A generic approach to decomposition algorithms, with an application to digraph decomposition*

Binh-Minh Bui-Xuan[1], Pinar Heggernes[1], Daniel Meister[2], and Andrzej Proskurowski[3]

[1] Department of Informatics, University of Bergen, Norway, buixuan@ii.uib.no, buixuan@lip6.fr, pinar.heggernes@ii.uib.no
[2] Theoretical Computer Science, University of Trier, Germany, daniel.meister@uni-trier.de
[3] Department of Information and Computer Science, University of Oregon, USA, andrzej@cs.uoregon.edu

Abstract A set family is a collection of sets over a universe. If a set family satisfies certain closure properties then it admits an efficient representation of its members by labeled trees. The size of the tree is proportional to the size of the universe, whereas the number of set family members can be exponential. Computing such efficient representations is an important task in algorithm design. Set families are usually not given explicitly (by listing their members) but represented implicitly.
We consider the problem of efficiently computing tree representations of set families. Assuming the existence of efficient algorithms for solving the Membership and Separation problems, we prove that if a set family satisfies weak closure properties then there exists an efficient algorithm for computing a tree representation of the set family. The running time of the algorithm will mainly depend on the running times of the algorithms for the two basic problems. Our algorithm generalizes several previous results and provides a unified approach to the computation for a large class of decompositions of graphs. We also introduce a decomposition notion for directed graphs which has no undirected analogue. We show that the results of the first part of the paper are applicable to this new decomposition. Finally, we give efficient algorithms for the two basic problems and obtain an $\mathcal{O}(n^3)$-time algorithm for computing a tree representation.

1 Introduction

The running time of an algorithm that finds a solution by exhaustive search over the family of possible solutions is dependent on the number of possible solutions. For most practical applications, the number of possible solutions is large compared to the input size itself, which makes this brute-force algorithm very inefficient. If the family of possible solutions has some structure, an efficient and compact representation of the family may be a key step to designing an

* This work was supported by the Research Council of Norway. The first author was supported by the French National Research Agency, project MAGNUM.

efficient algorithm for the problem. As an example, Gabow used an efficient representation for minimum cuts in a network to improve max-flow algorithms [12].

The above example is one of many possible applications of space-efficient representations, and different specifications of the problem do exist. The fundamental problem that we are considering is the following: given a set family over a finite universe, we ask for an efficient algorithm for computing a space-efficient representation of the set family. An extremal example of such a set family is the power set of the universe, which is exponential in the size of the universe. The space-efficient representation that we are aiming at is a tree whose nodes are labeled, so that each family member can be determined from the tree and only family members can be determined from the tree. The size of the tree, i.e., the number of nodes of the tree, must be linear in the size of the universe. We will call such a representation a *tree representation*. By a simple counting argument, it is clear that a tree representation cannot exist for arbitrary set families. It is, however, known that efficient representations do exist for set families that satisfy certain closure conditions [7,10,8,6,12,3,5]; a summary of such results and examples for their application can be found in [2]. The resulting tree representation is dependent on the actual closure conditions. In this paper, we focus on so-called *weakly partitive crossing families*, which are set families that are closed under union, intersection and difference of its crossing members. Two members *cross* if their intersection is non-empty, none of the two is a subset of the other and the union of the two members does not cover the whole universe. Many studied set families are in fact weakly partitive crossing families. To name only one, the family of non-trivial minimizers of a symmetric submodular function is weakly partitive crossing. This has numerous consequences, since many families arise naturally from such functions [16], such as min-cuts and splits of a graph.

As mentioned in the previous paragraph, it is known that set families of certain properties admit tree representations. However, the bare existence of such representations does not imply their efficient computation. In the first part of this paper, we investigate exactly this question: when does a set family admit an efficient tree representation computation? We will give a sufficient condition for this question. We will identify two basic problems whose efficient solutions directly yield an efficient algorithm for the computation of a tree representation. We briefly describe the two basic problems, the *oracle* problem and the *separation* problem. The oracle problem decides whether a given set is a member of the given set family $\mathcal{F}$. Note that the oracle problem usually requires a non-trivial algorithm, since the given input represents the set family only implicitly. The separation problem, given two sets A and B, computes (if possible) a member M of the set family that satisfies $A \subset M \subset B$; we would say that M *separates* A and B. In many situations, e.g., the ones mentioned in the last paragraph, a solution for the separation problem exists using a polynomial number of calls to the oracle problem. The algorithm for computing the tree representation will work in three steps: first, it computes the intersection $\mathcal{T}$ of the maximal cross-free subfamilies of $\mathcal{F}$, second, it builds a tree representation for $\mathcal{T}$, and third, it labels the nodes and edges of the tree. A *cross-free* subfamily of $\mathcal{F}$ consists of members that pairwise do not cross. Since it is not possible to list all maximal cross-free subfamilies for obtaining an efficient algorithm, our algorithm for the first step will determine the cross-free kernel from a specific maximal cross-free subfamily. This algorithm part will rely heavily on the algorithms for the oracle and separation problem and the properties of weakly partitive crossing families. Independent of our application, this first step, sometimes also called "uncrossing", is a crucial pre-processing step in numerous algorithms (see, for instance, [16]). The tree representation for $\mathcal{T}$ is built by applying a fundamental result by Edmonds and Giles [11]. Our results unify and generalise several individual results, such as in [5,9,13,14].

