Computer Science
Technical Report

On the Common Factors in a Set of Linear Orders

Ross M. McConnell *, Fabien de Montgolfier †

February 12, 2004

Technical Report CS-04-102

Computer Science Department
Colorado State University
Fort Collins, CO 80523-1873

Phone: (970) 491-5792 Fax: (970) 491-2466
WWW: http://www.cs.colostate.edu

*Colorado State University rmm@cs.colostate.edu
†LIRMM, Montpellier, montgolfier@lirmm.fr

On the Common Factors in a Set of Linear Orders

Ross M. McConnell [*], Fabien de Montgolfier [†]

February 12, 2004

Abstract

We give an $O(n)$ algorithm for finding the PQ tree of a consecutive-ones matrix if a consecutive-ones ordering is given, where n is the number of columns and an $O(n)$-space representation of the matrix is given. We use this to obtain the modular decomposition of permutation graphs and two-dimensional partial orders in $O(n)$ time when their compact representation with two linear orders is given. More generally, given a set of k linear orders on a set V, we find a decomposition tree that gives a representation of all sets that form consecutive intervals in all of the linear orders. There is a natural associative, commutative intersection operator on such decomposition trees and show how to evaluate it in $O(|V|)$ time. We use these results to obtain a linear time bound for modular decomposition of 2-structures.

1 Introduction

A 0-1 matrix has the *consecutive-ones property* if there exists a permutation of the set of columns such that the 1's in each row occupy a consecutive block. Such a permutation is called a *consecutive-ones ordering.* (See Figure 1).

A family $\mathcal{F}$ of subsets of a set V has the consecutive-ones property if there exists an ordering $(x_1, x_2, ..., x_n)$ of elements of V such that every member of $\mathcal{F}$ consists of a consecutive interval $\{x_i, x_{i+1}, ..., x_j\}$ of the ordering. This is equivalent to the matrix formulation, where the columns of the matrix denote V and the rows are bit-vector representations of the members of $\mathcal{F}$.

In general, the number of consecutive-ones orderings need not be polynomial; there may be $|V|!$ of them. However, the *PQ tree* of a family that has the consecutive-ones property gives a way to represent all of its consecutive-ones orderings using $O(|V|)$ space, as in Figure 1. The PQ tree is a rooted ordered tree whose leaves are the elements of V, and

a	b	c	d	e	f	g	h	i	j	k
1	1	1	1	1	1					
					1	1	1	1	1	1
1	1									
	1	1								
							1	1	1	1
						1	1	1	1	
									1	1

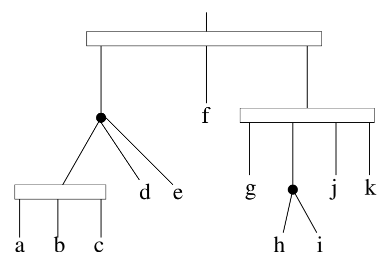

Figure 1: A consecutive-ones ordering of a matrix, and the corresponding PQ tree. The zeros in the matrix are omitted. The ordering of the columns is a consecutive-ones ordering because the 1's in each row are consecutive. The left-to-right leaf order of the PQ tree gives this ordering. Reversing the left-to-right order of children of a Q node (rectangles) or permuting arbitrarily the left-to-right order of children of a P node (points) induces a new leaf order, which is also a consecutive-ones ordering. For instance, permuting the order of children of the left child of the root and reversing the order of children of the right child gives $(d, a, b, c, e, f, k, j, h, i, g)$ as a consecutive-ones ordering. An ordering of columns of the matrix is a consecutive-ones ordering iff it is the leaf order of the PQ tree induced by reversing the children of some set of Q nodes and permuting the children of some set of P nodes.

[*] Colorado State University rmm@cs.colostate.edu
[†] LIRMM, Montpellier, montgolfier@lirmm.fr

whose internal nodes are each labeled either P or Q. The left-to-right leaf order gives a consecutive-ones ordering, and any new leaf order that can be obtained by permuting arbitrarily the children of a P node or reversing the order of children of a Q node is also a consecutive-ones ordering. There are no other consecutive-ones orderings.

One of the most significant applications of PQ trees is in finding planar embeddings of planar graphs [18]. Booth and Lueker used PQ trees to develop an algorithm for determining whether a family of sets has the consecutive-ones property [2]. The algorithm runs in $O(|V| + CardSum(\mathcal{F}))$ time, where $CardSum(\mathcal{F})$ is the sum of cardinalities of members of $\mathcal{F}$

A set family $\mathcal{F}$ with the consecutive-ones property gives rise to an *interval graph*, which has one vertex for each member of $\mathcal{F}$, and an adjacency between two vertices if and only if the corresponding members of $\mathcal{F}$ intersect. Booth and Lueker's result gave a linear-time algorithm for determining whether a given graph is an interval graph, and, if so, finding such a set family $\mathcal{F}$ for it. This problem played a key role during the 1950's in establishing that DNA has a linear topology [1], though linear-time algorithms were unavailable at that time. Variations on this problem come up in the assembly of the genome of an organism from laboratory data [26, 29].

A set X of vertices of a graph $G = (V, E)$ is a *module* iff it satisfies the following conditions for to every $y \in V - X$:

1. Either every element of X is a neighbor of y or no element of X is a neighbor of y;

2. Either y is a neighbor of every element of X or a neighbor of no element of X.

The *modular decomposition* of a directed graph $G = (V, E)$ is a recursive decomposition of a graph into *modules* (Figure 2). The decomposition is an ordered, rooted tree. The nodes of the tree are each a subset of V that is equal to the union of its leaf descendants. Each internal node is labeled *linear* (L), *prime* (P), or *degenerate*(D). A subset of V is a module iff it is a node of the tree, the union of a set of children of a degenerate node, or the union of a set of children of a linear node that is consecutive in the left-to-right order of its children.

