An $O(n^2)$ Incremental Algorithm for Modular Decomposition of Graphs and 2-Structures

Algorithmica, **14** (1995), 209-227.

Abstract

This paper gives an $O(n^2)$ incremental algorithm for computing the modular decomposition of 2-structure [1, 2]. A 2-structure is a type of edge-colored graph, and its modular decomposition is also known as the prime tree family. Modular decomposition of 2-structures arises in the study of relational systems. The modular decomposition of undirected graphs and digraphs is a special case, and has applications in a number of combinatorial optimization problems. This algorithm generalizes elements of a previous $O(n^2)$ algorithm of Muller and Spinrad [3] for the decomposition of undirected graphs. However, Muller and Spinrad's algorithm employs a sophisticated data structure that impedes its generalization to digraphs and 2-structures, and limits its practical use. We replace this data structure with a scheme that labels each edge with at most one node, thereby obtaining an algorithm that is both practical and general to 2-structures.

Keywords: modular decomposition, substitution decomposition, prime tree family, graph decomposition, 2-structures.

1 Introduction

A *module* of a graph $G = (V, E)$ is a set X of nodes such that for every node $x \in V - X$, either all members of X are adjacent to x or no element of X is adjacent to x. The *trivial modules* are the empty set, V, and the singleton subsets of V. A graph is *prime* if it has at least three nodes but no nontrivial modules.

Modules and the partitions they induce on the nodes of a graph have strong algebraic properties [4, 5]. The *modular decomposition* of a graph is a unique linear-space representation of the modules the graph, whose number may be exponential in the number of nodes. It is given as a tree whose leaves are the nodes of the graph. It makes possible the efficient solution of some important combinatorial problems on certain classes of graphs, such as comparability graphs, partial orders, and non-prime graphs. Examples include finding maximum-weighted cliques and independent sets, maximum-weighted matchings, minimum colorings, finding the dimension of a partial order, finding the partial orders with a given comparability graph, and solving scheduling problems when jobs are subject to certain types of precedence constraints. Möhring [6] gives a comprehensive review of the literature.

The modular decomposition and its algebraic properties have been extended to to any class of structures where a definition of a module may be given that satisfies certain basic properties [4, 5]. In particular, in [1, 2] the definition of a module satisfying these properties is given for *2-structures*. A 2-structure is a complete graph with an equivalence relation on its edges.

The subject of this paper is an $O(n^2)$ algorithm for modular decomposition of 2-structures. Since modular decomposition of graphs and digraphs is a special case of modular decomposition of 2-structures, the algorithm solves those problems within the same time bound. In addition, 2-structures model systems of binary relations and their modules [7], so the algorithm also computes the modular decomposition for such systems.

The modular decomposition of graphs was first described by Gallai [8], and has since been rediscovered independently by different researchers. Kelly [9] gives a survey of the history of the idea. It is also called *substitution decomposition* [6] and *X-join decomposition* [10], while modules are also known as *closed sets* [8], *clans* [1], *autonomous sets* [6], *partitive sets* [11], *clumps* [12], *stable sets* [13], and *intervals* [7]. The modular decomposition in the case of 2-structures has been known as the *prime tree family* [2], but we will use the term "modular decomposition" for consistency with prior terminology on graphs. The cotree decomposition of cographs [14], and the series-parallel decomposition of General Series-Parallel partial orders [15] are the special cases of the decomposition on these graph classes. A definition of a module that satisfies the requirements of the algebraic model of [5] exists for Boolean functions, set systems, k-ary relations [5] and k-structures [16], so a modular decomposition exists for these structures as well. The decomposition is also closely related to the *split decom-*

position of Cunningham and Edmonds [17], which has been generalized to k-ary relations by Wagner [18].

James, Stanton, and Cowan [19] give an early $O(n^4)$ algorithm for computing the modular decomposition for the case of undirected graphs. Various $O(n^3)$ and $O(nm)$ algorithms for partial orders and graphs have been given [8, 10, 20, 21, 22, 23, 24]. Algorithms with $O(n^2)$ [3], $O(n+mlogn)$ [25], $O(n+m\alpha(m,n))$ [26], and $O(n+m)$ [27] time bounds have recently been developed for arbitrary undirected graphs. Maurer [28] gives an $O(n^4)$ algorithm for arbitrary directed graphs. Ehrenfeucht, Harju, and Rozenberg [29] give an $O(n^3)$ incremental algorithm for arbitrary 2-structures, while Ehrenfeucht, *et. al.* [30] give an $O(n^2)$ divide-and-conquer algorithm for arbitrary 2-structures.

Our algorithm is incremental in the same sense as those given in [3, 29]. That is, let g be a substructure, and let g' be the larger substructure that is obtained by adding some node z to g. Given the modular decomposition for g, the algorithm computes the modular decomposition for g'. Since the modular decomposition for a one-node 2-structure is trivial, the modular decomposition for an arbitrary 2-structure may be obtained by a series of such incremental steps.

A similar time bound is obtained by Ehrenfeucht *et. al.* [30], but that algorithm is not incremental. The $o(n^2)$ algorithms of [25], [26], [27] are based on the $O(n+m)$ cograph recognition algorithm of [14], which is specific to undirected graphs, and no $O(n^2)$ generalization of these decomposition algorithms to arbitrary digraphs or 2-structures has been found.

To get an incremental algorithm for 2-structures, we take the basic approach given in [3] for undirected graphs. That paper gives a characterization of the change to the modular decomposition when a new node is added to a finite graph whose modular decomposition is known. A similar characterization, for arbitrary 2-structures, is given in [29]. Using three devices, which we call $\mathcal{M}(g',z)$, Z_o, and $DL(X,g',z)$, we characterize the incremental change for finite 2-structures succinctly and on an abstract level that is general to finite k-structures and k-ary relations. Z_o is a node of the modular decomposition before the update, $\mathcal{M}(g',z)$ is the family of maximal modules of g that are also modules in g', and $DL(Z_o,g',z)$ is a particular member of $\mathcal{MD}(g')$ which is either the parent of $\{z\}$ or equal to $\{z\}$. Theorem 5.3, below, says that all nontrivial changes to the decomposition tree occur in the subtree rooted at Z_o. Theorem 5.8 characterizes those changes in terms of $\mathcal{M}(g',z)$ and $DL(Z_o,g',z)$.

$\mathcal{M}(g',z)$ and $DL(Z_o,g',z)$ are easily computed. For generalizing the algorithm of [3] to 2-structures without degrading the the $O(n^2)$ time bound, the primary obstacle is generalizing the complex data structure used in that paper for finding Z_o quickly. This data structure, called an *N representation*, allows one to traverse efficiently from parent to child in the modular decomposition, seeking out Z_o. The N representation is specific to undirected graphs. It must also be incrementally updated, and its update is considerably more complex than the one for the modular decomposition itself. This limits both the gener-

ality and the practical usefulness of the decomposition algorithm.

The primary contribution of the current paper is the demonstration that the N representation may be replaced by a simple scheme that involves labeling some of the edges of the graph with a node, called a *split node*. This gives a practical algorithm for undirected graphs, and the labeling scheme is general to directed graphs and 2-structures, thereby allowing the algorithm to be generalized. In this scheme, each edge is labeled at most once over all incremental updates, and the edges that must be labeled after an update are easily identified and labeled in constant time per edge. Thus, maintaining the labels requires $O(n^2)$ time over all updates. The disadvantages of this scheme are that the algorithm requires $\Omega(n^2)$ space in the worst case, and that the time bound is an amortized one, since $\Omega(n^2)$ edges might require labeling after a particular update.

2 Background

All graphs and 2-structures will be assumed to be finite in this paper. Two sets X and Y *overlap* if $X \cap Y$, $X - Y$, and $Y - X$ are nonempty.

A *point-partitive hypergraph* [31] on a universe U is any set family satisfying the following properties:

1. U and its singleton subsets are members of the family;
2. If X and Y are overlapping members of the family, then $X \cap Y$, $X \cup Y$, $X - Y$ and $Y - X$ are members of the family.