In the second part of this paper, we will show an application of the results from the first part. We will introduce a new decomposition notion for directed graphs (*digraphs*). Usually, decomposition notions for undirected graphs naturally carry over to digraphs. However, the structure of digraphs is much more complex in comparison to undirected graphs, so that these decomposition notions may not have the comparable power for digraphs as their undirected analogue for undirected graphs. To give a short description, our decomposition combines properties of modules and splits. For a digraph G, we say that a set M of vertices of G is a *splitmodule* if the vertices in M have the same in-neighbours in the outside of M and vertices in M with out-neighbours in the outside of M have the same out-neighbours in the outside of M. We can say that M is a module with respect to in-neighbours and a split with respect to out-neighbours. Splitmodules generalise modules, as every module is a splitmodule. Therefore, splitmodules are less restrictive than modules and seem better prepared for coping with the rich and complex structure of digraphs. We show that the family of splitmodules of a strongly connected digraph forms a weakly partitive crossing family, and, therefore, a tree representation exists. As the main algorithmic results in this part, we will give efficient algorithms for the oracle and separation problem. By applying the results from the first part, we will obtain an $\mathcal{O}(n^3)$-time algorithm for computing a tree representation of the splitmodules of a strongly connected digraph.

Due to space restrictions, proofs may be omitted. We assume the reader is familiar with the basic graph-theoretic notions about undirected graphs and trees. Edges of an undirected graph are denoted as uv, meaning that the vertices u and v are adjacent.

2 Efficient tree representations of set families

A universe is a finite set, which we will usually denote as $\mathcal{U}$. A *set family* over a universe is a set of subsets of the universe. The elements of a set family are called *members*. The size of a set family can be exponential in the size of the universe, which means that a fixed concise representation cannot represent all possible set families. In this section, we consider set families of special properties and present an algorithmic framework for efficiently computing tree representations of such set families. A main result, Theorem 1, will imply a sufficient condition on a set family to admit a polynomial-time computation of the representation. To be more precise, we will identify two basic algorithmic tasks that will be used as subroutines for the computation algorithm. Efficient algorithms for the two subroutines will directly imply an efficient algorithm for the computation of the desired tree representation.

We begin by formally defining our general form of tree representation. Let T be a tree that has at least two nodes. The node set of T is denoted as $V(T)$, and the edge set of T is denoted as $E(T)$. A *leaf* of T is a node with exactly one neighbour in T; a node that is not a leaf is called an *inner node*. For an edge $e = uv$ of T, $T{-}e$ denotes the graph after deletion of e. Note that $T{-}e$ consists of exactly two connected components, which are also trees: the one that contains node u and the other that contains node v. For a node x of T, $(T{-}e)_x$ denotes the connected component of $T{-}e$ that contains node x. Assume that δ is a function that labels the leaves of T. By $\delta((T{-}e)_x)$, we denote the set of labels that are assigned to the nodes of $(T{-}e)_x$ that are leaves of T.

Definition 1. *Let $\mathcal{U}$ be a universe. Let $(T, \delta, \lambda, \kappa)$ be a quadruple where T is a tree with $|\mathcal{U}|$ leaves, δ is a bijective function from the leaves of T to the elements of $\mathcal{U}$, λ is a function such that for every edge uv of T, $\lambda(uv) \in \{(u,v),(v,u),\{u,v\}\}$, and κ assigns to every inner node of T a set family over $\mathcal{U}$. Let $\mathcal{F}$ be the union*

$$\Big\{\delta((T{-}uv)_v) : (u,v) \in V(T) \times V(T),\ uv \in E(T),\ \lambda(uv) \neq (v,u)\Big\} \ \cup \bigcup_{u \text{ inner node of } T} \kappa(u)\,.$$

We call $(T, \delta, \lambda, \kappa)$ a tree representation*, and $\mathcal{F}$ is called the set family* represented by $(T, \delta, \lambda, \kappa)$.

A set family $\mathcal{F}$ over a universe $\mathcal{U}$ is called *normalised* if $\emptyset \notin \mathcal{F}$ and $\mathcal{U} \notin \mathcal{F}$ and $\{u\} \in \mathcal{F}$ for every $u \in \mathcal{U}$. It is not difficult to see that every normalised set family has a tree representation, for instance, by using a star as a tree and assigning the set family to the centre node of the star by function κ. Note that this observation does not contradict the claims from the Introduction, since this star representation is not space-efficient. The size of a tree representation, i.e., the space required for the representation, is mainly determined by the size of the assignment function κ. Thus, the problem of obtaining a space-efficient representation is to "structure" the tree T in such a way that the assignment function κ admits a space-efficient representation.