A *comparability relation* is the symmetric closure of a partial order. That is, if R is a partial order, and $(a, b) \in R$, then (b, a) is in its symmetric closure. The *transitive orientation problem* is the problem of

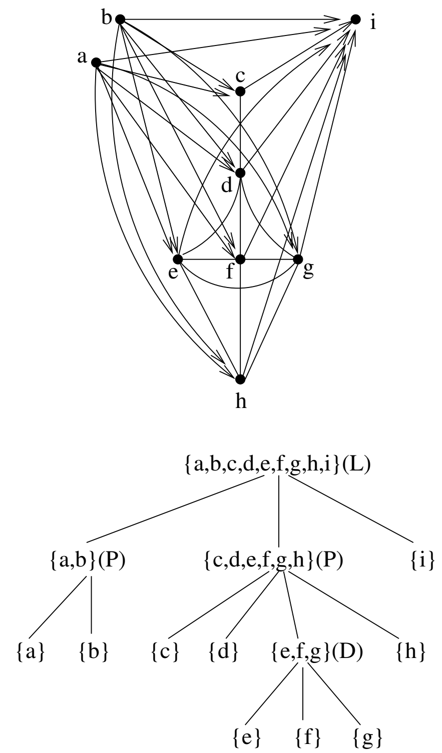

Figure 2: The modular decomposition represents the modules of a graph with an ordered tree whose nodes are subsets of V. Each internal node is labeled *linear* (L), *prime* (P), or *degenerate*(D). A subset of the vertices is a module iff it is a node of the tree, the union of a set of children of a degenerate node, or the union of a set of children of a linear node that is consecutive in the left-to-right order of its children. The modules of the depicted graph that are not nodes of the tree are unions of children of $\{e, f, g\}$, namely, $\{e, f\}$, $\{e, g\}$, and $\{f, g\}$, and unions of consecutive children of $\{a, b, c, d, e, f, g, h, i\}$, namely, $\{a, b, c, d, e, f, g, h\}$ and $\{c, d, e, f, g, h, i\}$. To represent the decomposition, it is not necessary to label internal nodes with the set that they correspond to, as this is given by the union of leaf descendants of the node.
.

inverting this closure operation: given a comparability relation, one must find a partial order that has the comparability relation as its symmetric closure. It is often described as the problem of orienting the edges of an undirected graph to obtain a transitive digraph.

When such an orientation can be found, otherwise NP-hard problems such as finding a maximal clique can be solved in linear time. A transitive orientation of a comparability relation R on a set V can be found in linear $O(|V|+|R|)$ time [22]. The ability to find the modular decomposition in linear time played a key role in this result. It also played a key role in the linear time bound for recognition of *cographs* [5].

The modular decomposition also gives a means of breaking down many NP-complete problems in graphs into smaller subproblems, leading to polynomial-time algorithms on graph classes, such as cographs, which always have a nontrivial decomposition tree.

A partial order R on a set V is a *linear order* if there exists an ordering $(x_1, x_2, ..., x_n)$ of the elements of V such that $(x_i, x_j) \in R$ iff $1 \leq i \leq j \leq n$. The ordering $(x_1, x_2, ..., x_n)$ gives a compact representation in $O(|V|)$ space. A partial order is k *dimensional* if it is the intersection $R_1 \cap R_2 \cap ... \cap R_k$ of k linear orders, and the linear orders give a representation in $O(k|V|)$ space. Every partial order is the intersection of a set of linear orders [10], so every partial order has a dimension. If the k linear orders are given, then the modular decomposition of the partial order can be labeled to give an implicit representation of all sets of k linear orders whose intersection is the partial order.

For $k > 2$, it is NP-complete to determine wither a given partial order is k-dimensional [28]. However, two-dimensional partial orders can be decomposed into two constituent linear orders in linear time [22], hence they can be recognized in linear time. The comparability relation of a two-dimensional partial order R is known as a *permutation graph*; these can be recognized by transitively orienting them and determining whether the result is two-dimensional.

A *2-structure* is a complete graph, with edge set $\{(a,b)|a,b \in V \text{ and } a \neq b\}$, together with a coloring of its (directed) edges. It is *symmetric* if, for every $a,b \in V$, (a,b) and (b,a) are the same color, and *antisymmetric* if, whenever $a,b \in V$, (a,b) and (b,a) are different colors.

Let H be a 2-structure on vertex set V. For $x,y,z \in V$, let us say that z *distinguishes* x and y if (z,x) and (z,y) are of different colors, or (x,z) and (y,z) are of different colors. A module of H is a set X of vertices such that no $y \in V - X$ distinguishes two members of X. The modules of a 2-structure can be represented in the same way as they are in graphs, with a tree whose internal nodes are labeled prime, degenerate, and linear. The modular decomposition of a graph is just a special case: the modules of a graph are the modules of of the 2-structure on the same vertex set that has one color for edges of G and one color for edges of the complement $\overline{G}$ of G. Therefore, everything we show about modules of a 2-structure applies to modules of graphs as a special case.

To get a sparse representation of a 2-structure on vertex set V, delete the edges of the most common color, and let E be the remaining set of colored edges. The edges of the most common color are represented implicitly by their absence from this graph. A 2-structure algorithm can only be considered linear if it runs in $O(|V|+|E|)$ time.

The modular decomposition is a device for representing a potentially large family of subsets of V (the modules) with a tree whose nodes are labeled degenerate, prime, and linear. Let a *partitive family* be any family of subsets of V that admits such a representation. That is, let T be any rooted ordered tree whose leaves are V, whose internal nodes each have at least two children and a label that is either *degenerate*, *prime*, or *linear*. Let $\mathcal{F}(T)$ denote the family of sets where $X \in \mathcal{F}$ iff it is the set of leaf descendants of a node, the union of leaf descendants of a set of children of a degenerate node, or the union of leaf descendants of a set of consecutive children of a linear node. Then $\mathcal{F} = \mathcal{F}(T)$ is a partitive family, and T is $\mathcal{F}$'s *(partitive) decomposition tree*.

Whenever a set family is partitive, its decomposition tree is unique up to labeling of nodes that have only two children (the labeling is irrelevant in this case), the ordering of children of degenerate and prime nodes, and reversal of the order of children of linear nodes.