Theorem 2.1 *[4, 5]: Let $\mathcal{F}'$ be the members of a point-partitive hypergraph $\mathcal{F}$ that overlap no other. The transitive reduction of the containment relation imposes a tree on $\mathcal{F}'$. Every member of $\mathcal{F}$ is a union of siblings in this tree. Each member of the tree is of one of the following types:*

1. degenerate: *the union of every subfamily of its children is a member of $\mathcal{F}$;*
2. prime: *the only proper subfamilies of its children whose unions are members of $\mathcal{F}$ are those that consist of only one child.*
3. linear: *there is a linear order on the children such that the union of a subfamily is a member of $\mathcal{F}$ if and only if the members of the subfamily are consecutive.*

The modules of a graph or digraph $G = (V, E)$ are a point-partitive hypergraph on V, and the modular decomposition is precisely the tree of Theorem 2.1.

Let D be a set and let $E_2(D)$ denote the set $\{(x, y) : x, y \in Dandx \neq y\}$. A *2-structure* g consists of a set D and an equivalence relation on $E_2(D)$. D is the *domain* of the 2-structure, and is denoted $dom(g)$. The members of D are the *nodes* of the 2-structure, and the members of $E_2(D)$ are its *edges*.

For notational convenience, we will view a 2-structure as an edge-colored complete directed graph, where the color of an edge gives the equivalence class to which it belongs. This also makes it possible to associate equivalence classes in two 2-structures, by coloring them the same color. The *substructure* induced by $X \subseteq dom(g)$, denoted $g|X$, is the 2-structure on domain X where each $(x, y) \in E_2(X)$ is the same color as it is in g.

Let u,v, and w be three nodes of a 2-structure g. We will say that u *distinguishes* v and w if (u, v) and (u, w) are different colors, or if (v, u) and (w, u) are different colors. A *module* in a 2-structure g is a set X of nodes in a 2-structure that are not distinguished by any node in $dom(g) - X$. The empty set is clearly a module, but all modules referred to in this paper will be assumed to be nonempty unless otherwise indicated.

Theorem 2.2 *[1]: The modules of a 2-structure g are a point-partitive hypergraph on $dom(g)$.*

By Theorems 2.2 and 2.1, the modular decomposition is defined for 2-structures. $\mathcal{MD}(g)$ denotes the modular decomposition of g. The *strong modules* are those modules that overlap no other; these are the modules that make up the nodes of $\mathcal{MD}(g)$. Thus, we may say $X \in \mathcal{MD}(g)$ if X is a strong module in g. If X is a non-singleton strong module, then $children_g(X)$ denotes its children in $\mathcal{MD}(g)$.

Note that if X and Y are disjoint modules, all edges of $X \times Y$ are the same color. A *congruence partition* is a partition $\mathcal{P}$ of the nodes such that whenever X and Y are two distinct partition classes of $\mathcal{P}$, all edges of $X \times Y$ are the same color. A partition of $dom(g)$ is a congruence partition iff the partition classes are modules. The *quotient* $g/\mathcal{P}$ is the 2-structure whose nodes are the members of $\mathcal{P}$, and where the color of (X, Y) in $g/\mathcal{P}$ is given by the color of the edges in $X \times Y$ in g. A *factor* is the substructure induced by a member of $\mathcal{P}$. The quotient gives the colors of all edges not in any factor, so the congruence partition and the factors together uniquely specify the 2-structure.

For the purposes of this paper, a 2-structure g is considered to be *isomorphic* to a 2-structure h if there exists a bijection ϕ from $dom(g)$ to $dom(h)$ such that for every $(u, v) \in E_2(dom(g))$, (u, v) is the same color in g as $(\phi(u), \phi(v))$ is in h.

If $\mathcal{F}$ is a family if disjoint sets, then a set S is a *system of representatives* from $\mathcal{F}$ if $S \subseteq \bigcup \mathcal{F}$ and $|S \cap X| = 1$ for each $X \in \mathcal{F}$.

Theorem 2.3 *[1]: If $\mathcal{P}$ is a congruence partition on a 2-structure g and S is a system of representatives from $\mathcal{P}$, then $g/\mathcal{P}$ is isomorphic to $g|S$.*

Every family of disjoint modules is a subset of a congruence partition. Thus, if U,V, and W are three disjoint modules of a 2-structure g, we may say that U *distinguishes* V and W if for arbitrarily selected $u \in U$, $v \in V$ and $w \in W$, u distinguishes v and w. W is *uniform* with respect to U if W is a module

in $g|(U \cup W)$. We may mix nodes and modules in these expressions, saying, for example, that "u distinguishes V and w" in order to avoid the the more cumbersome "$\{u\}$ distinguishes V and $\{w\}$."

Theorem 2.4 *[1, Thm. 4.10]: Let g be a 2-structure, and let X be a module of g. Then the modules of $g|X$ are those modules of g that are subsets of X. The strong modules of $g|X$ that are proper subsets of X are those strong modules of g that are proper subsets of X.*

Let g be a 2-structure. Let $\mathcal{P}$ be a congruence partition on g. If $X \subseteq dom(g)$, then the *image of* X in $g/\mathcal{P}$ is the set $\{Y : Y \in \mathcal{P} and Y \cap X \neq \emptyset\}$. If $\mathcal{F} \subseteq dom(g/\mathcal{P})$, then the inverse image of $\mathcal{F}$ in g is $\bigcup \mathcal{F}$.

Theorem 2.5 *[1, Thm 4.17]: Let g be a 2-structure, and let $\mathcal{P}$ be a congruence partition on g.*

1. *If X be a module of g, then its image in $g/\mathcal{P}$ is a module in $g/\mathcal{P}$. If X is a strong module in g, then its image is a strong module in $g/\mathcal{P}$.*
2. *Conversely, if $\mathcal{F}$ is a module of $g/\mathcal{P}$, then its inverse image is a module of g. If the members of $\mathcal{P}$ are strong modules in g and $\mathcal{F}$ is a strong module in $g/\mathcal{P}$, then its inverse image is a strong module in g.*

Theorem 2.6 *[1, Thm 4.10]: If X is a module of g and $Y \subseteq dom(g)$, then $X \cap Y$ is a (possibly empty) module of $g|Y$.*

A 2-structure g is *prime* if it has only trivial modules and has at least three nodes . It is *degenerate* if $|dom(g)| \geq 2$, and all edges of g are the same color. It is *linear* if $|dom(g)| \geq 2$, its edges are each colored one of two colors, say, a and b, and there exists a linear ordering of its edges such that each $(x, y) \in E_2(dom(g))$ is colored a if x precedes y, and colored b otherwise. In this case, there are actually two such orders, where one is the reverse of the other, but the "linear order" on $dom(g)$ will refer to one of these two orders, selected arbitrarily.

Let X be an arbitrary non-singleton module of a 2-structure g, and let $\mathcal{F}$ be the maximal strong modules of g that are proper subsets of X. Theorems 2.1, 2.4, and 2.5 imply that $(g|X)/\mathcal{F}$ is prime, degenerate, or linear. Thus, we may classify any module, not just the strong modules, as prime, degenerate, or linear.

If D denotes the nodes of an ordinary graph or digraph, then the graph gives a partition of $E_2(D)$ into two equivalence classes: the members of $E_2(D)$ that are edges, and the members of $E_2(D)$ that are not. A graph thus corresponds to a 2-structure, and all results about 2-structures in this paper apply immediately to graphs and digraphs as a special cases. In addition, consider a system of binary irreflexive relations $\{R_1, R_2, \ldots, R_k\}$ defined on a set D. The 2-structure given by the equivalence relation on $E_2(D)$ where $(x, y), (u, v) \in E_2(D)$ are equivalent iff for all $i : 1 \leq i \leq k, uR_iv \Leftrightarrow xR_iy$. 2-structures obtained in this way and their modules were studied in [7], and in [32], where they are called *skeletons*.