It was shown that weakly partitive crossing set families admit tree representations where the assignment function κ assigns only a bounded number of different set types to the nodes of the representation tree [5]. Let $\mathcal{U}$ be a universe. Two sets A and B over $\mathcal{U}$ *cross* if the four sets $A \cap B$, $A \setminus B$, $B \setminus A$, $\mathcal{U} \setminus (A \cup B)$ are non-empty. It is important to observe that A and B cross if and only if A and $\mathcal{U} \setminus B$ cross. The crossing property is depicted in Figure 1. A set family is called *cross-free* if no pair of its members cross. Cross-free families admit easy tree set representations.

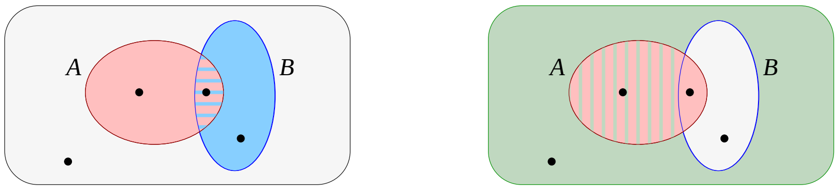

Figure 1. The left picture represents the situation when sets A and B cross. The right picture shows the situation for A and $\mathcal{U} \setminus B$, under the assumption that A and B cross.

Lemma 1 ([11]). *Let $\mathcal{F}$ be a normalised set family over a universe $\mathcal{U}$ of at least three elements. If $\mathcal{F}$ is cross-free then $\mathcal{F}$ has a unique tree representation $(T, \delta, \lambda, \kappa_0)$ such that T is a tree without vertices of degree 2 and assignment function κ_0 is empty.*

An easy but important corollary of Lemma 1 is that every cross-free subfamily of a set family over a universe of n elements can have at most $4n$ members. This observation will be important later in this section for the running time of our algorithms.

For computing the tree of the tree representation, we will employ a special cross-free family. For a set family $\mathcal{F}$ over a universe $\mathcal{U}$, the *canonical cross-free subfamily* of $\mathcal{F}$ is the set of members A of $\mathcal{F}$ for which there is no member B of $\mathcal{F}$ such that A and B cross. We show that the canonical cross-free subfamily is the unique maximal subfamily of the maximal cross-free subfamilies of $\mathcal{F}$.

Lemma 2. *Let $\mathcal{F}$ be a set family over a universe $\mathcal{U}$. The canonical cross-free subfamily of $\mathcal{F}$ is equal to the intersection of the maximal cross-free subfamilies of $\mathcal{F}$.*

Proof. Let $\mathcal{C}$ be the canonical cross-free subfamily of $\mathcal{F}$, and let $\mathcal{T}$ be the intersection of the maximal cross-free subfamilies of $\mathcal{F}$. Let $\mathcal{S}$ be a maximal cross-free subfamily of $\mathcal{F}$. We show that $\mathcal{C} \subseteq \mathcal{S}$. Let $A \in \mathcal{C}$. Due to the definition of $\mathcal{C}$, for all $B \in \mathcal{F}$, A and B do not cross, so that $\mathcal{S} \cup \{A\}$ is cross-free. The maximality of $\mathcal{S}$ implies that $A \in \mathcal{S}$. Due to the choice of $\mathcal{S}$, it follows that $\mathcal{C} \subseteq \mathcal{T}$. For the converse, let $D \in \mathcal{F}$ be such that $D \notin \mathcal{C}$. Due to the definition of $\mathcal{C}$, there is $B \in \mathcal{F}$ such that D and B cross. Since $\{B\}$ is a cross-free subfamily of $\mathcal{F}$, there is a maximal family $\mathcal{S}$ with $\{B\} \subseteq \mathcal{S} \subseteq \mathcal{F}$ and $\mathcal{S}$ is cross-free. Observe that $D \notin \mathcal{S}$. Thus, $D \notin \mathcal{T}$, which shows that $\mathcal{T} \subseteq \mathcal{C}$. □

Using the canonical cross-free subfamily, we compute the tree of the tree representation by applying Lemma 1. The labeling function λ represents the members of the canonical cross-free subfamily. It remains to represent the remaining members of the family. They will be represented through the assignment function κ. It turns out that κ has a very restricted structure, if the considered set family satisfies certain closure properties. A set family $\mathcal{F}$ over a universe $\mathcal{U}$ is called *weakly partitive crossing* if for every pair A, B of members of $\mathcal{F}$ that cross, the four sets $A \cup B$, $A \cap B$, $A \setminus B$, $B \setminus A$ are members of $\mathcal{F}$. By $\overline{A}$, we denote the complement of set A over $\mathcal{U}$, that is $\overline{A} =_{\text{def}} \mathcal{U} \setminus A$. For a partition $\mathcal{P} = \{P_1, \ldots, P_r\}$ of $\mathcal{U}$, let

$$f(\mathcal{F}, \mathcal{P}) =_{\text{def}} \left\{A \in \mathcal{F} : \text{ either } Z \subseteq A \text{ or } Z \subseteq \overline{A} \text{ for all } Z \in \mathcal{P}\right\}.$$