The modules of a graph or two-structure are a partitive family [27, 13], and the modular decomposition is just their partitive decomposition tree. Other partitive families have played a role in linear time bounds for recognizing circular-arc graphs [19, 23], $O(n + m \log n)$ bounds for recognizing probe interval graphs [25], and finding a certificates for showing that a set family does not have a consecutive-ones ordering or that a graph is not a permutation graph [17].

Two sets X and Y *overlap* if they intersect, but neither is a subset of the other.

Theorem 1. *[4, 27, 12] A set family $\mathcal{F}$ on domain V is partitive iff it has the following properties:*

- *$V \in \mathcal{F}$, $\emptyset \notin \mathcal{F}$, and for all $v \in V$, $\{v\} \in \mathcal{F}$*
- *For all $X, Y \in \mathcal{F}$, if X and Y overlap, then $X \cap Y \in \mathcal{F}$, $X \cup Y \in \mathcal{F}$, $X - Y \in \mathcal{F}$, and $Y - X \in \mathcal{F}$.*

Theorem 1 provides a useful test of whether a set family has a partitive decomposition tree.

1.1 New results

Any algorithm can be made to run in time linear in the size of its input simply by selecting a suitably space-inefficient representation for the input. For instance, many algorithms for NP-complete problems can be made to run in "linear" time by choosing a unary representation for integer inputs. Linearity of an algorithm does not imply an optimal time bound unless the representation of the input is also asymptotically optimal.

When Booth and Lueker's algorithm [2] for finding the PQ tree is applied to a set family that is not known to have the consecutive-ones property, the algorithm either returns the PQ tree, or else rejects the family as not having the consecutive-ones property. The running time of $O(|V| + CardSum(\mathcal{F}))$ is an optimum time bound, since it uses a space-efficient representation of arbitrary set families.

However, when it is applied to a set family that is already known to have the consecutive-ones property, the proof of optimality of the time bound is no longer valid because it assumes an input of size $\Theta(|V| + Cardsum(\mathcal{F}))$. Families with the consecutive-ones property have a representation that is more compact than the standard listing of elements of each member of the family. A consecutive-ones family $\mathcal{F}$ can be represented in $O(|V| + |\mathcal{F}|)$ space by giving a consecutive-ones ordering, and representing each member X of $\mathcal{F}$ in $O(1)$ space by giving the first and last member of the interval occupied by X in this ordering. Our first result is the following:

Theorem 2. *It takes $O(|V| + |\mathcal{F}|)$ time to find the PQ tree of a consecutive-ones family $\mathcal{F}$, given a consecutive-ones ordering and, for each $X \in \mathcal{F}$, the first and last element of X in the ordering.*

An $O(|V| + |E|)$ bound for modular decomposition of arbitrary undirected graphs was first given by the algorithm in [21, 22]; other algorithms with a variety of desirable properties have since followed [6, 8, 15]. The first $O(|V| + |E|)$ bound for directed graphs is given in [9]; a simpler approach is given in [24]. These are all linear in the size of the input when it is given as a listing of elements in each set in the set family of as an adjacency-list representation of the graph. Since these are asymptotically space-optimal representations of arbitrary graphs, these algorithms have provably optimal time bounds.

However, when the input is a permutation graph or two-dimensional partial order, optimality of the time bound again does not follow, since these have an $O(|V|)$ representation in the form of two linear orderings of the vertices. Our second result, which we obtain using Theorem 2, is the following:

Theorem 3. *Given an $O(|V|)$ representation of a two-dimensional partial order or permutation graph using two ordered lists, it takes $O(|V|)$ time to find its modular decomposition.*

If R_1 and R_2 are two linear orders on a set V, their *common factors* are those subsets of V that are consecutive in each of R_1 and R_2. For instance, if $R_1 = (1, 2, 3, ..., 12)$ and $R_2 = (3, 1, 10, 12, 11, 8, 5, 9, 7, 4, 6, 2)$, then $\{4, 5, 6, 7, 8, 9\}$ is a factor, since it appears as the interval $(4, 5, 6, 7, 8, 9)$ in R_1 and as the interval $(8, 5, 9, 7, 4, 6)$ in R_2.

It is well-known that the modules of the two-dimensional partial order $R_1 \cap R_2$ are just the common factors of R_1 and R_2. More generally, if $\mathcal{R} = \{R_1, R_2, ..., R_k\}$ are linear orders on on domain V, then those subsets of V that are consecutive in all of the linear orders are just the modules of a 2-structure $H = H(R_1, R_2, ..., R_k)$, where, for $x, y, u, w \in V$, (x, y) and (u, w) have the same color iff for all $i \in \{1, 2, ..., k\}$, (x precedes y in R_i) $\Longleftrightarrow$ (u precedes w in R_i). That is, they have the same colors if x precedes y and u precedes w in the same subset of the linear orders. Therefore, the common factors of $\mathcal{R}$ are a partitive family whose decomposition tree is given by the modular decomposition of H.

Using Theorem 2, we obtain the following generalization of Theorem 3:

Theorem 4. *If linear orders $\{R_1, R_2, ..., R_k\}$ on domain V are given with k ordered listings of elements of V, it takes $O(k|V|)$ time to find the decomposition tree of their common factors.*

By Theorem 1, the intersection of two partitive set families is a partitive set family. This suggests the problem of finding the decomposition tree T of the intersection $\mathcal{F}_1 \cap \mathcal{F}_2$, given the decomposition trees T_1 and T_2 of $\mathcal{F}_1$ and $\mathcal{F}_2$. We can refer to this as *finding the intersection of two partitive decomposition trees*, which defines an associative, commutative operator $T = T_1 \cap T_2$ on partitive decomposition trees.

The following is given in [24], and plays a key role in the linear time bound for modular decomposition of digraphs:

Theorem 5. *Given partitive decomposition trees* T_1, T_2 *on a set* V*, where* T_1 *and* T_2 *have no linear nodes,* $T_1 \cap T_2$ *can be found in time proportional to the sum of cardinalities of their nodes.*

The difficulties posed by linear nodes are illustrated by the simple case of two trees T_1 and T_2 that each have V as their only internal node. If V is linear in at most one of T_1 and T_2, $T_1 \cap T_2$ has V as its only internal node, so finding $T_1 \cap T_2$ is trivial. If V is linear in both, then $T_1 \cap T_2$ can have a large number of internal nodes. However, given Theorem 4, we can now solve this problem in $O(|V|)$ time, since $T_1 \cap T_2$ is just the decomposition tree of the common factors of the two linear orders on children of V in the two trees. Using a generalization of this trick, we obtain the following:

Theorem 6. *Given arbitrary partitive decomposition trees* T_1 *and* T_2 *on domain* V*, it takes* $O(|V|)$ *time to find* $T_1 \cap T_2$.