3 Additional Definitions

Let X and Y be nonempty disjoint subsets of $dom(g)$. An edge *connects* X and Y if it is a member of $X \times Y$ or a member of $Y \times X$. An edge is *internal to* X if it is a member of $E_2(X)$.

Prefixes, suffixes, and intervals of linear orders have the expected definitions: if $(v_1, v_2, \ldots, v_n)$ is a linear ordering of $dom(g)$, then $(v_1, v_2, \ldots, v_j)$ is a prefix of the linear ordering, $(v_k, v_{k+1}, \ldots, v_n)$ is a suffix, and $(v_i, v_{i+1}, \ldots, v_j)$ is an interval.

Let Y be a module in a 2-structure, g, and let $\mathcal{F}$ be a congruence partition on $g|Y$, where $|\mathcal{F}| > 1$. Then $\mathcal{F}$ is a *prime family* iff $(g|Y)/\mathcal{F}$ is prime, a *degenerate family* iff $(g|Y)/\mathcal{F}$ is degenerate, and a *linear family* iff $(g|Y)/\mathcal{F}$ is linear. In the case of a linear family, the *linear ordering* of the members of $\mathcal{F}$ is given by the linear ordering of $(g|Y)/\mathcal{F}$.

Let a and b be two not-necessarily distinct edge colors in a 2-structure g, and let $x, y \in dom(g)$. Let $R_{a,b}$ be the relation where $xR_{a,b}y$ iff (x, y) is colored a and (y, x) is colored b. If X and Y are disjoint subsets of the domain of g, then by $XR_{a,b}Y$ we denote that every edge in $X \times Y$ is colored a and every edge in $Y \times X$ is colored b.

4 The Algorithm

We will assume that the modular decomposition of a 2-structure g is represented by a tree that has one node for each nonempty strong module of g. The leaves correspond to the singleton modules, and the edges of the tree are given by the transitive reduction of the containment relation on the strong modules. Each node of the tree is labeled with an arbitrary member of the module it represents, so that the color of the edges of g that connect two disjoint modules may be determined in constant time. Clearly, the module represented by any node may be retrieved by collecting the labels of its leaf descendants. In addition, the nodes are labeled prime, degenerate, or linear, as appropriate.

Any action on $\mathcal{MD}(g)$ referred to in the algorithm will be assumed to be implemented as the obvious analogous action on this data structure.

In the rest of this paper, unless otherwise stated, g denotes a substructure whose modular decomposition is known, and z is a node to be added to g to get a larger substructure g'. To get an $O(n^2)$ incremental algorithm, it suffices to show how to compute $\mathcal{MD}(g')$ from $\mathcal{MD}(g)$ in $O(n)$ time.

DEFINITION: $\mathcal{M}(g', z)$ is the family of maximal-cardinality modules of g' that are contained in $dom(g') - \{z\}$. ($\mathcal{M}(g', z)$ is a partition of $dom(g)$ [30].) Z_o is the smallest-cardinality member Z of $\mathcal{MD}(g)$ such that $Z \cup \{z\}$ is a member of $\mathcal{MD}(g')$. $DL(X, g', z)$ denotes a minimum-cardinality $Y \subset dom(g')$ such that $Y = \{z\} \cup (\bigcup \mathcal{F})$ for some $\mathcal{F} \subset children_g(X)$ and such that $\{Y\} \cup (children_g(X) - \mathcal{F})$ is a degenerate or linear family. (Note that $\mathcal{F}$ may be empty,

in which case $DL(X, g', z) = \{z\}$.) If no such set Y exists, $DL(X, g', z) = X$.

The following gives the algorithm for the incremental update.

Algorithm 4.1 *Compute* $\mathcal{MD}(g')$ *from* $\mathcal{MD}(g)$.

1. Find Z_o in $\mathcal{MD}(g)$
2. Compute $DL(Z_o, g', z)$ and $\mathcal{M}(g', z)$
3. Create a new node corresponding to $DL(Z_o, g', z)$ and add it as a child of Z_o in $\mathcal{MD}(g)$. For each $U \in children_g(Z_o)$ that is contained in $DL(Z_o, g', z)$, remove U from the list of children of Z_o.
4. Let $\{z\}$ and the members of $\mathcal{M}(g', z)$ that are contained in $DL(Z_o, g', z)$ be the children of $DL(Z_o, g', z)$.
5. For each $V \in \mathcal{M}(g', z)$ that is contained in $DL(Z_o, g', z)$, V is a union of siblings in $\mathcal{MD}(g)$; move the subtrees rooted at these siblings in $\mathcal{MD}(g)$ so that they are the children of V.
6. Collapse out of the tree any internal nodes that now have only one child so that chains of nodes do not develop.

The correctness of the algorithm follows from Theorems 5.3 and 5.8, below, which are general to k-structures. As stated, Algorithm 4.1 is thus general to k-structures. However, to execute the six steps in $O(n)$ time, we assume a 2-structure whose edge colors are given in an adjacency array.

Like the algorithm of [3], our algorithm begins by identifying which strong modules of g are uniform with respect to z and which are not. This is accomplished by labeling each leaf x of $\mathcal{MD}(g)$ with the colors of (x, z) and (z, x) in g'. Then, using the obvious postorder traversal of $\mathcal{MD}(g)$, each internal node is labeled either with a pair of colors, or else as nonuniform with respect to z, depending on the labelings of its children.

Given the labeling of members of $\mathcal{MD}(g)$ as uniform or nonuniform with respect to z, it is easy to find $\mathcal{M}(g', z)$ in $O(n)$ time from $\mathcal{MD}(g)$. The approach follows from the fact that each member of $\mathcal{M}(g', z)$ is a maximal module of g whose members are not distinguished by z, hence, a maximal union of siblings in $\mathcal{MD}(g)$ that is a module and not distinguished by z. Since this approach computes this set of siblings for each member of $\mathcal{M}(g', z)$, steps 4 and 5 are trivial operations.

$DL(X, g', z)$ may be computed in $O(|children_g(X)|)$ time using elementary arguments involving which children of Z_o are uniform with respect to z, and the colors of edges connecting z with the uniform children.

The only difficult step to carry out in $O(n)$ time is finding Z_o. As with the algorithm of [3], we find Z_o by traversing the ancestors of Z_o from the root. If X is an ancestor, we find whether there is a child Z of X in $\mathcal{MD}(g)$ such that $Z \cup \{z\}$ is the child of $X \cup \{z\}$ in $\mathcal{MD}(g')$. If there is no such child, then $Z_o = X$. Otherwise Z is an ancestor of Z_o, so let $X := Z$ and repeat the operation. If X is degenerate or linear in g, this step reduces to finding $DL(X, g', z)$ and checking whether it contains only one child of X in $\mathcal{MD}(g)$ (Lemma 5.6).

The main problem is performing this operation in constant time per child of X when X is prime. This is the reason for the N representation of [3]. Instead of using that data structure, we note the following. Since $U \cup V$ is not a module in g, U and V are distinguished by a third child X of the prime node, by Theorems 2.4 and 2.5. We assume that all edges connecting U and V are labeled with a *split node* $w \in W$ for some W satisfying the definition of X.

Given such a labeling, it is easy to find if there is a unique child Z such that $Z \cup \{z\}$ is a module in g in constant time per child. Retrieve an edge connecting U and V in order to find its split node w. Since w distinguishes U and V, it must either distinguish z and U or z and V. This means that $U \cup \{z\}$ or $V \cup \{z\}$ is not a module in g'. Thus, in constant time, we may exclude either U or V in the search for Z. Repeating this operation on pairs of remaining candidate children eventually leaves only one child Z as a candidate. $Z \cup \{z\}$ is a module in g' iff Z and $\{z\}$ are not distinguished by any sibling of Z, which may be found in constant time per sibling of Z.