Clearly, $f(\mathcal{F}, \mathcal{P})$ is the set of members of $\mathcal{F}$ that are unions of partition classes of $\mathcal{P}$. Weakly partitive crossing families admit a concise description of the assignment function κ.

Lemma 3 ([5]). *Let $\mathcal{F}$ be a normalised weakly partitive crossing family over a universe $\mathcal{U}$. Let $\mathcal{C}$ be the canonical cross-free subfamily of $\mathcal{F}$, and let $(T, \delta, \lambda, \kappa_0)$ be the tree representation of $\mathcal{C}$ with T contains no nodes of degree 2 and κ_0 is empty. Let u be an inner node of T. Let $\mathcal{P}_u =_{\text{def}} \{\delta((T{-}ux)_x) : ux \in E(T)\}$. If $\mathcal{P}_u$ has more than four members, then one of the following cases holds:*

1) $f(\mathcal{F}, \mathcal{P}_u) = \left\{X, \overline{X} : X \in \mathcal{P}_u\right\}$ *or* $f(\mathcal{F}, \mathcal{P}_u) = \left\{\bigcup_{X \in \mathcal{A}} X : \emptyset \subset \mathcal{A} \subset \mathcal{P}_u\right\}$

2) there is an ordering $\langle P_1, \ldots, P_r\rangle$ of the members of $\mathcal{P}_u$ such that one of the three cases holds:

a) $f(\mathcal{F}, \mathcal{P}_u) = \left\{P_i \cup \cdots \cup P_j : 1 \leq i \leq j \leq r\right\}$

b) $f(\mathcal{F}, \mathcal{P}_u) = \left\{P_i \cup \cdots \cup P_j, \overline{P_i \cup \cdots \cup P_j} : 1 \leq i \leq j \leq r\right\}$

c) $\left\{P_i \cup \cdots \cup P_j : 2 \leq i < j \leq r \text{ and } j - i < r - 2\right\} \subseteq f(\mathcal{F}, \mathcal{P}_u)$ *and* $X \cap P_1 = \emptyset$ *for all* $X \in f(\mathcal{F}, \mathcal{P}_u)$

3) there is $Y \in \mathcal{P}_u$, and let $\mathcal{P}'_u =_{\text{def}} \mathcal{P}_u \setminus \{Y\}$, such that $X \cap Y = \emptyset$ for all $X \in f(\mathcal{F}, \mathcal{P}_u)$ and $\left\{\bigcup_{X \in \mathcal{A}} X : \emptyset \subset \mathcal{A} \subset \mathcal{P}'_u\right\} \setminus \left\{X : X \in \mathcal{P}'_u\right\} \subseteq f(\mathcal{F}, \mathcal{P}_u)$.

Let $\kappa(u) =_{\text{def}} f(\mathcal{F}, \mathcal{P}_u)$ for every inner node u of T. Then, $(T, \delta, \lambda, \kappa)$ is a tree representation for $\mathcal{F}$. This tree representation is unique.

As a corollary of Lemma 3, we can present our algorithm for computing the tree representation of a normalised weakly partitive crossing family. The algorithm works in four steps. Let $\mathcal{F}$ be a set family over a universe $\mathcal{U}$. Then,

step (1)	compute a maximal cross-free subfamily $\mathcal{T}$ of $\mathcal{F}$
step (2)	compute the canonical cross-free subfamily $\mathcal{C}$ of $\mathcal{F}$ from $\mathcal{T}$
step (3)	compute the tree representation of $\mathcal{C}$ according to Lemma 1
step (4)	assign the remaining family members to the inner nodes of the tree according to Lemma 3, which defines assignment function κ.

In the remaining part of this section, we will show that each of the four steps can be solved efficiently, assuming efficient algorithms for two basic problems. These two problems are the following. Let $\mathcal{U}$ be a universe, and let $\mathcal{F}$ be a set family over $\mathcal{U}$.

ORACLE$_{\mathcal{F}}$
Input $A \subseteq \mathcal{U}$
Question Is A a member of $\mathcal{F}$?

