A General Algorithmic Scheme for Modular Decompositions of Hypergraphs and Applications

Michel Habib[1,3](✉), Fabien de Montgolfier[1,3], Lalla Mouatadid[2], and Mengchuan Zou[1,3]

[1] IRIF, UMR 8243 CNRS & Université de Paris, Paris, France
habib@irif.fr
[2] Department of Computer Science, University of Toronto, Toronto, ON, Canada
[3] Gang Project, Inria, Paris, France

Abstract. We study here algorithmic aspects of modular decomposition of hypergraphs. In the literature one can find three different definitions of modules, namely: the standard one [19], the k-subset modules [6] and the Courcelle's one [11]. Using the fundamental tools defined for combinatorial decompositions such as partitive and orthogonal families, we directly derive a linear time algorithm for Courcelle's decomposition. Then we introduce a general algorithmic tool for partitive families and apply it for the other two definitions of modules to derive polynomial algorithms. For standard modules it leads to an algorithm in $O(n^3 \cdot l)$ time (where n is the number of vertices and l is the sum of the size of the edges). For k-subset modules we obtain $O(n^3 \cdot m \cdot l)$ (where m is the number of edges). This is an improvement from the best known algorithms for k-subset modular decomposition, which was not polynomial w.r.t. n and m, and is in $O(n^{3k-5})$ time [6] where k denotes the maximal size of an edge. Finally we focus on applications of orthogonality to modular decompositions of tournaments, simplifying the algorithm from [18]. The question of designing a linear time algorithms for the standard modular decomposition of hypergraphs remains open.

1 Introduction

In this paper we study hypergraph modular decomposition; an important generalization of graph modular decomposition. Hypergraph modular decomposition is equivalent to the modular decomposition of both set systems [19] as well as monotone Boolean functions [20], while that of general Boolean functions was shown to be NP-hard [5]. We study here algorithmic aspects of modular decomposition of hypergraphs. In the literature one can find three different definitions of modules, namely: the standard one [19], the k-subset modules [6] and Courcelle's one [11]. In the following we recall the fundamental tools defined for combinatorial decompositions such as partitive and orthogonal families. This directly yields a linear time algorithm for Courcelle's decomposition.

C. J. Colbourn et al. (Eds.): IWOCA 2019, LNCS 11638, pp. 251–264, 2019.
https://doi.org/10.1007/978-3-030-25005-8_21

In Sect. 3 we propose a general algorithmic tool for partitive families and apply it for the other two definitions of modules to derive polynomial algorithms. For standard modules it leads to an algorithm in $O(n^3 \cdot l)$ time (where n is the number of vertices and l the sum of the size of the m edges). For k-subset modules we obtain $O(n^3 \cdot m \cdot l)$ algorithm, improving the previous known $O(n^{3k-5})$ time algorithm [6]. In Sect. 5 we show that the orthogonality may also bring some new insights to graph modular decomposition, i.e. application to factorizing permutations and simplify decomposition algorithm for tournaments.

1.1 Definitions

Following Berge's definition of hypergraphs [1], a hypergraph H over a finite ground set $V(H)$ is made by a family of subsets of $V(H)$, denoted by $\mathcal{E}(H)$ such that (i) $\forall e \in \mathcal{E}(H)$, $e \neq \emptyset$ and (ii) $\cup_{e \in \mathcal{E}(H)} e = V(H)$. In other words, a hypergraph admits no empty edge and no isolated vertex. Furthermore we deal only with ***simple*** hypergraphs, where $\mathcal{E}(H) \subseteq 2^{V(H)}$ (no multiple edges). When analyzing algorithms, we use the standard notations: $|V(H)| = n$, $|\mathcal{E}(H)| = m$ and $l = \Sigma_{e \in \mathcal{E}(H)} |e|$. For every edge $e \in \mathcal{E}(H)$, we denote by $H(e) = \{x \in V(H)$ such that $x \in e\}$, and for every vertex $x \in V(H)$, we denote by $N(x) = \{e \in \mathcal{E}(H)$ such that $x \in H(e)\}$. To each hypergraph one can associate a bipartite graph G, namely its ***incidence bipartite graph***, such that: $V(G) = V(H) \cup \mathcal{E}(H)$ and $E(G) = \{xe$ with $x \in V(H)$ and $e \in \mathcal{E}(H)$ such that $x \in H(e)\}$. For a hypergraph H and a subset $M \subseteq V(H)$, let $H(M)$ denote the ***hypergraph induced by*** M, where $V(H(M)) = M$ and $\mathcal{E}_{H(M)} = \{e \cap M \in \mathcal{E}(H)$, for $e \cap M \neq \emptyset\}$. Similarly, let H_M denote the ***reduced hypergraph*** where $V(H_M) = (V \setminus M) \cup \{m\})$ with $m \notin V$, and $\mathcal{E}(H_M) = \{e \in \mathcal{E}(H)$ with $e \cap M = \emptyset\} \cup \{(e \setminus M) \cup \{m\}$ with $e \in \mathcal{E}(H)$ and $e \cap M \neq \emptyset\}$. By convention in case of multiple occurrences of a similar edge, only one edge is kept and so H_M is a simple hypergraph.