This takes care of finding Z_o in $O(n)$ time. In section 7 , we give the algorithm for updating the labeling of edges with split nodes. The key point for the time bound of that algorithm is that once an edge is labeled with a split node, this label is still valid after all subsequent updates (Lemma 7.2), so it never needs to be relabeled. It is easy to identify those edges that need a split label and that don't already have one (Lemma 7.3), and to compute appropriate labels for them, in constant time per edge. This gives an amortized $O(n^2)$ time bound for updating split node labels. However, when the algorithm runs on a prime 2-structure, all edges are eventually labeled, so it requires $\Omega(n^2)$ space in the worst case.

5 Proof of Correctness of Algorithm 4.1

In this section, we make use only of Theorems 2.2, 2.4, 2.5, 2.3, and 2.6 to prove the correctness of Algorithm 4.1. Because the corresponding theorems are true for k-structures [16] and k-ary relations [5], Algorithm 4.1 is applicable to those structures also, though not within the $O(n^2)$ time bound. However, because the algebraic model of [5] requires only Theorems 2.2, 2.4, and 2.5, the proof is not general to all structures encompassed by that model. In particular, it does not apply to Boolean functions or set systems.

Lemma 5.1 *Z_o exists and is unique.*

Proof: Since $dom(g)$ is a strong module of g and $dom(g) \cup \{z\}$ is a strong module of g', there exists a minimum-cardinality strong module X of g such that $X \cup \{z\}$ is a strong module of g'. Suppose U and V are two such modules. Then $U \cup \{z\}$ and $V \cup \{z\}$ are overlapping strong modules of g', a contradiction. □

Lemma 5.2 *Let $Z \cup \{z\}$ be a module of g' such that $Z \neq \emptyset$.*

1. *X is a module of g' that is disjoint from $Z \cup \{z\}$ iff X is a module of g that is disjoint from Z. $X \cup \{z\}$ is a module of g' that contains $Z \cup \{z\}$ iff X is a module of g that contains Z.*

2. *X is a strong module of g' that is disjoint from $Z \cup \{z\}$ iff X is a strong module of g that is disjoint from Z.*

Proof: Z is a module of g by Theorem 2.6. Let $\mathcal{P}$ be the congruence partition on g consisting of Z and the singleton subsets of $dom(g) - Z$. Let $\mathcal{P}'$ be the congruence partition on g' consisting of $Z \cup \{z\}$ and the singleton subsets of $dom(g) - Z$. $g/\mathcal{P}$ is isomorphic to $g'/\mathcal{P}'$ by Theorem 2.3, since $\mathcal{P}$ and $\mathcal{P}'$ share a system of representatives. Each pair of corresponding sets mentioned in Part 1 consists of the inverse images in g and g' of a common set of nodes of this quotient. Part 1 thus follows from Theorem 2.5.

For part 2, if X is a module of g' that is not strong, then it overlaps a module W of g'. The inverse image in g of the image of W in $g/\mathcal{P}$ is a module of g, by Theorem 2.5, and it overlaps X. Thus, X is not strong in g. By the symmetric proof, if X is a module of g that is not strong, it is not strong in g'. □

Theorem 5.3 *A member of $\mathcal{MD}(g)$ that is disjoint from Z_o is a member of $\mathcal{MD}(g')$. Adding z to a member of $\mathcal{MD}(g)$ that contains Z_o gives a member of $\mathcal{MD}(g')$. All other members of $\mathcal{MD}(g')$ not given by this rule are contained in $Z_o \cup \{z\}$.*

Proof: Follows immediately from Lemma 5.2. □

Lemma 5.4 *If $X \in \mathcal{MD}(g)$ for a 2-structure g and $\mathcal{F}$ is a degenerate or linear family that partitions X, then:*

1. *Each member of $\mathcal{F}$ is a union of one or more members of $children_g(X)$.*

2. *$U \in \mathcal{F}$ is a member of $children_g(X)$ iff there is no partition $\mathcal{F}'$ of U such that $(\mathcal{F} - \{U\}) \cup \mathcal{F}'$ is a degenerate or linear family.*

Proof: (1) X is degenerate or linear. Whether $\mathcal{F}$ is degenerate or linear, there exists a linear ordering of $\mathcal{F}$ such that the union of any family of consecutive members of $\mathcal{F}$ is a module of $g|X$, hence of g, by Theorem 2.4. If some member of $\mathcal{F}$ overlaps a member of $children_g(X)$ or is a proper subset of $children_g(X)$, then some union of consecutive members of $\mathcal{F}$ (*i.e.* a module of g) overlaps a member of $children_g(X)$, contradicting the fact that the members of $children_g(X)$ are strong modules in g. Part (2) then follows from the fact that $children_g(X)$ is a degenerate or linear family. □

Lemma 5.5 *Let $X \subseteq dom(g')$, let $z \in X$, let $\mathcal{F}$ be a congruence partition on $g'|X$, and let $Z \in \mathcal{F}$.*

1. *If* $Z - \{z\} \neq \emptyset$, *then* $(\mathcal{F} - Z) \cup (Z - \{z\})$ *is a degenerate or linear family iff* $\mathcal{F}$ *is.*

2. *If* $|\mathcal{F}| \geq 3$ *and* $\mathcal{F}$ *is a degenerate or linear family, then* $\mathcal{F} - \{Z\}$ *is a degenerate or linear family.*

Proof: If $Z - \{z\} \neq \emptyset$, let $\mathcal{F}' = (\mathcal{F} - \{Z\}) \cup \{Z - \{z\}\}$. $\mathcal{F}'$ is congruence partition on $X - \{z\}$ by Theorem 2.6. $(g'|(X - \{z\}))/\mathcal{F}'$ is isomorphic to $(g'|X)/\mathcal{F}$ by Theorem 2.3, so either both of these quotients are degenerate or linear or neither one is, proving part 1. If $|\mathcal{F}| \geq 3$, let $\mathcal{F}'' = \mathcal{F} - \{Z\}$. For any module $W \neq Z$ of $(g'|X)$ that is a union of members of members of $\mathcal{F}$, $W - Z$ is a module of $(g'|\bigcup \mathcal{F}'')$ by Theorem 2.6. Thus, if $\mathcal{F}$ is a degenerate or linear family, so is $\mathcal{F}''$. □

Lemma 5.6 *If* $X \in \mathcal{MD}(g)$, $X \cup \{z\} \in \mathcal{MD}(g')$, *and* $X - DL(X, g', z)$ *is nonempty, then the members of* $children_{g'}(X \cup \{z\})$ *are* $DL(X, g', z)$ *and the members of* $children_g(X)$ *that are disjoint from* $DL(X, g', z)$.

Proof: $X \cup \{z\}$ is a degenerate or linear, since $X - DL(X, g', z)$ is nonempty. Let Y be the member of $children_{g'}(X \cup \{z\})$ that contains z. If $Y \neq \{z\}$, then $(children_{g'}(X \cup \{z\}) - \{Y\}) \cup \{Y - \{z\}\}$ is a degenerate or linear family that partitions X, by Lemma 5.5. Y is a union of $\{z\}$ and one or more members of $children_g(X)$, by Part 1 of Lemma 5.4, so the lemma follows from the definition of $DL(X, g', z)$, Part 2 of Lemma 5.4, and Lemma 5.2.