Algorithm 1 UNCROSSING

```
INPUT:  set family $\mathcal{F}$ over a universe $\mathcal{U}$, partition $\mathcal{P}$ of universe $\mathcal{U}$
 1: if $\mathcal{P}$ has at most three partition classes then
 2:     $\mathcal{T} \leftarrow f(\mathcal{F},\mathcal{P})$
 3: else
 4:     pick any $X \in \mathcal{P}$ and $Y \in \mathcal{P}$ such that $X \neq Y$
 5:     $A \leftarrow$ SEPARATION$_{\mathcal{F}}(\mathcal{P}, X, Y)$
 6:     if $A \neq \emptyset$ then
 7:         if $\overline{A} \in \mathcal{F}$ then $\mathcal{T} \leftarrow \{A, \overline{A}\}$ else $\mathcal{T} \leftarrow \{A\}$ end if
 8:         $\mathcal{T} \leftarrow \mathcal{T} \cup$ UNCROSSING$(\mathcal{F}, \{Z \in \mathcal{P} : Z \subseteq A\} \cup \{\overline{A}\})$
 9:         $\mathcal{T} \leftarrow \mathcal{T} \cup$ UNCROSSING$(\mathcal{F}, \{Z \in \mathcal{P} : Z \subseteq \overline{A}\} \cup \{A\})$
10:     else
11:         $\mathcal{T} \leftarrow \emptyset$
12:         if $X \cup Y \in \mathcal{F}$ then $\mathcal{T} \leftarrow \mathcal{T} \cup \{X \cup Y\}$ end if
13:         if $\overline{X \cup Y} \in \mathcal{F}$ then $\mathcal{T} \leftarrow \mathcal{T} \cup \{\overline{X \cup Y}\}$ end if
14:         $\mathcal{T} \leftarrow \mathcal{T} \cup$ UNCROSSING$(\mathcal{F}, (\mathcal{P} \setminus \{X, Y\}) \cup \{X \cup Y\})$
15:     end if
16: end if
OUTPUT: $\mathcal{T}$
```

SEPARATION$_{\mathcal{F}}$
Input partition $\mathcal{P}$ of $\mathcal{U}$ and $X, Y \in \mathcal{P}$
Output $A \in f(\mathcal{F},\mathcal{P})$ such that $X \subset A \subset \overline{Y}$; $\emptyset$, if no such A exists.

Employing algorithms for the two problems ORACLE$_{\mathcal{F}}$ and SEPARATION$_{\mathcal{F}}$ as subroutines, we can compute a maximal cross-free subfamily of $\mathcal{F}$. The algorithm for computing such a subfamily is given as Algorithm 1, called UNCROSSING. It takes as input a partition of $\mathcal{U}$, and in each recursive step, a new element for the cross-free subfamily is found and the partition is made coarser. Note that this directly implies that the number of recursive calls of UNCROSSING is linear in $|\mathcal{U}|$. The oracle problem is denoted as a usual membership test of the form: $X \in \mathcal{F}$. The following lemma shows the main property about the result computed by UNCROSSING. Note that $\mathcal{F}$ can be an arbitrary set family.

Lemma 4. *Let $\mathcal{F}$ be a set family over a universe $\mathcal{U}$ and let $\mathcal{P}$ be a partition of $\mathcal{U}$. Let $\mathcal{R}$ be the output of* UNCROSSING$(\mathcal{F},\mathcal{P})$. *Then, $\mathcal{R} \cup \{\mathcal{U}, \emptyset\} \cup \{Z, \overline{Z} : Z \in \mathcal{P}\}$ is a maximal cross-free subfamily of $f(\mathcal{F},\mathcal{P}) \cup \{\mathcal{U}, \emptyset\} \cup \{Z, \overline{Z} : Z \in \mathcal{P}\}$.*

As a consequence of Lemma 4, we can efficiently compute a maximal cross-free subfamily of $\mathcal{F}$: UNCROSSING$(\mathcal{F}, \{\{u\} : u \in \mathcal{U}\})$ outputs the non-trivial part of a maximal cross-free subfamily $\mathcal{T}$ of $\mathcal{F}$. We briefly discuss the number of calls to SEPARATION$_{\mathcal{F}}$. This number is equal to the number of recursive calls to UNCROSSING. Observe that each call increases the size of $\mathcal{T}$ by at least one member (the assignment in line 7) or decreases the size of the involved partition by 1 (the new partition in line 14). Therefore, for an n-element universe, this makes a total of at most $4n + n$ calls to UNCROSSING.

We use the output cross-free family $\mathcal{T}$ to compute the canonical cross-free subfamily of $\mathcal{F}$. The algorithm starts from a tree representation of $\mathcal{T}$ and explore its properties. The algorithm, which cannot be presented here, due to the space restrictions, is strongly dependent on the properties of normalised weakly partitive crossing families. The running-time result is stated in the next lemma.

Lemma 5. *Let $\mathcal{F}$ be a normalised weakly partitive crossing family over a universe $\mathcal{U}$. Let $\mathcal{T}$ an arbitrary maximal cross-free subfamily of $\mathcal{F}$ be given. Then, the canonical cross-free subfamily $\mathcal{C}$ of $\mathcal{F}$ can be computed from $\mathcal{T}$ in linear time with $\mathcal{O}(|\mathcal{U}|)$ calls to* $\textsc{Oracle}_{\mathcal{F}}$.