Details are given below, but it is instructive to see how it can be proven in the case where there are no degenerate nodes in T_1 or T_2. For each $T_i | i \in \{1,2\}$, it is easy to construct three orderings of V whose common factors have T_i as their decomposition tree. This gives 6 linear orders altogether, and the common factors of these 6 linear orders have $T = T_1 \cap T_2$ as their decomposition tree. The time bound follows from Theorem 4.

The best published bound for modular decomposition of arbitrary 2-structures is $O(|V|^2)$ [11]. However, an $O(|V|+|E|)$ bound for the special case of a symmetric 2-structure with $O(1)$ colors is given in [24], and this is a key element in the the linear time bound for modular decomposition of directed graphs given in that paper. Due to Theorem 4, we can now get a more general bound:

Corollary 7. *It takes* $O(k|V|+|E|)$ *time to find the modular decomposition of a 2-structure that has* k *colors.*

Proof. Let G_i denote the graph on V given by edges of color i, and suppose G_k is the graph of the color class given implicitly by the edges that are absent from E. Find the modular decomposition T_i of each G_i for each i from 1 to $k-1$ using the linear-time modular decomposition algorithm for directed graphs given in [24]. Since the edge sets are disjoint, this takes a total of $O(k|V|+|E|)$ time. The modular decomposition of the 2-structure is given by $T_1 \cap T_2 \cap \ldots \cap T_{k-1}$, which takes $O(k|V|)$ time to find, by Theorem 6. □

Using somewhat more careful methods, we refine these methods to obtain a linear time bound:

Theorem 8. *Let* T_1, $T_2, \ldots, T_k$ *be partitive trees on domain* V*, let* s *be the sum of cardinalities of their non-root internal nodes, and let* l *be the number of the* k *roots that are linear nodes. Given the non-root internal nodes, it takes* $O(s+(l+1)|V|)$ *time to find* $T_1 \cap T_2 \cap \ldots \cap T_k$.

Using Theorem 8, and linear-time modular decomposition of directed graphs, it is easy to obtain the following corollary.

Corollary 9. *It takes* $O(|V|+|E|)$ *time to find the modular decomposition of a 2-structure.*

The proof is similar to that of Corollary 7, but avoids touching isolated vertices in each G_i.

It is worth noting that Theorem 8 gives the following remarkably simple alternative to Booth and Lueker's algorithm for finding the PQ tree of a consecutive-ones family when a consecutive-ones ordering is not given. Let $\mathcal{F} = \{X_1, X_2, \ldots, X_m\}$ be a set family on domain V, and let T_i denote the trivial PQ tree of the one-member set family $\{X_i\}$. The only non-leaf, non-root internal node of T_i is X_i. By Theorem 12 (below), the PQ tree of $\mathcal{F}$ is given by $T_1 \cap T_2 \cap \ldots \cap T_m$, which takes $O(|V| + Cardsum(\mathcal{F}))$ time to compute, by Theorem 8. This is somewhat surprising, since this problem does not assume that the consecutive-ones ordering is already given, yet the algorithm is derived indirectly from Theorem 2, which assumes that it is.

Theorems 3, 4, 6 and Corollary 9 give applications of Theorem 2. We hope that others will arise. Theorem 4, in particular, gives an optimal bound for finding the factors of a set of linear orders. These factors are natural combinatorial objects, and they might have applications in scheduling theory, for example.

2 Sketches of proofs

2.1 Additional background

Our algorithms make extensive use of the following:

Theorem 10. *[14] Given a length-*n *list* L *of real values and a set of* p *intervals* $\{[i_1,j_1],[i_2,j_2],\ldots,[i_p,j_p]\}$ *of* L*, it takes* $O(n+p)$ *time to find a maximum element of* L *in each of the intervals.*

A partitive family is *symmetric* if, in addition to the properties given in Theorem 1, it has the property that whenever X and Y are overlapping members, then their *symmetric difference* $X \Delta Y$ is also a member. It is *antisymmetric* if this is never the case. It is not hard to show that the modules of a symmetric 2-structure are a symmetric partitive family, and that those of an antisymmetric 2-structure are an antisymmetric partitive family. A partitive family is symmetric iff its decomposition tree has no linear nodes with at least three children, and antisymmetric if it has no degenerate nodes with at least three children.

If $\mathcal{F}$ is a family of subsets of a universe V, then $\mathcal{F}$'s *non-overlapping family*, denoted $\mathcal{N}(\mathcal{F})$ is the family of nonempty subsets of V that do not overlap with any member of $\mathcal{F}$.

Theorem 11. *[16] If $\mathcal{F}$ is an arbitrary set family, then $\mathcal{N}(\mathcal{F})$ is a symmetric partitive family.*

Theorem 12. *[16] If $\mathcal{F}$ has the consecutive-ones property, the PQ tree is the decomposition tree of $\mathcal{N}(\mathcal{F})$, where the prime nodes are interpreted as the Q nodes and the degenerate nodes are interpreted as the P nodes.*

If $\mathcal{F}$ is a set family, let its *overlap graph* $G_o(\mathcal{F})$ be the graph that has one vertex for each member of $\mathcal{F}$ and an edge between two vertices iff the corresponding members of $\mathcal{F}$ overlap.

Given a connected component $\mathcal{C}$ of $G_o(\mathcal{F})$, let $\equiv_{\mathcal{C}}$ be an equivalence relation on $\bigcup \mathcal{C}$, where if $x, y \in \bigcup \mathcal{C}$, then $x \equiv_{\mathcal{C}} y$ iff the family of members of $\mathcal{C}$ that contain x is the same as the family of members of $\mathcal{C}$ that contains y. Let $\mathcal{C}$'s *blocks* be the equivalence classes of $\equiv_{\mathcal{C}}$.