Two non-empty sets A and B ***overlap*** if $A \cap B \neq \emptyset$, $A \setminus B \neq \emptyset$, and $B \setminus A \neq \emptyset$. Sets that do not overlap are said to be ***orthogonal***, which is denoted by $A \perp B$. Let $\mathcal{F}$ be a family of subsets of a ground set V. We denote by $\mathcal{F}^{\perp}$ the family of subsets of V which are orthogonal to ***every*** element of $\mathcal{F}$. A set $S \in \mathcal{F}$ is called ***strong*** if $\forall S' \neq S \in \mathcal{F} : S \perp S'$. Let Δ denote the symmetric difference operation.

Definition 1 [10]**.** *A family of subsets $\mathcal{F}$ over a ground set V is* ***partitive*** *if it satisfies the following properties:*

(i) $\emptyset$, V and all singletons $\{x\}$ for $x \in V$ belong to $\mathcal{F}$.
(ii) $\forall A, B \in \mathcal{F}$ that overlap, $A \cap B, A \cup B, A \setminus B$ and $A \Delta B \in \mathcal{F}$

Both orthogonal and partitive families play fundamental roles in combinatorial decompositions [10,17]. Every partitive family admits a unique decomposition tree, with only two types of nodes: ***complete*** and ***prime***. As for graphs, a node in a decomposition tree is said to be complete if the subgraph rooted at that node is either a clique or an independent set, and said to prime if the subgraph rooted at that node cannot be decomposed any further. It is well known that the strong elements of $\mathcal{F}$ form a tree ordered by the inclusion relation [10]. In this decomposition tree, every node corresponds to a set of the elements of the ground set V of $\mathcal{F}$, and the leaves of the tree are single elements of V. For a ***complete*** (resp. ***prime***) node, every union of its child nodes (resp. no union of its child nodes other than itself) belongs to the partitive family.

Here we introduce some properties on orthogonality that will be useful for Courcelle's module and applications on graphs.

Property 1 [16]**.** Given a family $\mathcal{F}$ of subsets over a ground set V, $\mathcal{F}^{\perp}$ is partitive. Furthermore $\mathcal{F}$ is partitive iff $\mathcal{F} = (\mathcal{F}^{\perp})^{\perp}$.

Moreover, if $\mathcal{F}$ and $\mathcal{F}'$ are two partitive families on the same ground set V, then $\mathcal{F} \cap \mathcal{F}'$ is also partitive and thus we can search for the smallest partitive family that contains a given family.

Definition 2. *Let $\mathcal{F}$ be a family of subsets over a ground set V. Let $\mathcal{P}(\mathcal{F})$ denote the smallest – by inclusion – completion of $\mathcal{F}$ that admits a unique tree decomposition with nodes labeled prime and complete.*

Property 2. For every subset family $\mathcal{F}$, $\mathcal{P}(\mathcal{F}) = (\mathcal{F}^{\perp})^{\perp}$.