To complete our algorithm, it remains to determine assignment function κ. The result of Lemma 3 shows that κ does not require an explicit representation of the assigned subfamilies but can be efficiently represented by simply storing the information about which of the cases in the statement applies. The computational problem to resolve is to decide the actual case for each node of the tree. Given the maximal cross-free subfamily of $\mathcal{F}$ and the canonical cross-free subfamily, κ can be computed efficiently.

Lemma 6. *Let $\mathcal{F}$ be a normalised weakly partitive crossing family over a universe $\mathcal{U}$. Assume that the canonical cross-free subfamily $\mathcal{C}$ of $\mathcal{F}$ and a maximal cross-free subfamily $\mathcal{T}$ of $\mathcal{F}$ are given. Then, the tree representation of $\mathcal{F}$ can be computed in linear time by making $\mathcal{O}(|\mathcal{U}|)$ calls to* $\textsc{Oracle}_{\mathcal{F}}$. *The space required for the representation is linear in $|\mathcal{U}|$.*

Combining Lemma 4, Lemma 5 and Lemma 6, we obtain

Theorem 1. *For a normalised weakly partitive crossing family $\mathcal{F}$ over a universe $\mathcal{U}$, the (unique) tree representation of $\mathcal{F}$ can be computed in linear time by making $\mathcal{O}(|\mathcal{U}|)$ calls to* $\textsc{Oracle}_{\mathcal{F}}$ *and* $\textsc{Separation}_{\mathcal{F}}$. *The space required for the representation is linear in $|\mathcal{U}|$.*

3 A module-split digraph decomposition

We will introduce a digraph decomposition notion, which bridges the gap between modular and split decomposition for digraphs. We will show that the decomposition satisfies the closure properties of weakly partitive crossing families, so that the algorithmic results from Section 2 are applicable. We will also show that the two basic problems, Oracle and Separation, are efficiently solvable, so that our main theorem from Section 2 directly implies an efficient algorithm for computing a tree representation.

We consider only simple finite digraphs. For a digraph G, the vertex set of G is denoted as $V(G)$, and the arc set of G is denoted as $A(G)$. For an arc (u, v) of G, we say that u is an *in-neighbour* of v and v is an *out-neighbour* of u. For a vertex v of G, the *in-neighbourhood* of v is $N_G^{\mathrm{in}}(v) =_{\mathrm{def}} \{u : (u, v) \in A(G)\}$, and the *out-neighbourhood* of v is $N_G^{\mathrm{out}}(v) =_{\mathrm{def}} \{u : (v, u) \in A(G)\}$. For two vertices u, v of G, a *directed path* from u to v is a sequence $u = x_1, x_2, \ldots, x_k = v$ of vertices of G such that $(x_i, x_{i+1}) \in A(G)$ for $1 \leq i < k$. A digraph is *strongly connected* if there is a directed path from every vertex to every other vertex.

3.1 Splitmodules of digraphs

We say that two vertices u and v of a digraph are not distinguishable by a third vertex w if w is an in-neighbour of u and v or if w is not an in-neighbour of u and v. "Being indistinguishable" is a fundamental property that is explored in graph theory as well as by graph algorithms. We study variants of this property.

Definition 2. *Let G be a digraph. A set M of vertices of G satisfies:*

a) the module condition *if $N_G^{\text{in}}(u) \setminus M = N_G^{\text{in}}(v) \setminus M$ for every vertex pair u, v from M;*
b) the split condition *if $N_G^{\text{out}}(u) \setminus M = N_G^{\text{out}}(v) \setminus M$ for every vertex pair u, v from M where $N_G^{\text{out}}(u) \setminus M \neq \emptyset$ and $N_G^{\text{out}}(v) \setminus M \neq \emptyset$.*

Let M be a set of vertices of G. If M satisfies the module condition then M is called a genuine module *of G. If M satisfies the split condition then M is called a* genuine split *of G. If M satisfies the module and split condition then M is called a* splitmodule *of G.*

A set family is *crossing* if it is closed under union and intersection of its crossing members. It is known that the genuine modules of a digraph form a crossing family [4]. A similar result holds for genuine splits and splitmodules.

Lemma 7. *Let G be a stongly connected digraph. The genuine splits of G form a crossing family, and the splitmodules of G form a weakly partitive crossing family.*

We show that splitmodules admit a local characterisation property. For a digraph G and $X \subseteq V(G)$, the *subgraph of G induced by X*, denoted as $G[X]$, is the digraph on vertex set X and with arc set $A(G) \cap X^2$.