Suppose $Y = \{z\}$. If $|children_{g'}(X \cup \{z\})| = 2$, then $DL(X, g', z) = X \cup \{z\}$ by Theorem 2.1, and $X - DL(X, g', z)$ is empty, contradicting the assumption of the lemma. Thus, $|children_{g'}(X \cup \{z\})| \geq 3$. $children_{g'}(X \cup \{z\}) - \{\{z\}\}$ is a degenerate or linear family that partitions X, by Lemma 5.5. Thus, each member of $children_{g'}(X \cup \{z\})$ other than $\{z\}$ is a union of one or more members of $children_g(X)$ by Lemma 5.4. Whether $X \cup \{z\}$ is degenerate or linear, there exists a linear ordering of $children_{g'}(X \cup \{z\})$ such that the union of any consecutive subfamily is a module of g'. If $\{z\}$ is neither at the beginning nor at the end of this ordering, then for each $U \in children_{g'}(X \cup \{z\})$ - $\{\{z\}\}$, there is a module W that is a union of members of $children_{g'}(X \cup \{z\})$, where $\{z\} \subset W \subseteq (X \cup \{z\}) - U$. By Lemma 5.2, $U \in children_g(X)$. If $\{z\}$ is at the beginning or end of the linear ordering, then X is a non-prime module of g', and the maximal prime modules of $g'|X = g|X$ that are proper subsets of X are given by $children_g(X)$. By Theorem 2.4, $children_{g'}(X \cup \{z\}) = \{\{z\}\} \cup children_g(X)$, and $DL(X, g', z) = \{z\}$. □

Lemma 5.7 *If* $DL(Z_o, g', z) \neq \{z\}$, *then let* $\mathcal{P}'$ *denote the members of* $\mathcal{M}(g', z)$ *that are contained in* $DL(Z_o, g', z)$. $children_{g'}(DL(Z_o, g', z)) = \mathcal{P}' \cup \{\{z\}\}$, *and either* $|children_{g'}(DL(Z_o, g', z))| = 2$, *or* $DL(Z_o, g', z)$ *is prime in* g'.

Proof: Suppose there is a module Y of g' such that $\{z\} \subset Y \subseteq DL(Z_o, g', z)$. $Y - \{z\}$ is a module of g, so $Y - \{z\}$ may not overlap any member of $children_g(Z_o)$.

Thus, it consists of one or more members of $children_g(Z_o)$ or is contained in a member of $children_g(Z_o)$. If $Y - \{z\}$ consists of one or more members of $children_g(Z_o)$, then Y and the members of $children_g(Z_o)$ that are disjoint from it are a degenerate or linear family by Lemmas 5.2, 5.4, and Part 1 of Lemma 5.5. $DL(Z_o, g', z) = Y$ by the definition of $DL(Z_o, g', z)$.

If Y is a proper subset of $V \in children_g(Z_o)$, then $V \cup \{z\}$ is a module of g' by Lemma 5.2. In this case $DL(Z_o, g', z) = V \cup \{z\}$, again, by Lemma 5.2. Thus, $V \cup \{z\}$ is strong in g', and V is strong in g, so $Z_o \subseteq V$ by the definition of Z_o, a contradiction.

There is no module Y of g' such that $\{z\} \subset Y \subset DL(Z_o, g', z)$, so $\{z\}$ is a child of $DL(Z_o, g', z)$ and $DL(Z_o, g', z)$ either has two children or is prime. In either case, the siblings of $\{z\}$ in $\mathcal{MD}(g')$ are the members of $\mathcal{M}(g', z)$ that are contained in $DL(Z_o, g', z)$ by Theorem 2.1. □

Theorem 5.8 *A subset of $Z_o \cup \{z\}$ is a strong module in g' if and only if it is one of the following:*

1. *$\{z\}$ or $Z_o \cup \{z\}$;*
2. *$DL(Z_o, g', z)$;*
3. *A descendant of Z_o in $\mathcal{MD}(g)$ that is disjoint from $DL(Z_o, g', z)$;*
4. *A member X of $\mathcal{M}(g', z)$ that is contained in $DL(Z_o, g', z)$ or a member of $\mathcal{MD}(g)$ contained in such an X.*

Proof: We first prove that each named set is a strong module in g', and then show that no other set can be a strong module in g'.

(1) $\{z\}$ is trivially strong in g', and $Z_o \cup \{z\}$ is strong in g' by the definition of Z_o.

(2 & 3) By Lemma 5.6, $DL(Z_o, g', z)$ is either $Z_o \cup \{z\}$ or a child of $Z_o \cup \{z\}$ in $\mathcal{MD}(g')$. In either case, $DL(Z_o, g', z)$ is a strong module in g'. If $DL(Z_o, g', z) \neq Z_o \cup \{z\}$, then by Lemma 5.6, the members of $children_g(Z_o)$ that are disjoint from $DL(Z_o, g', z)$ are the other members of $children_{g'}(Z_o \cup \{z\})$. By Theorem 2.4, the subtrees of $\mathcal{MD}(g')$ rooted at these children are the same as the subtrees of $\mathcal{MD}(g)$ that are rooted at them.

(4) By Lemma 5.7, $children_{g'}(DL(Z_o, g', z))$ is given by $\{z\}$ and the members of $\mathcal{M}(g', z)$ contained in $DL(Z_o, g', z)$. If V is a such a member of $\mathcal{M}(g', z)$, then V is a module of g'. Since it is a module of g' and does not contain z, it is a module of g by Theorem 2.6. By Theorem 2.4, the strong modules of g that are proper subsets of V give the strong modules of g' that are proper subsets of V.

Since all of $\mathcal{MD}(g')$ is specified by the proofs of statements (1) through (4), there can be no other strong modules. □

The correctness of Algorithm 4.1 follows from Theorems 5.3 and 5.8

6 Implementation of Algorithm 4.1 on 2-Structures

Proposition 6.1 *Suppose X and Y are members of $\mathcal{MD}(g)$ that are known to be uniform with respect to z. It may be determined whether z distinguishes X and Y by determining whether z distinguishes any $x \in X$ and $y \in Y$. Since the nodes for X and Y in the data structure for $\mathcal{MD}(g)$ are labeled with such an x and y, this operation takes constant time.*

Algorithm 6.2 *Determine which members of $\mathcal{MD}(g)$ are uniform with respect to z.*

A member of $\mathcal{MD}(g)$ is uniform with respect to z if and only if all of its children are uniform with respect to z and not distinguished by z. The labeling of members of $\mathcal{MD}(g)$ as uniform or nonuniform with respect to z may be accomplished by starting at the leaves of $\mathcal{MD}(g)$ and inductively labeling each internal node according to this rule once all of its children are labeled.

This requires constant time per member of $\mathcal{MD}(g)$, by Proposition 6.1, hence $O(n)$ time.

Algorithm 6.3 *Compute $DL(X, g', z)$ for some $X \in \mathcal{MD}(g)$.*

Suppose all nodes of $\mathcal{MD}(g)$ have been labeled as uniform or nonuniform with respect to z, using Algorithm 6.2. If X is prime, $DL(X, g', z) = X$. If X is degenerate, suppose the edges connecting its children in g are colored a. Let $\mathcal{F} = \{Y\colon Y \in children_g(X), Y \text{ is uniform with respect to } z, \text{ and } Y R_{a,a}\{z\}\}$. $DL(X, g', z) = \{z\} \cup (\bigcup(children_g(X) - \mathcal{F}))$. If X is linear, suppose the edges from earlier to later children in the linear order are labeled a, and the edges from later children to earlier children are labeled b. Let $\mathcal{F}_1$ be the maximal prefix of the linear order on $children_g(X)$ such that for each member Y of $\mathcal{F}_1$, Y is uniform with respect to z and $Y R_{a,b}\{z\}$. Symmetrically, let $\mathcal{F}_2$ be the maximal suffix such that for each member Y of $\mathcal{F}_2$, Y is uniform with respect to z and $Y R_{b,a}\{z\}$. Let $\mathcal{F} = \mathcal{F}_1 \cup \mathcal{F}_2$. $DL(X, g', z) = \{z\} \cup (\bigcup(children_g(X) - \mathcal{F}))$.

Note that $DL(X, g', z)$ is a module in $g'|(X \cup \{z\})$, so $DL(X, g', z) - \{z\}$ is a module in $g|X$. If X is linear, then the children of X contained in $DL(X, g', z)$ are consecutive in the linear ordering of the children of X. The correctness of Algorithm 6.3 is thus immediate. By Proposition 6.1, it runs in $O(|children_g(X)|)$ time.

Lemma 6.4 *Either $DL(Z_o, g', z) = \{z\}$ or $DL(Z_o, g', z)$ contains more than one member of $children_g(Z_o)$.*

Proof: Suppose $DL(Z_o, g', z)$ contains exactly one member V of $children_g(Z_o)$. Since $DL(Z_o, g', z) = V \cup \{z\}$ is strong in g', Z_o is not a minimum-cardinality member Y of $\mathcal{MD}(g)$ such that $Y \cup \{z\}$ is strong in g', a contradiction. □

The following shows how to implement Steps 1-6 of Algorithm 4.1 in $O(n)$ time.