We denote by $\mathcal{O}(\mathcal{F})$ the ***overlap graph*** of $\mathcal{F}$ constructed as follows: The vertices are the elements of $\mathcal{F}$, and two vertices are adjacent if their corresponding subsets overlap. A pair of vertices is said to be ***twins*** if the vertices appear in exactly the same members of $\mathcal{F}$. The ***block of twins*** are the equivalence classes of the twin relation. Putting together Theorems 3.3 and 5.1 of [16] we get:

Theorem 1. *Let $\mathcal{F}$ be a subset family on V. A subset N is a node of the decomposition tree of $\mathcal{F}^{\perp}$ if and only if N is either:*

1. V (the ground set), or $\{v\}$ (a one-subset element), or a block of twins, or
2. $\cup\mathcal{C}$ for some connected component $\mathcal{C}$ of the overlap graph of $\mathcal{O}(\mathcal{F})$

An internal node N is labeled prime if there exists a component $\mathcal{C}$ of $\mathcal{O}(\mathcal{F})$ with at least two members of $\mathcal{F}$ such that $N = \cup\mathcal{C}$. Otherwise N is labeled complete.

Non-trivial nodes are mostly given by Case 2, since in Case 1 we either get the root of the tree, or a leaf, and all the siblings of any block-of-twins node are leaves. In [16] the following theorem is proposed, using as a blackbox Dahlhaus's algorithm [12] plus some post-treatment. But it has been largely simplified by [9].

Theorem 2. *Let $\mathcal{F}$ be a subset family on V. The decomposition tree of $\mathcal{F}^{\perp}$ can be computed in $O(n + l)$ time.*

Corollary 1. *$\mathcal{P}(\mathcal{F})$ can be computed in $O(n + l)$ time.*

2 Hypergraph Modular Decomposition

Hypergraph Substitution: ***Substitution*** in general is the action of replacing a vertex v in a graph G by a graph $H(V', E')$ while preserving the same neighborhood properties. To apply this concept to hypergraphs, we use the following definition presented in [19,20]:

Definition 3. *Given two hypergraphs H, H_1, and a vertex $v \in V(H)$ the* ***substitution*** *of vertex $v \in V(H)$ by hypergraph H_1 is an hypergraph, denoted $H' = H_v^{H_1}$, which satisfies $V(H') = \{V \setminus v\} \cup V(H_1)$, and $\mathcal{E}(H') = \{e \in \mathcal{E}(H)$ s.t. $v \notin e\} \cup \{f \setminus v \cup e_1$ s.t. $f \in \mathcal{E}(H)$ and $v \in f$ and $e_1 \in \mathcal{E}(H_1)\}$.*

Let us consider the example in Fig. 1 where hypergraphs are described using their incidence matrices. In this example, we substitute vertex v_3 in H by the hypergraph H_1 to create H'. Note that even if H, H_1 are undirected graphs, the substitution operation may create edges of size 3, and therefore the resulting hypergraph H' is no longer a graph.

Definition 4 (Standard Hypergraph Module) [19,20]**.** *Given a hypergraph H, a* ***module*** *$M \subseteq V(H)$ satisfies: $\forall A, B \in \mathcal{E}(H)$ s.t. $A \cap M \neq \emptyset$, $B \cap M \neq \emptyset$ then $(A \setminus M) \cup (B \cap M) \in \mathcal{E}(H)$.*

When M is a module of H then $H = (H_M)_m^{H(M)}$. On the previous example: let us take $A, B \in \mathcal{E}$ (resp. 1^{st} and 6^{th} columns of H') then $(A \setminus M) \cup B \cap M = 2^{nd}$ column of H' and therefore belongs to $\mathcal{E}$. If M is a module of H then $\forall e \in \mathcal{E}(H_M)$, the edges of H that strictly contain e and are not included in M are the same. In other words, ***all edges in $\mathcal{E}(H_M)$ behave the same with respect to the outside***, which is an equivalence relation between edges.

Property 3 [20]. The family of modules of a simple hypergraph H is partitive.

Since every partitive family has a unique decomposition tree [10], it follows that the family of the modules of a simple hypergraph admits a uniqueness decomposition theorem and a unique hypermodular decomposition tree. For hypergraphs, as for graphs we have 2 types of complete node, namely series and parallel. Therefore the modular decomposition tree for hypergraphs has three types of nodes: series, parallel and prime. If $\mathcal{E}(H)$ is the set of all singletons of $V(H)$, then every subset of $V(H)$ is a module; this corresponds to the parallel case. On the other hand if $\mathcal{E}(H) = 2^{|V(H)|}$, then also every subset of $V(H)$ is a module, which corresponds to the series case.

Modular decomposition, applied to bipartite graphs, just leads to the computation of sets of ***false twins*** in the bipartite graphs (vertices sharing the same neighborhood) and connected components. As Fig. 1 example shows, hypergraph modules are not always set of twins of the associated incidence bipartite.