Theorem 13. *[20] If $\mathcal{F}$ is a set family on domain V, then $X \subseteq V$ is a node of the decomposition tree of $\mathcal{N}(\mathcal{F})$ iff it is one of the following:*

1. *V or a one-element subset of V;*
2. *$\bigcup \mathcal{C}$ for some connected component $\mathcal{C}$ of $\mathcal{F}$'s overlap graph;*
3. *A block of a connected component of $\mathcal{F}$'s overlap graph.*

2.2 Theorem 2

By Theorem 13, it suffices to find the connected components of $\mathcal{F}$'s overlap graph and, for each component, find the component's union and its blocks. The sum of cardinalities of these unions and blocks is not $O(|V| + |\mathcal{F}|)$, but, since they each correspond to intervals in the consecutive-ones ordering, we can represent each of them in $O(1)$ space by giving the starting and ending position of the interval it occupies in the consecutive-ones ordering. Since the decomposition tree has $|V|$ leaves and each node of the decomposition tree of $\mathcal{N}(\mathcal{F})$ has at least two children, this takes $O(|V|)$ space.

The overlap graph does not have $O(|V| + |\mathcal{F}|)$ size. However, to find the overlap components, it suffices to find a subgraph H of the overlap graph whose connected components are the same as the connected components of the overlap graph, but whose size is $O(|\mathcal{F}|)$. This subgraph will be the union of two spanning forests.

Each block of ones in a consecutive-ones ordering of a matrix can be viewed as an interval on the real line whose endpoints happen to be integers, namely, the column numbers of the first and last interval. Assume that no two rows are identical. It is easy to radix sort the endpoints of the intervals in $O(|V| + |\mathcal{F}|)$ time, and then perturb them by epsilon amounts to obtain a list of endpoints where no two endpoints coincide, without disturbing the overlap relation among the intervals. For instance, subtracting 1/4 from each left endpoint and adding 1/4 to each right endpoint. It is then easy to add epsilon values to a set of coinciding right endpoints without disburbing the containment relation among the intervals. Coinciding left endpoints can be handled similarly. The result is a sorted list of endpoints, where no two endpoints coincide and where the original overlap relation is preserved.

Next, if x is an interval, let $R(x)$ denote the set of intervals that overlap with x and whose right endpoints lie to the right of x. If $R(x)$ is nonempty, let x's *right parent* be the member of $R(x)$ with the rightmost right endpoint. It's *left parent* is defined symmetrically: let $L(x)$ denote the set of intervals that overlap with x and whose left endpoints lie to the left of x. If $L(x)$ is nonempty, then x's *left parent* is the member of $L(x)$ whose left endpoint is leftmost. The *parent graph* is the graph whose vertex set is the intervals and whose edge set is $\{xy |$ one of x and y is the left or right parent of the other$\}$.

Lemma 14. *The connected components of the parent graph are the same as the connected components of the overlap graph.*

Proof. Each edge of the parent graph corresponds to an overlap, so each component of the parent graph is a subset of a component of the overlap graphs. Let us suppose that there are two components C_1 and C_2 of the parent graph that are subsets of the same component C of the overlap graph. We may select C_1 and C_2 such that there is an edge of the overlap graph from a member a of C_1 to a member b of C_2. We will now derive a contradiction.

Suppose without loss of generality that the left endpoint of b is to the left of a. Let $x_2 = a$ and $y_1 = b$. Since $y_1 \in L(x_2)$, x_2 has a left parent, so let x_1 be x_2's left parent. Similarly, let y_2 be y_1's right parent. Since $x_1 \in C_1$, $x_1 \neq y_1$, so x_1's left endpoint is to the left of y_1's. Similarly, y_2's right endpoint is to the right of x_2's. If x_1 fails to overlap y_1, then it contains it and overlaps y_2. Similarly, y_2 overlaps one of x_1 and x_2.

This shows (for $k = 2$) that there exists a sequence $(x_1, y_1, x_2, y_2, ..., x_k, y_k)$ such that $\{x_1, x_2, ..., x_k\} \subseteq C_1$, $\{y_1, y_2, ..., y_k\} \subseteq C_2$, the right endpoints of $(x_2, y_2, ..., x_k, y_k)$ are an increasing sequence, y_k overlaps a member of $\{x_1, x_2, ..., x_k\}$, and x_k overlaps a member of $\{y_1, y_2, ..., y_k\}$. So let us select such a sequence $(x_1, y_1, x_2, y_2, ..., x_k, y_k)$ of maximum size.

Let x_i be a member of $\{x_1, x_2, ..., x_k\}$ that overlaps y_k. Since $y_k \notin C_1$, it is not x_i's right parent, so let x_{k+1} be x_i's right parent, which must be in C_1 and have its right endpoint to the right of y_k. Since $x_{k+1} \notin C_2$, it is not y_k's right parent, so let y_{k+1} be y_k's right parent. Then y_{k+1} is in C_2 and its right endpoint is to the right of x_{k+1}. The new sequence satisfies the conditions for $k+1$, contradicting our choice of k. □

To find the right parents, we create a sorted list L of left endpoints. We label each of these with the matching right endpoint. For each interval $[a, b]$, the set of left endpoints in $(a, b]$ defines an interval of L, which is easily found off-line for all intervals in the set of intervals. The right parent of $[a, b]$ is just the maximum right endpoint that occurs in this interval. By Theorem 10, this may be found for all intervals in $O(|V| + |\mathcal{F}|)$ time. The left parents can be found by a symmetric operation.

This gives the connected components of the overlap graph of $\mathcal{F}$ in $O(|V| + |\mathcal{F}|)$ time. To get the blocks of the components, we may number the $O(|\mathcal{F}|)$ components, label each member of $\mathcal{F}$ with its component number, and then radix sort all beginning and ending positions of members of $\mathcal{F}$ using component number as primary sort key and position as the secondary sort key. As a tertiary key, use 0 for a left endpoint and 1 for a right endpoint; this ensures that when a set of endpoints in the component are tied, the left endpoints in the set come before the right endpoints in the sort. This takes $O(|V| + |\mathcal{F}|)$ time and gives, for each component, a sorted list of endpoints of members of the component. We may obtain the blocks as follows. Treat a left endpoint at position i as occurring just before i and a right endpoint as occurring just after it. Each block of the component is a set of elements of V that occur between consecutive endpoints of the sorted list.