Step 1 *Find Z_o in $\mathcal{MD}(g)$*

$dom(g)$ is a (not necessarily proper) ancestor of Z_o. Given an ancestor X of Z_o, determine either that $Z_o = X$ or else find $Z \in children_g(X)$ that is also an ancestor of Z_o, and iterate with $X = Z$.

To accomplish this if X is degenerate or linear, compute $DL(X, g', z)$ using Algorithm 6.3. By Theorem 5.3 and Lemma 6.4, if $DL(X, g', z)$ contains a unique $Z \in children_g(X)$, then Z is an ancestor of Z_o; otherwise $X = Z_o$. If X is prime. Use the following algorithm:

Algorithm 6.5 *If X is prime in g and $X \cup \{z\}$ is a module in g', find whether there exists $Y \in children_g(X)$ such that $Y \cup \{z\}$ is a module of g'.*

> Let U and V be two members of $children_g(X)$. The split node label of any edge in $U \times V$ gives a node that distinguishes U and V. This node thus distinguishes either z and U or z and V. Without loss of generality, suppose it distinguishes z and U. $U \cup \{z\}$ is not a module of g. We repeat the process on the members of $children_g(Z_o)$ - U, and terminate when only one child Y is left. We may determine whether $Y \cup \{z\}$ is a module of g' by finding whether any other member of X distinguishes Y and $\{z\}$.

By Proposition 6.1, Algorithm 6.5 takes $O(|children_g(X)|)$ time. If X is degenerate or linear, Algorithm 6.3 takes $O(|children_g(X)|)$ time to compute $DL(X, g', z)$. There is one iteration for each ancestor X of Z_o in $\mathcal{MD}(g)$, so Step 1 takes $O(n)$ time.

Step 2: *Find $DL(Z_o, g', z)$ and $\mathcal{M}(g', z)$*

Use Algorithm 6.3 to find $DL(Z_o, g', z)$. To compute $\mathcal{M}(g', z)$, use the following:

Algorithm 6.6 *Compute $\mathcal{M}(g', z)$.*

> Each member of $\mathcal{M}(g', z)$ is a maximal module of g that is uniform with respect to z. Let Y be a member of $\mathcal{MD}(g)$ that is not uniform with respect to z, but which has children that are uniform with respect to z. By Theorem 2.1, if Y is prime, then each uniform child is a member of $\mathcal{M}(g', z)$. If Y is degenerate, the union of each maximal set of uniform children that are not distinguished by z is a member of $\mathcal{M}(g', z)$; these may be found with a radix sort on the children of Y that are uniform with respect to z, using the colors of edges connecting them with z as the sort key. If Y is linear, then the union of each maximal interval of children that are not distinguished

by z on the linear ordering of Y's children is a member of $\mathcal{M}(g', z)$; these may be found with a single

Algorithm 6.6 clearly requires only constant time per child at each internal node Y of $\mathcal{MD}(g)$, so it is $O(n)$.

Step 3: *Create a new node corresponding to $DL(Z_o, g', z)$ and add it as a child of Z_o in $\mathcal{MD}(g)$. For each $U \in children_g(Z_o)$ that is contained in $DL(Z_o, g', z)$, remove U from the list of children of Z_o.*

The members of $children_g(Z_o)$ that are contained in $DL(Z_o, g', z)$ are found during computation of $DL(Z_o, g', z)$, so this step is trivial.

Step 4: *Let $\{z\}$ and the members of $\mathcal{M}(g', z)$ that are contained in $DL(Z_o, g', z)$ be the children of $DL(Z_o, g', z)$.*

It is easily determined in $O(n)$ time which members of $\mathcal{MD}(g)$ are subsets of $DL(Z_o, g', z)$, by marking those members of $\{Z_o\} \cup children_g(Z_o)$ that are found to be subsets of $DL(Z_o, g', z)$ during computation of $DL(Z_o, g', z)$, and then marking all descendants of these marked sets in $\mathcal{MD}(g)$. It is then trivially determined during Algorithm 6.6 which members of $\mathcal{M}(g', z)$ are contained in $DL(Z_o, g', z)$. They are prime, degenerate, or linear if they are prime, degenerate, or linear, respectively, in g, which is again trivially determined from the way they are computed in Algorithm 6.6. $DL(Z_o, g', z)$ is labeled prime iff it has at least three children, by Lemma 5.7, and may be labeled degenerate if the edges connecting its children are all one color and linear if they are two colors.

Step 5: *For each $V \in \mathcal{M}(g', z)$ that is contained in $DL(Z_o, g', z)$, V is a union of siblings in $\mathcal{MD}(g)$; move the subtrees rooted at these siblings in $\mathcal{MD}(g)$ so that they are the children of V.*

The siblings in $\mathcal{MD}(g)$ whose union is V are found when V is computed in Algorithm 6.6. This step is thus trivial.

Step 6 is clearly trivially accomplished in $O(n)$ time.

7 Updating split-node labels

We have shown how to do the incremental update from $\mathcal{MD}(g)$ to $\mathcal{MD}(g')$ in $O(n)$ time. However, Step 1 of Algorithm 4.1 requires that each prime edge be labeled with a split node. This label will be called a *split label.* Recall from Section 2 that an edge (u, v) is *prime* in g if the least common ancestor of u and v in $\mathcal{MD}(g)$ is a prime node X, and a *split node* is a node w that lies in a different child of X from u and v and that distinguishes u and v.

The key elements of the algorithm are that once an edge becomes prime, it remains prime over all subsequent increments, and its initial split-node label remains valid over all subsequent increments (Lemma 7.2). Thus, an edge needs to be labeled at most once over all increments. The edges that first become prime after a particular increment are easy to identify (Lemma 7.3), and label, in constant time per edge. This gives an $O(n^2)$ bound on assignment of split labels over all increments.

Theorem 7.1 *([2], Theorem 4.7): Suppose a member U of $\mathcal{MD}(g)$ is linear or degenerate and it has a non-singleton child V that is linear or degenerate. If X and Y are children of V, and W is a sibling of V, then X distinguishes Y and W.*

Lemma 7.2 *If an edge (u, v) of g is prime, then (u, v) is also prime in g'. Any split node of (u, v) in g is a split node of (u, v) in g'.*

Proof: Let w be a split node for (u, v). u, v, and w are nodes in different children of a prime member X of $\mathcal{MD}(g)$. Suppose there is a module Y in g' that contains two, but not three, members of $\{u, v, w\}$. Then $Y - \{z\}$ is a module of g and $|(Y - \{z\}) \cap \{u, v, w\}| = 2$ by Theorem 2.6, contradicting the fact that X is prime in g. Thus, u,v, and w are in different children of a prime member X' of $\mathcal{MD}(g')$. If (u, v) is prime in g, it is prime in g', and if w is a split node for it in g, it is a split node for it in g'. □

Lemma 7.3 *Any non-prime edge of g that is prime in g' connects two children of $DL(Z_o, g', z)$ in $\mathcal{MD}(g')$.*

Proof: It suffices to establish the following claim: If $U \in \mathcal{MD}(g')$ - $\{DL(Z_o, g', z)\}$, then an edge of g that connects two of U's children in g' is prime in g' only if it is prime in g. The claim follows vacuously whenever U is linear or degenerate in g'. If U is strong in g and g', the claim follows from Theorem 2.4.