Lemma 8. *Let G be a digraph. For a set M of vertices of G and $a \in M$ and $c \notin M$ where $(a, c) \in A(G)$, M is a splitmodule of G if and only if there is no ordered vertex pair (b, d) of G such that $b \in M$ and $d \notin M$ and $\{a, b\}$ is not a splitmodule of $G[\{a, b, c, d\}]$.*

Proof. Let M and a and c be as assumed. We have to show two implications. First, assume that M is a splitmodule of G. Let H be an induced subgraph of G. Then, $M \cap V(H)$ is a splitmodule of H, and thus, $\{a, b\}$ is a splitmodule of $G[\{a, b, c, d\}]$ for every choice of $b \in M$ and $d \notin M$. For the converse, assume that M is no splitmodule of G. Then, M does not satisfy the module or the split condition. First assume that M does not satisfy the module condition. This means that there is a vertex triple u, v, w of G such that $u, v \in M$ and $w \notin M$ and $(w, u) \in A(G)$ and $(w, v) \notin A(G)$. If $a = u$ or $a = v$ then $\{u, v\}$ is not a splitmodule of $G[\{u, v, c, w\}]$, and the claim follows. Otherwise, if $a \notin \{u, v\}$ then either $(w, a) \in A(G)$ and $\{a, v\}$ is not a splitmodule of $G[\{a, v, c, w\}]$ or $(w, a) \notin A(G)$ and $\{a, u\}$ is not a splitmodule of $G[\{a, u, c, w\}]$. Note that we do not exclude the possibility of $c = w$. Second, assume that M does not satisfy the split condition. Then, there are four vertices u, v, w, x of G such that $u, v \in M$ and $w, x \notin M$ and $(u, w) \in A(G)$ and $(v, w) \notin A(G)$ and $(v, x) \in A(G)$. Analogously to the module condition case, if $a \in \{u, v\}$, then $\{u, v\}$ is not a splitmodule of $G[\{u, v, c, w\}]$. Let $a \notin \{u, v\}$. If $(a, w) \notin A(G)$ then $\{a, u\}$ is not a splitmodule of $G[\{a, u, c, w\}]$. If $(v, c) \notin A(G)$ then $\{a, v\}$ is not a splitmodule of $G[\{a, v, c, x\}]$. If $(v, c) \in A(G)$ then $\{a, v\}$ is not a splitmodule in $G[\{a, v, c, w\}]$. □

The proof of Lemma 8 actually shows a local characterisation property for genuine modules and genuine splits, which combine into the local characterisation property for splitmodules.

3.2 A tree representation for splitmodules

We present an efficient algorithm for computing a tree set representation for the splitmodules of a strongly connected digraph. We apply the results of Section 2, which is possible, since the splitmodules of a strongly connected digraph form a weakly partitive crossing family, due to Lemma 7. According to Theorem 1, it remains to specify algorithms for the oracle and separation problem.

We begin with the separation problem. Our procedure is given as Algorithm 2. It receives as input a digraph G, a partition of its vertex set and two vertices. To give a brief description, the algorithm tries to compute a union of partition classes of the given partition such that this union is properly between classes X and $\overline{Y}$ (selected in line 1). The proper inclusion is partly ensured by the chosen partition class Z (line 2) and partly by the checks in line 7 and line 12. Note here that Z is chosen without any restriction. Also note that the algorithm always terminates, since the two **while** loops, in lines 4–5 and 9–10, can be executed at most $k - 2$ times, where k is the number of partition classes of the partition. We show that, under a certain condition on the selection of the input vertices, Algorithm 2 implements SEPARATION.

Lemma 9. *Let G be a digraph. Let $\mathcal{F}$ be the family of splitmodules of G and let $\mathcal{P}$ be a partition of $V(G)$ with at least three partition classes. Let $X, Y \in \mathcal{P}$ be such that $X \neq Y$ and $(a, c) \in A(G)$ for some $a \in X$ and $c \in Y$. Let A be the output of Algorithm 2 on input $(G, \mathcal{P}, a, c)$. Then, $X \subset A \subset \overline{Y}$ and $A \in f(\mathcal{F}, \mathcal{P})$ if and only if there exists $E \in f(\mathcal{F}, \mathcal{P})$ such that $X \subset E \subset \overline{Y}$.*

Lemma 10. *There is a linear-time algorithm, given a digraph G and a set M of vertices of G, to decide whether M is a splitmodule of G.*

Combining Lemma 9, Lemma 10 and Theorem 1, we obtain

Theorem 2. *The (unique) tree representation of the family of splitmodules of a strongly connected digraph can be computed in $\mathcal{O}(n^3)$ time, where n is the number of vertices of the input digraph.*