2.3 Theorems 3 and 4

We sketch the proof of Theorem 4, since Theorem 3 is a special case.

If $\mathcal{F}$ is a partitive set family on set V, a *factorizing permutation* of $\mathcal{F}$ is an ordering of elements of V such that the set represented by each node of $\mathcal{F}$'s decomposition tree is consecutive [3]. It is *strong* if, whenever $C_1, C_2, ..., C_k$ are children of a linear node U, the intervals occupied by C_1, $C_2, ..., C_k$ match the linear order of children of U. When $\mathcal{F}$ is antisymmetric, a strong factorizing permutation is a consecutive-ones ordering.

Lemma 15. *If $\mathcal{F}$ is a partitive family on domain V and $R = (x_1, x_2, ..., x_n)$ is a factorizing permutation, then the subfamily $\mathcal{F}'$ of members of $\mathcal{F}$ that are consecutive in R is a partitive family, and its decomposition tree is obtained from $\mathcal{F}$'s decomposition by relabeling each degenerate node as linear, and making the order of its children consistent with the order in which they appear in R.*

When $\mathcal{F}$ and $\mathcal{F}'$ are as in Lemma 15, it is easy to see that $\mathcal{F}'$ is a maximal consecutive-ones subfamily of $\mathcal{F}$.

Definition 16. [3] Let $(x_1, x_2, ..., x_n)$ be a factorizing permutation of the modules of a 2-structure. Let x_i, x_{i+1} be two consecutive elements. If x_i and x_{i+1} are distinguished by an element earlier than i in the ordering, let p be the minimum index such that x_p distinguishes x_i and x_{i+1}. Then $\{x_p, x_{p+1}, ..., x_i\}$ is a *fracture for* i. Similarly, if x_i and x_{i+1} are distinguished by elements greater than $i+1$, then let q be the maximal index such that x_q distinguishes them; $\{x_{i+1}, x_{i+2}, ..., x_q\}$ is a fracture for i. The *fractures* of the factorizing permutation are just the family of sets that are fractures for any of the indices from 1 to n.

Theorem 17. *Let H be a 2-structure on V and $R = (x_1, x_2, ..., x_n)$ be a strong factorizing permutation for its modules. Then the modular decomposition of H is given by the PQ tree of the fractures, where the labeling of internal nodes is given by the following rule:*

- *The Q nodes are interpreted as prime nodes.*

- *A P node is interpreted as a degenerate node if the edges of* H *that go between its children are symmetric, and it is interpreted as a linear node otherwise.*

Proof. The PQ tree of the family $\mathcal{F}_1$ of the fractures is the partitive decomposition tree of the family $\mathcal{N}(\mathcal{F}_1)$ of subsets of V that overlap no fracture, by Theorem 13. To get the decomposition tree of the subfamily $\mathcal{F}_2$ of $\mathcal{N}(\mathcal{F}_1)$ consisting of members of $\mathcal{N}(\mathcal{F}_1)$ that are consecutive in R, we must change the order the children of each degenerate node of the decomposition tree of $\mathcal{N}(F_1)$ to be consistent with R, and change their label to linear, by Lemma 15. Therefore, the partitive decomposition tree of $\mathcal{F}_2$ is the PQ tree of the factors, except for the relabeling of P nodes as linear and Q nodes as prime.

Let $\mathcal{F}_3$ be the modules of H. The subfamily $\mathcal{F}_4$ of modules of H that are consecutive in R is a maximal consecutive-ones subfamily of $\mathcal{F}_3$, so its decomposition tree is the same as that of $\mathcal{F}_3$ except that degenerate nodes are relabeled linear, by Lemma 15. It is easy to see that a consecutive set in R that overlaps a factor cannot be a module, and that a consecutive set in R that overlaps no factor is a module. Therefore, $\mathcal{F}_4 = \mathcal{F}_2$. The modular decomposition of H must be the PQ tree of the factors, except that Q nodes are relabeled prime and P nodes are relabeled degenerate or linear. A node of the modular decomposition of H that is known to be either linear or prime must be linear iff the edges of H that go between the children are antisymmetric, and degenerate iff the edges of H that go between the children are symmetric [13]. □

2.3.1 The algorithm

Given k linear orders $\{R_1, R_2, ..., R_k\}$, recall that their common factors are the modules of $H = H(R_1, R_2, ..., R_k)$, defined in Section 1.1. R_1 is a strong factorizing permutation for the common factors. By Theorem 17, to obtain an $O(k|V|)$ bound for finding the common factors of $\{R_1, R_2, ..., R_k\}$, it suffices to give an algorithm for find the fractures H induces in R_1 in $O(k|V|)$ time.

Let $R_1 = (x_1, x_2, ..., x_n)$. Then for $x_j \notin \{x_i, x_{i+1}\}$, x_j distinguishes x_i and x_{i+1} in H iff there exists R_p such that $2 \leq p \leq k$ where x_j falls between x_i and x_{i+1}. For $R_p = (y_1, y_2, ..., y_n)$, we create a list $L_1 = (p_1, p_2, ..., p_n)$, where p_i denotes the position j of x_i in R_p. That is $p_i = j$ such that $y_j = x_i$. We also create a list $L_2 = (q_1, q_2, ..., q_n)$, where q_i denotes the index of y_i in R_1. That is, $q_i = j$ such that $x_j = y_i$.