If U is not a subset of Z_o, the claim follows from Theorem 5.3. If $Z_o \neq DL(Z_o, g', z)$, Z_o is linear or degenerate in g', establishing the claim when $U = Z_o \neq DL(Z_o, g', z)$. If U is a descendant of Z_o that is not contained in $DL(Z_o, g', z)$, U is strong in g' and in g, by Theorem 5.8, establishing the claim in this case. If U is a non-singleton child of $DL(Z_o, g', z)$, it is a member of $\mathcal{M}(g', z)$, hence a module of g. If it is strong in g, it is strong in both g' and g by Theorem 5.8, and if it is not strong in g, it is linear or degenerate, establishing the claim in either case. If U is a descendant of $DL(Z_o, g', z)$ that is not a child, it is strong in g and g' by Theorem 5.8, establishing the claim in this case. The claim is established in all cases except $U = DL(Z_o, g', z)$. □

Conceptually, the algorithm for updating the split labels works as follows. An edge (x, y) in g needs a split label assignment in g' only if it is non-prime in g by Lemma 7.2, and thus connects two children of a degenerate or linear $W \in \mathcal{MD}(g)$. Let X and Y be children of W that are contained in $DL(Z_o, g', z)$. X and Y are strong modules in g, and since each member of $\mathcal{M}(g', z)$ is a module of g, either X and Y are subsets of the same member of $\mathcal{M}(g', z)$, or every member of X is in a different member of $\mathcal{M}(g', z)$ from every member of Y. Thus, the edges connecting X and Y need a split label update iff X and Y are not subsets of the same member of $\mathcal{M}(g', z)$; otherwise none of them need a split label.

Let the *M partition* for W be the partition of $children_g(W)$ where two children are members of the same partition class iff they are subsets of the same

member of $\mathcal{M}(g', z)$. By the foregoing, the edges connecting children of W that need a split label assignment are those that connect different M-partition classes.

The M partitions for all nodes of $\mathcal{MD}(g)$ are already available: if a child of W is marked nonuniform with respect to z by Algorithm 6.2, then it is the only member of its M partition class; the remaining M partition classes for W are computed explicitly by Algorithm 6.3 when it reaches W.

The implementation of Algorithm 4.1 on 2-structures already gives implicitly an algorithm for finding the split labels of these edges. If A,B are distinct members of $children_{g'}(DL(Z_o, g', z)) - \{\{z\}\}$, then when Algorithm 4.1 is run on g and z, it determines that $DL(Z_o, g', z)$ is prime. To determine this, it is necessary to determine that A and B are distinguished by some third member of of $children_{g'}(DL(Z_o, g', z))$. Since any $c \in C$ is a valid split node for the edges connecting A and B, an algorithm for assigning split labels is found by reexamining Algorithm 4.1 with this idea in mind.

A similar approach applies to finding split node labels on edges that are incident to z.

Algorithm 7.4 *Update split labels of edges that are in g.*

For each linear or degenerate member W of $\mathcal{MD}(g)$ and for each pair $\{S, T\}$ of members of $children_g(W)$ that are contained in $DL(Z_o, g', z)$ and members of different classes of W's M-partition:

1. If S and T are are each uniform with respect to z
   (a) If W is degenerate, then z is a split node for all edges connecting S and T
   (b) If W is linear, then let $\mathcal{S}$ and $\mathcal{T}$ be the M-partition classes that contain S and T. If $\mathcal{S}$ and $\mathcal{T}$ contain adjacent members in the linear order of $children_g(W)$, then z is a split node for the edges connecting S and T. Otherwise, any node in a member of $children_g(W)$ that lies between the members of $\mathcal{S}$ and $\mathcal{T}$ in the linear order is a split node.
2. If at least one of S and T, say T, is nonuniform with respect to z
   (a) If T is linear or degenerate, let $\mathcal{A}$ be a class in the M-partition of $children_g(T)$, and let $A = \bigcup \mathcal{A}$. $T - A$ is nonempty. Any node in A is a split node for the edges connecting S and $T - A$, while any node in $T - A$ is a split node for the edges connecting S and A.
   (b) If T is prime, then when Algorithm 6.5 is run with $X = T$ and $z = s \in S$, it returns, for each $U \in children_g(T)$, a node v of $T - U$ that distinguishes s and U. Assign v as the split label of edges connecting S and U.

Correctness: (1a) If W is degenerate and S and T are uniform with respect to z, then all members of $children_g(W)$ that z does not distinguish from S are

subsets of the same member of $\mathcal{M}(g', z)$. Since S and T are not in the same member of $\mathcal{M}(g', z)$, they are distinguished by z. S, T and $\{z\}$ are subsets of different members of $children_{g'}(DL(Z_o, g', z))$ by Theorem 5.8, so $\{z\}$ is a split node for edges connecting them.

(1b) $\mathcal{S}$ and $\mathcal{T}$ are maximal intervals of the linear order on $children_g(W)$ whose members are not distinguished by $\{z\}$. If $\mathcal{S}$ and $\mathcal{T}$ contain adjacent elements of the linear order, $\{z\}$ distinguishes $\mathcal{S}$ and $\mathcal{T}$ and S, T, and $\{z\}$ are subsets of different members of $\mathcal{M}(g', z)$, hence of $children_{g'}(DL(Z_o, g', z))$ by Theorem 5.8. If a node v lies in a child between $\mathcal{S}$ and $\mathcal{T}$, $vR_{a,b}S$ iff $vR_{b,a}T$, and thus w distinguishes S and T. S, T, and v are subsets of different members of $\mathcal{M}(g', z)$, hence of $children_{g'}(DL(Z_o, g', z))$.

(2a) A node in A distinguishes nodes in S and $T - A$, by Theorem 7.1. Nodes in S, $T - A$, and A are in different members of $\mathcal{M}(g', z)$, hence of $children_{g'}(DL(Z_o, g', z))$.

(2b) By Theorem 5.8, nodes in S and T are in different members of $children_{g'}(DL(Z_o, g', z))$, and if T is prime and nonuniform with respect to z, then nodes in different members of $children_g(T)$ are in different members of $\mathcal{M}(g', z)$, hence of $children_{g'}(DL(Z_o, g', z))$. □

Algorithm 7.2 takes $O(n)$ time to identify the linear and degenerate members of $\mathcal{MD}(g)$ that have nontrivial M-partitions, and constant time per updated edge to update the split labels.

Algorithm 7.5 *Assign split labels to edges incident to z*

1. Let u be an element of Z_o. For each $v \in dom(g) - Z_o$, if (u, v) has a split label, copy it to (z, v) and to (v, z). Do nothing else if $DL(Z_o, g', z)$ is not prime. Otherwise:
2. If Z_o is prime, applying Algorithm 6.5 with $X = Z_o$ finds, for each $U \in children_g(Z_o)$ a node v that distinguishes U and $\{z\}$, since $U \cup \{z\}$ is not a module of g'. Let v be the split label of all edges connecting U and $\{z\}$.
3. If Z_o is degenerate, select $U \in children_g(Z_o)$ that is contained in $DL(Z_o, g', z)$. Let X be the union of members of $children_{g'}(DL(Z_o, g', z))$ - $\{z\}$ that are disjoint from U. Search U for a member u that distinguishes z and x for arbitrary $x \in X$, and assign it as the split label of all edges connecting $\{z\}$ and X. Then select $V \in children_g(Z_o)$ that is contained in X. Search V for a node that distinguishes z and u, and assign it as the split label of all edges connecting $\{z\}$ and $(DL(Z_o, g', z) - \{z\})$ - X.
4. If Z_o is linear, the procedure is identical to the one where Z_o is degenerate, except that U and V must be the first and last members in the linear ordering of $children_g(Z_o)$ that are contained in $DL(Z_o, g', z)$.

Correctness: (1) The labelings in Part 1 require $O(n)$ time, and their correctness follows from Theorem 5.3. Any remaining edges incident to $\{z\}$ that

might be prime connect $\{z\}$ and members of Z_o. However, if $Z_o - DL(Z_o, g', z)$ is nonempty, then $Z_o \cup \{z\}$ is degenerate or linear, so no edges between $DL(Z_o, g', z)$ and $Z_o - DL(Z_o, g', z)$ are prime in g'. Thus, if there are remaining prime edges incident to z, they are internal to $DL(Z_o, g', z)$, and only then if $DL(Z_o, g', z)$ is prime, since $\{z\}$ is one of its children.