Algorithm 2 (Separation)

```
INPUT:   partition $\mathcal{P}$ of $V(G)$ of a digraph $G$ and $a, c \in V(G)$
 1: let $X, Y \in \mathcal{P}$ with $a \in X$ and $c \in Y$
 2: pick $Z \in \mathcal{P}$ such that $Z \neq X$ and $Z \neq Y$
 3: $A \leftarrow X \cup Z$
 4: while exist $b \in A$ and $d \notin A$ with $\{a, b\}$ is not a splitmodule of $G[\{a, b, c, d\}]$ do
 5:     $A \leftarrow A \cup D$ for $D \in \mathcal{P}$ with $d \in D$
 6: end while
 7: if $A \subset \overline{Y}$ then output $A$ end if
 8: $A \leftarrow \overline{Y \cup Z}$
 9: while exist $b \in A$ and $d \notin A$ with $\{a, b\}$ is not a splitmodule of $G[\{a, b, c, d\}]$ do
10:     $A \leftarrow A \setminus B$ for $B \in \mathcal{P}$ with $b \in B$
11: end while
12: if $X \subset A$ then output $A$ end if
13: output $\emptyset$
```

4 Conclusion

We introduced the notion of splitmodule for digraphs. The splitmodules of a strongly connected digraph form a weakly partitive crossing family, a set family that admits efficient tree representation. We gave an $\mathcal{O}(n^3)$-time algorithm for computing this representation. The algorithm is specific for splitmodules only in two respects: splitmodules admit a local characterisation property (Lemma 8), and, given a digraph G and $M \subseteq V(G)$, it can be checked in linear time whether M is a splitmodule of G (Lemma 10). It directly follows for an arbitrary weakly partitive crossing family that a similar tree representation algorithm exists, if these two easy conditions can be satisfied. The second task, the oracle problem, is basic. The more challenging problem may be the local characterisation property. The running time of Algorithm 2 was mainly determined by the structure of the local characterisation property, which required consideration of almost all vertex pairs (lines 4 and 9). Thus, our approach provides efficient tree representation algorithms for a large class of decomposition notions.

A first question is whether our algorithm can be improved to $\mathcal{O}(nm)$ or even $\mathcal{O}(n^2)$ running time for the case of splitmodules of strongly connected digraphs. Another question is whether a similar approach also works for generalisations of weakly partitive crossing families, such as crossing families. It is known that crossing families admit $\mathcal{O}(|\mathcal{U}|^2)$-space representations [1,12]. The main difficulty is to determine a concise representation of assignment function κ.

Acknowledgement

We would like to thank Ross McConnell for fruitful initial discussions on the topic and for comments on the final version of the paper.

References

1. A. Bernáth. A note on the directed source location algorithm. Technical report, TR-2004-12, Egerváry Research Group, Budapest, 2004.
2. B.-M. Bui-Xuan. *Tree-representation of set families in graph decompositions and efficient algorithms.* PhD thesis, University of Montpellier II, 2008.
3. B.-M. Bui-Xuan and M. Habib. A representation theorem for union-difference families and application. *LATIN 2008*, Springer LNCS, 4957:492–503, 2008.
4. B.-M. Bui-Xuan, M. Habib, V. Limouzy, F. de Montgolfier. Algorithmic Aspects of a General Modular Decomposition Theory. *Discrete Applied Mathematics*, 157:1993–2009, 2009.
5. B.-M. Bui-Xuan, M. Habib, M. Rao. Tree-representation of set families and applications to combinatorial decompositions. *European Journal of Combinatorics*, to appear.
6. M. Chein, M. Habib, M.-C. Maurer. Partitive hypergraphs. *Discrete Mathematics*, 37:35–50, 1981.
7. W. Cunningham. *A combinatorial decomposition theory.* PhD thesis, University of Waterloo, 1973.
8. W. Cunningham and J. Edmonds. A combinatorial decomposition theory. *Canadian Journal of Mathematics*, 32:734–765, 1980.
9. W. Cunningham. Decomposition of directed graphs. *SIAM Journal on Algebraic and Discrete Methods*, 2:214–228, 1982.
10. E. Dinitz, A. Karzanov, M. Lomonosov. On the structure of a family of minimal weighted cuts in a graph. In: *Studies in Discrete Optimization*, A. Pridman (Ed.), pp. 290–306, Nauka, Moscow, 1976.
11. J. Edmonds and R. Giles. A min-max relation for submodular functions on graphs. *Annals of Discrete Mathematics*, 1:185–204, 1977.
12. H. Gabow. Centroids, Representations, and Submoduar Flows. *Journal of Algorithms*, 18:586–628, 1995.
13. W.-L. Hsu, C. Gabor, K. Supowit. Recognizing circle graphs in polynomial time. *Journal of the ACM*, 36:435–473, 1989.
14. F. de Mongolfier and M. Rao. The bi-join decomposition. *Electronic Notes in Discrete Mathematics*, 22:173–177, 2005.
15. M. Queyranne. Minimizing symmetric submodular functions. *Mathematical Programming*, 82:3–12, 1998.
16. A. Schrijver. *Combinatorial Optimization – Polyhedra and Efficiency.* Springer, 2003.