To find the maximum r such that x_r lies between x_i and x_{i+1} in R_p, we use L_1 to look up the positions p_a, p_b of x_i and x_{i+1} in R_p. This takes $O(1)$ time if L_1 is implemented with an array. We then find the maximum value that lies in the interval $[p_a, p_b]$ of L_2, and this gives the index r of x_r. By Theorem 10, we can perform this last lookup for all i from 1 to n in $O(n)$ time. Repeating this for all R_p such that $2 \leq p \leq n$ yields $k-1$ such r's for each pair x_i, x_{i+1} in $O(nk)$ time. The maximum of these is the index s of the rightmost vertex x_s in R_1 that distinguishes x_i, x_{i+1}. If $s > i+1$, then $\{x_{i+1}, x_{i+2}, ..., x_s\}$ is one of at most two possible fractures generated by $\{x_i, x_{i+1}\}$.

The other fracture generated by each $\{x_i, x_{i+1}\}$ can be found by symmetry, inverting the roles of i and $i+1$ and min and max. This also takes $O(nk)$ time. Therefore the fractures induced in R_1 by H can be found in $O(nk)$ time.

This proves Theorem 4, and Theorem 3 follows as a special case, since the factors are the same as the modules when the relation is two-dimensional. This gives modular decomposition of permutation graphs in the same time bound, since the modular decomposition of a permutation graph is obtained from that of its transitive orientation by relabeling linear nodes as degenerate.

2.4 Theorem 6

Let us first consider the case where T_1 and T_2 have no degenerate or prime internal nodes. For each T_i, we may construct three linear orders on V whose common factors have T_i as their decomposition tree, as follows. Arrange each node's children according to their implied linear order. Get the first linear order by listing the elements in the leaves according to their left-to-right order in this ordered tree. Then, reverse the order of children at each node that is at odd depth in the tree and once again list the ordering of elements in the leaves to obtain the second linear order. Finally, reverse the order of children at each node that is at even depth, and repeat the operation to obtain the third linear order. (This last step is unnecessary, but convenient when we generalize to trees that have nodes that are not linear.)

It is easy to see that a subset of V is a common factor of these three linear orders iff it is a union of consecutive children of a linear node in T_i. Therefore, T_i is the the decomposition tree of the common factors. It follows that $T = T_1 \cap T_2$ is the decomposition tree of the common factors of these 6 linear orders, and it can be obtained in $O(|V|)$ time with the algorithm of Theorem 4.

Next, let us consider what happens when prime nodes are allowed. Once again we obtain three linear orders to represent each T_i, and find $T = T_1 \cap T_2$ by applying Theorem 4 to the resulting 6 linear orders. To find three linear orders for T_i, we once again order children of internal nodes three times and read an ordering from the leaves. The orderings of children of linear nodes are handled as before. At each prime node P, permute the order of children as follows. Let $C_1, C_2, ..., C_p$ be an arbitrary ordering of children of P; this is their ordering used in obtaining the first linear order. In the second iteration, concatenate the even-numbered children, followed by the odd-numbered children, as follows: $(C_2, C_4, ..., C_{p-(p \bmod 2)}, C_1, C_3, ..., C_{p-(1-(p \bmod 2))})$ to obtain the new ordering of children. For the third iteration, reverse the roles of the odd- and even-numbered children: $(C_1, C_3, ..., C_{p-(1-(p \bmod 2))}, C_2, C_4, ..., C_{p-(p \bmod 2)})$. It is easy to see that the three linear orders again have T_i as their decomposition tree.

Let us now allow degenerate nodes. We assign an order to the children of each degenerate node and treat it as a linear node. The problem of intersecting these trees reduces to the foregoing case. When we are done, the intersection of the trees has some nodes wrongly labeled as linear nodes, when they should be degenerate, and we detect these cases and relabel them.

If $X \subseteq V$, we can find the maximal nodes of T_1 that are subsets of X in $O(|X|)$ time, by marking all nodes that are subsets of X. When a node is marked, it increments a *marked-children* counter in its parent that tells how many marked children the parent has. When a node's marked-children counter reaches its degree, the node is marked. Marking the leaves while observing these rules causes all nodes of T_1 that are subsets of X to be marked. Any marked node U with an unmarked parent W is a maximal node of T_1 that is a subset of X. Since each internal node has at least two children, this takes time proportional to the number of marked leaves, which is $O(|X|)$. Moreover, for each unmarked node with marked children, we can obtain a list of its marked children.

For each node U of T_2, we may perform this operation on T_1, by letting $X = U$. We may do the same for T_2 using nodes of T_1. The results of these markings allows us to order the children of degenerate nodes of T_1 and and T_2 and relabel them linear, obtaining T_1' and T_2', so that some maximal antisymmetric subfamily $\mathcal{F}'$ common to T_1 and T_2 retains this status for T_1' and T_2'. Then $\mathcal{F}'$ is a maximal antisymmetric subfamily of the family represented by T, so its decomposition tree $T' = T_1' \cap T_2'$ is the same as T except that some degenerate nodes have been relabeled linear. Since T_1' and T_2' have no degenerate nodes, $T' = T_1' \cap T_2'$ can be found in $O(|V|)$ time by the tree intersection algorithm given above for this case. Detecting nodes that must be relabeled linear to obtain T is easily accomplished by finding their least common ancestors in T_1 and T_2 using the marking algorithm; the maximal nodes of T_1 or T_2 that are subsets of a node of T' are the least common ancestor or a set of children of the least common ancestor.

The bottleneck is applying the marking algorithm on T_1 repeatedly for each node of T_2, which takes time proportional to the sum of cardinalities of nodes in T_2. However, we can get this down to $O(|V|)$ by observing that when Y is the parent of X in T_2, performing the marking operation with Y repeats all of the marking operations performed with X. Therefore, as we work inductively up T_2 processing nodes, we can continue the marking operation of each node Y at the points in T_1 where the marking of the children left off. The marking proceeds monotonically up T_1, and takes $O(|V|)$ time. Similarly, we get an $O(|V|)$ bound when marking T_2 with nodes of T_1, or when marking T_1 and T_2 with nodes of T'.

2.5 Theorem 8

Let $T_1, T_2, ..., T_k$ be as in Theorem 8.

Lemma 18. *For each* T_i *and every non-prime node* Z *of* $T = T_1 \cap T_2 \cap ... \cap T_k$*, there exists a non-prime node* X_i *of* T_i *such that each child of* Z *in* T *is a union of one or more children of* X_i.