Some authors [3,4] defined **clutters** hypergraphs, in which no edge is included into another one. In this case, clutters modules are called **committees** [3]. Trivial clutters are closed under hypergraph substitution. The committees of a simple clutter also yields a partitive family which implies a uniqueness

$$\begin{array}{c|c|c|c|}
\text{a} & 0 & 1 & 1 \\
\text{b} & 0 & 1 & 1 \\
\text{c} & 1 & 0 & 1
\end{array}$$

H_1

$$\begin{array}{c|c|c|c|c|}
v_1 & 1 & 0 & 1 & 1 \\
v_2 & 1 & 0 & 0 & 0 \\
v_3 & 1 & 0 & 1 & 0 \\
v_4 & 1 & 1 & 0 & 0
\end{array}$$

H

$$\begin{array}{c|c|c|c|c|c|c|c|c|}
v_1\ (u_1) & 1 & 1 & 1 & 0 & 1 & 1 & 1 & 1 \\
v_2\ (u_2) & 1 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\
\text{a}\ \ (u_3) & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
\text{b}\ \ (u_4) & 0 & 1 & 1 & 0 & 0 & 1 & 1 & 0 \\
\text{c}\ \ (u_5) & 1 & 0 & 1 & 0 & 1 & 0 & 1 & 0 \\
v_4\ (u_6) & 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0
\end{array}$$

$H' = H_{v_3}^{H_1}$

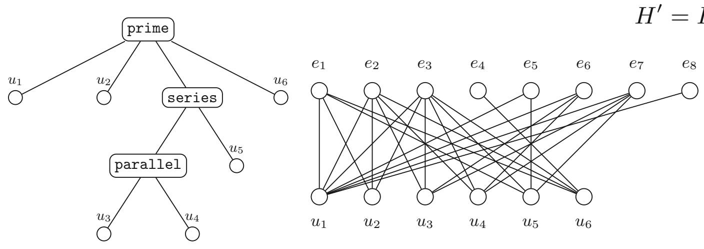

Fig. 1. An exp. of substitution, its decomposition tree, and its incidence bipartite graph. $\{u_3, u_4, u_5\}$ is a module, but only u_3, u_4 are false twins in the incidence bipartite.

decomposition theorem. From this one can recover a well-known Shapley's theorem on the modular decomposition of monotone Boolean functions. It should be noted however that finding the modular decomposition of a Boolean function is NP-hard [5]. It was shown in [7], that computing clutters in linear time would contradict the SETH conjecture.

2.1 Variants of Modular Decomposition of Hypergraphs

Often when generalizing graph concepts to hypergraphs there are several potential generalizations. In fact we found in the literature two variations on the hypergraph module definition: the k-subset modules defined in [6] and the Courcelle's modules defined in [11]. In this section we will first recall them and study their relationships to the standard one (Definition 4).

Definition 5 ***(k-subset module*** [6]***).*** *Given a hypergraph H, we call* ***k-subset module*** *$M \subseteq V(H)$ satisfies: $\forall A, B \subseteq V(H)$ s.t. $2 \leq |A|, |B| \leq k$ and $A \cap M \neq \emptyset$, $B \cap M \neq \emptyset$ and $A \setminus M = B \setminus M \neq \emptyset$ then $A \in \mathcal{E}(H) \Leftrightarrow B \in \mathcal{E}(H)$.*

If H is a 2-uniform hypergraph (i.e., an undirected graph) the 2-subset modules are simply the usual graph modules. Families of k-subset modules also yield a partitive family [6].

Definition 6 ***(Courcelle's module*** [11]***).*** *Given a hypergraph H, we call* ***Courcelle's module*** *a set $M \subseteq V(H)$ that satisfies $\forall A \in \mathcal{E}(H)$, $A \perp M$.*

Courcelle's modules using our notations of Sect. 1 just correspond to $\mathcal{E}(H)^{\perp}$. Using Property 1 these modules yield a partitive family. This notion seems to be far from the standard hypergraph module definition [19,20], this is why we called them Courcelle's modules. Indeed, applied to graphs, the orthogonal of the edge-set is the connected components (plus the vertex-set and the singletons) of the graph, not the modules. A direct application of Theorem 2 on orthogonal families gives the following corollary:

Corollary 2. *Courcelle's modular decomposition tree can be computed in $O(l)$.*

3 General Decomposition Scheme for Partitive Families

Definition 7. *For a partitive family $\mathcal{F}$ on a ground set V, using the closure by intersection of partitive families, we can define for every $A \subseteq V$, $Minmodule(A)$ as the smallest element of $\mathcal{F}$ that contains A. In particular, let us denote by $\mathcal{M}_{x,y}$ the family of all $Minmodule(\{x, y\})$ and $\forall x, y \in V$.*

Although $\mathcal{M}_{x,y}$ does not contain all $\mathcal{F}$, simply because $|\mathcal{F}|$ can be exponential in $|V|$ while $|\mathcal{M}_{x,y}|$ is always quadratic. In this section we propose an algorithm scheme to compute the decomposition tree of a partitive family if the only access to the family is a call of a function that computes: for every $A \subseteq V$, $Minmodule(A)$. Thus designing an efficient algorithm is to minimize the total number of calls. We will now show a simple way to extract the decomposition tree, i.e., the strong elements from of $\mathcal{M}_{x,y}$.

According to the definition, if a node has only two children, we cannot distinguish whether this node is prime or complete. We take the convention that the node is prime in this case. After constructing the tree, we can easily transform it into another convention just by labeling all nodes with only two children as complete nodes.

Theorem 3. *For every partitive family $\mathcal{F}$ over a ground set V, its decomposition tree can be computed using $O(|V|^2)$ calls to $Minmodule(\{x, y\})$, with $x, y \in V$.*

Proof. First choose an initial vertex x_0 and compute $Minmodule(\{x_0, x\}), \forall x \neq x_0 \in V$ and add them to a set $\mathcal{M}$. Then we add all singletons to $\mathcal{M}$. Let μ the unique path from x_0 to the root in the decomposition tree.

Claim 1: Every prime node of μ, belongs to $\mathcal{M}$.

Proof. Consider a prime strong element $A \in \mu$, it corresponds to some node of the tree which admits children $A_0, A_1, \ldots A_k$, with $k \geq 1$ in the decomposition tree. If $x_0 \in A_0$, and take $y \in A_1$, then $Minmodule(\{x_0, y\} = A$, since A is the least common ancestor in the decomposition tree. □

Claim 2: For a complete node $A \in \mu$, with children $A_0, \ldots A_k$, if $x_0 \in A_0$, then
(i) for every $1 \leq i \leq k$ the set $A_0 \cup A_i$ belongs to $\mathcal{M}$
(ii) when elements of $\mathcal{M}$ are sorted by their size, $A_0 \cup A_i$ appear consecutively.

Proof. (i) In fact for every $y \in A_i$, $Minmodule(\{x_0, y\}) = A_0 \cup A_i$. Note that it may be possible that $A_0 = \{x_0\}$.

(ii) If there exists a prime node P such that $\exists i, j$ such that $|A_0 \cup A_i| < |P| < |A_0 \cup A_j|$. Since $x_0 \in P$ and $x_0 \in A_0$ then P must overlap with $A_0 \cup A_i$ or

$A_0 \cup A_j$, which contradicts the fact that P is a prime node that overlaps no other element in the family. □

The above arguments also show that any $Minmodule(\{x_0, y\})$ corresponds to either a prime node, or the union of two children of a complete node. So the family $\mathcal{M}$ is made up with prime nodes that overlap no other subsets and some daisies, and they all contain x_0, where daisies are these subsets $A_0 \cup A_i$, all containing A_0, the A_i's being the petals of the daisy and A_0 its center. Note that a daisy is a simple particular case of overlap component.
Now to find the decomposition tree we can apply the following algorithm:

1. Sort elements in $\mathcal{M}$ by size, eliminate multiple occurrences of a subset in $\mathcal{M}$.
2. Scan this list in increasing order and checking if the new considered subset overlaps the previous, else merge it to the previous it with **complete** (label both the two as complete) and continue. After the iteration, there is no unlabeled set that overlaps with another unlabeled set, then mark every unlabeled set with the label **prime**.
   Then labeled sets $X_0, X_1, \ldots, X_h$ provide the path from x_0 to the root of the modular decomposition tree, namely: $\mu = [\{x_0\} = X_0, X_1, \ldots, X_h = V]$.
3. Let us consider the partition $\{V_0, \ldots, V_h\}$ of V defined as follows:
   $V_i = \{x \in V | Minmodule(x_0, x) = X_i\}$ for $0 \leq i \leq h$.
   For every $0 \leq i \leq h$ recurse on the partitive family over the ground set V_i by computing the path from a vertex $x \in V_i$ that haven't been computed and attach its tree to X_i.