(2) The algorithm for finding Z_o when Z_o is prime terminates when it determines that there is no $U \in children_g(Z_o)$ such that $U \cup \{z\}$ is a module of g' (see Step 1). Since there is no such U by Lemma 6.4, the correctness of Part 2 follows. It requires $O(n)$ time, since that is what Algorithm 6.5 requires.

(3) U is not the only member of $children_g(Z_o)$ that is contained in $DL(Z_o, g', z)$, by Lemma 6.4. X must be nonempty by Lemma 6.4, since $DL(Z_o, g', z)$ is assumed to be prime, and no member of $\mathcal{M}(g', z)$, hence no child of $DL(Z_o, g', z)$, may overlap U. If Z_o is degenerate, all edges connecting its children are the same color, which we will call a. U is included in $DL(Z_o, g', z)$ iff there is an edge connecting U and $\{z\}$ that is not labeled a. Thus, U contains a node u that distinguishes $\{z\}$ from the other members of $children_g(Z_o)$. X is a union of members of $children_g(Z_o)$, so all edges between U and X are of color a. Thus, u distinguishes $\{z\}$ and X. Since u is in a different member of $children_{g'}(DL(Z_o, g', z))$ from the members of X, the edges connecting $\{z\}$ and X are correctly labeled. All remaining members of $children_{g'}(DL(Z_o, g', z))$ - $\{\{z\}\}$ are contained in $(DL(Z_o, g', z) - \{z\}) - X$, so symmetric reasoning shows that the edges connecting $\{z\}$ and this set are also correctly labeled.

(4) If Z_o is linear, the proof is similar to the case when Z_o is degenerate. Let $R_{a,b}$ be the edge relation giving the linear order on its children. X must again be nonempty. U is included in $DL(Z_o, g', z)$ iff there is an edge from U to $\{z\}$ that is not labeled a or an edge from $\{z\}$ to U that is not labeled b. However, $U R_{a,b} X$, so U contains a node u that distinguishes $\{z\}$ and X. Since u is in a different member of $children_{g'}(DL(Z_o, g', z))$ from the members of X, the edges connecting $\{z\}$ and X are correctly labeled. Symmetric reasoning shows that the edges connecting $\{z\}$ and $(DL(Z_o, g', z) - \{z\}) - X$ are also correctly labeled.

Clearly, all steps of Algorithm 7.5 may be charged to edges of g' that are incident to $\{z\}$, at constant time per edge, so Algorithm 7.5 takes $O(n)$ time. This concludes the proof that the incremental algorithm takes $O(n^2)$ time to construct $\mathcal{MD}(g)$, for an arbitrary 2-structure.

8 Acknowledgment

The author would like to thank Andrzej Ehrenfeucht for many enlightening conversations about decomposition of 2-structures, Harold Gabow for pointing out its relation to modular decomposition, and an anonymous referee for pointing out the generality of Algorithm 4.1 to k-ary relations and k-structures.

References

[1] A. Ehrenfeucht and G. Rozenberg, Theory of 2-structures, part 1: Clans, basic subclasses, and morphisms, *Theoretical Computer Science*, **70** (1990), 277–303.

[2] A. Ehrenfeucht and G. Rozenberg, Theory of 2-structures, part 2: Representations through labeled tree families, *Theoretical Computer Science*, **70** (1990), 305–342.

[3] J.H. Muller and J. Spinrad, Incremental modular decomposition, *Journal of the ACM*, **36** (1989), 1–19.

[4] R.H. Möhring and F.J. Radermacher, Substitution decomposition for discrete structures and connections with combinatorial optimization, *Annals of Discrete Mathematics*, **19** (1984), 257–356.

[5] R.H. Möhring, Algorithmic aspects of the substitution decomposition in optimization over relations, set systems and boolean functions, *Annals of Operations Research*, **4** (1985/6), 195–225.

[6] R.H. Möhring, Algorithmic aspects of comparability graphs and interval graphs, in *Graphs and Orders* (I. Rival, editor). D. Reidel, Boston, 1985, pp. 41–101.

[7] J.H. Schmerl, Arborescent structures. II: interpretability in the theory of trees, *Transactions of the American Mathematical Society*, **266** (1981), 629–643.

[8] T. Gallai, Transitiv orientierbare graphen, *Acta Math. Acad. Sci. Hungar.*, **18** (1967), 25–66.

[9] D. Kelly, Comparability graphs, in *Graphs and Order* (I. Rival, editor). D. Reidel, Boston, 1985, pp. 3–40.

[10] M. Habib and M.C. Maurer, On the X-join decomposition for undirected graphs, *Discrete Applied Mathematics*, **1** (1979), 201–207.

[11] M.C. Golumbic, *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York, 1980.

[12] A. Blass, Graphs with unique maximal clumpings, *J. Graph Theory*, **2** (1978), 19–24.

[13] L.N. Shevrin and N.D. Filippov, Partially ordered sets and their comparability graphs, *Siberian Math. J.*, **11** (1970), 497–509.

[14] D.G. Corneil, Y. Perl, and L.K. Stewart, A linear recognition algorithm for cographs, *SIAM J. Algebraic and Discrete Methods*, **3** (1985), 926–934.

[15] J. Valdes, R.E. Tarjan, , and E.L. Lawler, The recognition of series-parallel digraphs, *Siam J. Comput.*, **11** (1982), 299–313.

[16] A. Ehrenfeucht and R.M. McConnell, A k-structure generalization of the theory of 2-structures, *Theoretical Computer Science*, (forthcomming).

[17] W.H. Cunningham and J. Edmonds, A combinatorial decomposition theory, *Canadian J. Math.*, **32** (1980), 734–765.

[18] D. Wagner, Decomposition of k-ary relations, *Discrete Math.*, **81** (1990), 303–322.

[19] 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, editors). Utilitas Mathematica, Winnipeg, 1972, pp. 281–290.

[20] B. Buer and R.H. Möhring, A fast algorithm for the decomposition of graphs and posets, *Mathematics of Operations Research*, **8** (1983), 170–184.

[21] M.C. Golumbic, Comparability graphs and a new matroid, *J. Combin. Theory Ser. B*, **22** (1977), 68–90.

[22] W.H. Cunningham, Decomposition of directed graphs, *SIAM J. Algebraic Discrete Methods*, **3** (1982), 214–228.

[23] G. Steiner, *Machine scheduling with precedence constraints*, PhD thesis, University of Waterloo, Waterloo, Ont., 1982.

[24] C.L. McCreary, *An Algorithm for Parsing a Graph Grammar*, PhD thesis, University of Colorado, Boulder, 1987.

[25] A. Cournier and M. Habib, An efficient algorithm to recognize prime undirected graphs, Technical Report R.R. LIRMM 92-023, Laboratoire D'Informatique, de Robotique et de Microelectronique de Montpellier, 1992.

[26] J.P. Spinrad, P_4 trees and substitution decomposition, *Discrete applied mathematics*, **39** (1992), 263–291.

[27] R.M. McConnell and J.P. Spinrad, Linear-time modular decomposition of undirected graphs and efficient transitive orientation of comparability graphs, in *Symposium on Discrete Algorithms*. 1993.

[28] M.C. Maurer, Joints et decompositions premières dans les graphes, These 3ème cycle, Université de Paris VI, 1977.

[29] A. Ehrenfeucht, T. Harju, and G. Rozenberg, Incremental construction of 2-structures, *Discrete Mathematics*, (forthcoming).

[30] A. Ehrenfeucht, H.N. Gabow, R.M. McConnell, and S.J. Sullivan, An $O(n^2)$ algorithm to compute the prime tree family of a 2-structure, *Journal of Algorithms*, (forthcoming).

[31] M. Chein, M. Habib, and M.C. Maurer, Partitive hypergraphs, *Discrete Mathematics*, **37** (1981), 35–50.

[32] J.H. Schmerl and W.T. Trotter, Critically indecomposable partially ordered sets, graphs,tournaments and other binary relational structures, *Discrete Mathematics*, **113** (1993), 191–205.