For Theorem 8, we find the connected components of the overlap graph of the set of non-root internal nodes of $T_1, ..., T_k$, using the algorithm for overlap components given in [7]. This takes time proportional to $|V|$ plus the sum of cardinalities of non-root internal nodes in $T_1, ..., T_k$.

We then find the unions of these connected components and their blocks, just as in Theorem 13. The Hasse diagram of the containment relation on V, the unions of connected components, their blocks, and the singleton subsets of V is a tree, which we may find in time proportional to $|V|$ plus the sum of cardinalities of non-root internal nodes of T_1 through T_k. The main technique is radix sorting.

Up to here, the algorithm is a straightforward generalization of the one for symmetric partitive families given in [24]; this is the result of relabeling each linear node as degenerate and finding the intersection T'. We must now reflect the additional constraints imposed by linear nodes in T_1 through T_k.

To do this, we assign each degenerate node Z of T' a representative element $z \in Z$. We then identify for each node Z of this tree T' the node X_i of each T_i given by Lemma 18. If X_i is linear, the linear order on its children imposes linear order on representatives of children of Z, which implies a linear order on children of Z. We collect all such linear orders on children of Z.

This is where Theorem 4 plays a critical role: we use its algorithm to find the decomposition tree of the common factors of these linear orders. The internal nodes of this decomposition tree can be spliced into T' between Z and its children to reflect the constraints imposed by linear nodes of $T_1 \ldots T_k$ on what unions of children of Z can be members of $\mathcal{F}$. Since the representatives of children of Z are members of each linear node that contributes a linear order, it is easy to see that this can be accomplished at all nodes of T' while staying within a time bound proportional to the the sum of cardinalities of linear nodes in $T_1, \ldots, T_k$.

References

[1] S. Benzer. On the topology of the genetic fine structure. *Proc. Nat. Acad. Sci. U.S.A.*, 45:1607–1620, 1959.

[2] S. Booth and S. Lueker. Testing for the consecutive ones property, interval graphs, and graph planarity using PQ-tree algorithms. *J. Comput. Syst. Sci.*, 13:335–379, 1976.

[3] C. Capelle, M. Habib, and F. d Montgolfier. Graph decomposition and factorising permutations. *Discrete Mathematics and Theoretical Computer Science*, 5:55–70, 2002.

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

[5] D. G. Corneil, Y. Perl, and L. K. Stewart. A linear recognition algorithm for cographs. *SIAM J. Comput.*, 3:926–934, 1985.

[6] A. Cournier and M. Habib. A new linear algorithm for modular decomposition. In Sophie Tison, editor, *Trees in Algebra and Programming, CAAP '94, 19th International Colloquium*, volume 787 of *Lecture Notes in Computer Science*, pages 68–82. Springer Verlag, Edinburgh, UK, 1994.

[7] E. Dahlhaus. Parallel algorithms for hierarchical clustering, and applications to split decomposition and parity graph recognition. *Journal of Algorithms*, 36:205–240, 2000.

[8] E. Dahlhaus, J. Gustedt, and R. M. McConnell. Efficient and practical modular decomposition. *Proceedings of the eighth annual ACM-SIAM symposium on discrete algorithms*, 8:26–35, 1997.

[9] F. de Montgolfier. Modular decomposition of tournaments, directed graphs and 2-structures: linear algorithms. Technical Report RR01379, LIRMM, 2001.

[10] B. Duschnik and E. W. Miller. Partially ordered sets. *Amer. J. Math.*, 63:600–610, 1941.

[11] A. Ehrenfeucht, H. N. Gabow, R. M. McConnell, and S. J. Sullivan. An $O(n^2)$ divide-and-conquer algorithm for the prime tree decomposition of two-structures and modular decomposition of graphs. *Journal of Algorithms*, 16:283–294, 1994.

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

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

[14] H.N. Gabow, J.L. Bentley, and R. E. Tarjan. Scaling and related techniques for geometry problems. *Proceedings of the ACM symposium on theory of computing*, 16:135–143, 1984.

[15] M. Habib, F. de Montgolfier, and C. Paul. A simple linear-time modular decomposition algorithm for graphs, using order extending. *LIRMM Research Report 03007*, pages 1–26, 2003.

[16] W.L. Hsu and R.M. McConnell. PC trees and circular-ones arrangements. *Theoretical Computer Science*, 296:59–74, 2003.

[17] D. Kratsch, R.M. McConnell, K. Mehlhorn, and J.P. Spinrad. Certifying algorithms for recognizing interval graphs and permutation graphs. *Proceedings of the Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 866–875, 2003.

[18] A. Lempel, S. Even, and I. Cederbaum. An algorithm for planarity testing of graphs. In P. Rosenstiehl, editor, *Theory of Graphs*, pages 215–232. Gordon and Breach, New York, 1967.

[19] R. M. McConnell. Linear-time recognition of circular-arc graphs. *Proceedings of the 42nd Annual IEEE Symposium on Foundations of Computer Science (FOCS01)*, 42:386–394, 2001.

[20] R. M. McConnell. A certifying algorithm for the consecutive-ones property. *Proceedings of the 15th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA04)*, 15:761–770, 2004.

[21] R. M. McConnell and J. P. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms*, 5:536–545, 1994.

[22] R. M. McConnell and J. P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201(1-3):189–241, 1999.

[23] R.M. McConnell. Linear-time recognition of circular-arc graphs. *Algorithmica*, 37:93–147, 2003.

[24] R.M. McConnell and F. de Montgolfier. Linear-time modular decomposition of directed graphs. *Discrete Applied Math, to appear*.

[25] R.M. McConnell and J.P. Spinrad. Construction of probe interval models. *Proceedings of the Thirteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 866–875, 2002.

[26] F.R. McMorris, C. Wang, and P. Zhang. On probe interval graphs. *Discrete Applied Mathematics*, 88:315–324, 1998.

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

[28] M. Yannakakis. The complexity of the partial order dimension problem. *SIAM J. Algebraic and Discrete Methods*, 3:303–322, 1982.

[29] P. Zhang. United states patent: Method of mapping DNA fragments, available at www.cc.columbia.edu/cu/cie/techlists/patents/5667970.htm. July 3, 2000.