the inductive hypothesis be that it returns the correct result whenever $|\mathrm{dom}(g)| < k$. Suppose $|\mathrm{dom}(g)| = k$. Let W be the child of $\mathrm{dom}(g)$ in $\mathrm{ptf}(g)$ that contains v, and let $g' = g/(\mathbf{M}(g, v) \cup \{\{v\}\})$. Let X' be the nodes of g' that correspond to the sinks of G''. By Lemma 6.1, W is the inverse image of $\mathrm{dom}(g') - X'$. The characterization of u as primitive is correct, by Lemma 5.1. If there are two sinks in G'', u is linear.

Otherwise, it is linear or complete. Which case holds may be found by examining whether an edge such as (v, x) that connects its children is symmetric. By the inductive hypothesis, the recursive call produces the correct tree for $g|X$, for each $X \in \mathbf{M}(g, v)$, so by Lemma 2.3 and Theorem 5.5, the main routine correctly attaches all siblings of W and their descendants as children of $\mathrm{dom}(g)$. By Proposition 5.4 and Lemma 2.3, subsequent iterations of the main loop of Algorithm 6.1 are computationally equivalent to a recursive call to Algorithm 6.1 on $g|W$. By the inductive hypothesis, they produce the subtree of $\mathrm{ptf}(g)$ rooted at W. Thus, Algorithm 6.1 produces the correct result when $|\mathrm{dom}(g)| \leq k$. Inductively, it produces the correct result for all two-structures. □

The proof of the $O(n^2)$ time bound for Algorithm 6.1 is identical to the one for Algorithm 3.2.

ALGORITHM 6.1. *Compute the prime tree family for an arbitrary two-structure.*

```
ptf(g)
    Select an arbitrary node v of g and compute M(g, v)
    Let g' = g/(M(g, v) ∪ {{v}}),
    Let {v'} be the image of {v} in g'
    Let G' = G(g', v')
    Let G'' be the component graph of G'
    Create a tree node t
    Let u := t
    While G'' isn't empty
        Create a tree node w and make it a child of u
        Remove all sinks of G''; Let F be the corresponding members of
        M(g, v)
        If only one sink was removed from G'' and |F| > 1, u is primitive;
        Else select an arbitrary node x from a member of F
            If (v, x) and (x, v) are the same color, u is complete
            Else u is linear
        For each member X of F
            compute ptf(g|X) recursively
            If u and the root of ptf(g|X) are both complete and the
            same color or both linear and the same colors
                then make the children of ptf(g|X) be children of u;
            else make ptf(g|X) be a child of u
        u := w
```

REFERENCES

1. A. EHRENFEUCHT AND G. ROZENBERG, Theory of 2-structures. Part 1. Clans, basic subclasses, and morphisms, *Theoret. Comput. Sci.* **70** (1990), 277–303.
2. A. EHRENFEUCHT AND G. ROZENBERG, Theory of 2-structures. Part 2. Representations through labeled tree families, *Theoret. Comput. Sci.* **70** (1990), 305–342.
3. T. GALLAI, Transitiv orientierbare graphen, *Acta Math. Acad. Sci. Hungar.* **18** (1967), 25–66.
4. D. KELLY, Comparability graphs, *in* "Graphs and Order" (I. Rival, Ed.), pp. 3–40, Reidel, Boston, 1985.
5. R. H. MOHRING, Algorithmic aspects of comparability graphs and interval graphs, *in* "Graphs and Order" (I. Rival, Ed.), pp. 41–101, Reidel, Boston, 1985.
6. J. H. MILLER AND J. SPINRAD, Incremental modular decomposition, *J. Assoc. Comput. Mach.* **36** (1989), 1–19.
7. M. HABIB AND M. C. MAURER, On the X-join decomposition for undirected graphs, *Discrete Appl. Math.* **1** (1979), 201–207.
8. M. C. GOLUMBIC, "Algorithmic Graph Theory and Perfect Graphs," Academic Press, New York, 1980.
9. A. BLASS, Graphs with unique maximal clumpings, *J. Graph Theory* **2** (1978), 19–24.
10. L. N. SHEVRIN AND N. D. FILIPPOV, Partially ordered sets and their comparability graphs, *Siberian Math. J.* **11** (1970), 497–509.
11. J. VALDES, R. E. TARJAN, AND E. L. LAWLER, The recognition of series-parallel digraphs, *SIAM J. Comput.* **11** (1982), 299–313.
12. D. G. CORNEIL, Y. PERL, L. K. STEWART, A linear recognition algorithm for cographs, *SIAM J. Comput.* **14** (1985), 926–934.
13. W. H. CUNNINGHAM AND J. EDMONDS, A combinatorial decomposition theory, *Canad. J. Math.* **32** (1980), 734–765.
14. B. BOLLOBAS, "Extremal Graph Theory," Academic Press, New York, 1978.
15. T. HIRAGUCHI, On the dimension of partially ordered sets, *Sci. Rep. Kanazawa Univ.* **1**, 77–94.
16. J. SPINRAD, On comparability and permutation graphs, *SIAM J. Comput.* **14** (1985), 658–670.
17. J. B. SIDNEY, Optimal sequencing by modular decomposition, *Oper. Res.* **34** (1986), 606.
18. L. O. JAMES, R. G. STANTON, AND D. D. COWAN, Graph decomposition for undirected graphs, *in* "3rd South-Eastern Conf. Combinatorics, Graph Theory and Computing" (F. Hoffman and R. B. Levow, Eds.), pp. 281–290, Utilitas Math., Winnipeg, 1972.
19. B. BUER AND R. H. MOHRING, A fast algorithm for the decomposition of graphs and posets, *Math. Oper. Res.* **8** (1983), 170–184.
20. C. L. MCCREARY, "An Algorithm for Parsing a Graph Grammar," Ph.D. thesis, University of Colorado, Boulder, 1987.
21. J. SPINRAD, P_4 trees and substitution decomposition, *Discrete Appl. Math.* **39** (1992), 263–291.
22. M. C. MAURER, "Joints et decompositions premières dans les graphes," These 3ème cycle, Université de Paris VI, 1977.
23. A. EHRENFEUCHT, T. HARJU, AND G. ROZENBERG, Incremental construction of 2-structures, *Discrete Math.*, to appear.
24. T. H. CORMEN, C. E. LEISERSON, AND R. L. RIVEST, "Algorithms," MIT Press, Cambridge, MA, 1990.
25. R. H. MOHRING AND F. J. RADERMACHER, Substitution decomposition for discrete structures and connections with combinatorial optimization, *Ann. Discrete Math.* **19** (1984), 257–356.
26. M. CHEIN, M. HABIB, AND M. C. MAURER, Partitive hypergraphs, *Discrete Math.* **37** (1981), 35–50.