Claim 3: Every node constructed is a strong module.

Proof. Assume node X, $x_0 \in X$ overlapping with some module X'. We take any element $x' \in X' \setminus X$, then $X \Delta X'$ is a module and thus $Minmodule(x_0, x') \subseteq X \Delta X'$, which overlaps with X. Thus $Minmodule(x_0, x')$ must have been merged into X, contradiction. □

The validity of the claim directly follows from Claims 1 to 3. For Step 1 we can use any linear sorting by value algorithm, since the size of the subsets are bounded by $n = |V|$. Clearly Step 2 can be done linear time in the size of $\mathcal{M}$. So the bottleneck of complexity is the number of calls of $Minmodule(\{x, y\})$, which is bounded by n^2. □

Consequently, if computing the function $Minimal$ of a given partitive family can be done $O(p(n))$time, then the computation of the decomposition tree can be done in $O(n^2 \cdot p(n))$. Applied to graphs it yields an $O(n^2 m)$ algorithm, far from being optimal. Such an approach was already used for graphs in [15]. Let us now consider how to compute this function for the three variations of hypergraph modules defined previously.

4 Computing Minimal-Modules for Hypergraphs

For undirected graphs, computing Minimal-modules can be done via a graph search in linear time. We generalize this to hypergraphs for two out of the three

Algorithm 1. Modular-closure

```
   Data: $H$ a simple hypergraph and $W \subsetneq V(H)$
   Result: The minimal module of $H$ that contains $W$
 1 Compute a lexicographic ordering $\tau$ of $\mathcal{E}(H)$ w.r.t an arbitrary ordering of $V$,
 2 $C \leftarrow W$, $X \leftarrow W$,
 3 Compute the induced hypergraph $H(C)$,
 4 for $1 \leq i \leq |\mathcal{E}(H(C))|$ do
 5     Compute the ordered lists $L_i$ of the restriction to $V(G) \setminus C$ of the edges in
       $\mathcal{E}(H)$ that contain $f_i \in \mathcal{E}(H(C))$
 6 $\mathcal{Q}(C) \leftarrow \{L_1, \dots, L_{|\mathcal{E}(H(C))|}\}$ the ordered partition made up with these lists
 7 if $|\mathcal{Q}(C)| = 1$ then
 8     $C$ is a module, STOP
 9 else
10     $L \leftarrow First(\mathcal{Q}(C))$, {% the first class in the ordered partition%}
11     while $Next(L) \neq NIL$ {% the next element of L in the ordered partition%}
       do
12         $X \leftarrow Comparison(L, Next(L))$
13         if $X = \emptyset$ then
14             $L \leftarrow Next(L)$
15         else $C \leftarrow C \cup X$, update $\mathcal{Q}(C)$ via partition refinement with $X$,
           $L \leftarrow First(L)$
16           {%if L has been split during the update we take its first part%}
17
18 $RESULT \leftarrow C$ {%$C$ is a minimal module that contains $W$%}
```

definitions of modules: the standard and the k-subset module. For efficiency purposes, we represent our hypergraphs using for each vertex x a list to represent $N(x)$ i.e., the edges its belongs to, and for each edge e a list to represent $H(e)$ i.e., the vertices it contains. For a hypergraph this yields a representation using $O(n + m + l)$ memory. If the hypergraph is simple then $O(n + m + l) = O(l)$.

4.1 Standard Modules

Definition 8. *For a set $C \subsetneq V(H)$, an edge $A \in \mathcal{E}(H)$ is a **edge-splitter** for C, if $A \setminus C \neq \emptyset$ and $A \cap C \neq \emptyset$ and if there exists $B \in \mathcal{E}(H)$ s.t. $B \cap C \neq \emptyset$, and $(A \setminus C) \cup (B \cap C) \notin \mathcal{E}(H)$.*

In other words, a set of vertices is a module iff it admits no edge-splitter.

Property 4. If $X \subseteq V(H)$ is a splitter of $C \subseteq V(H)$ respect to A, B as above. Let $B' \in \mathcal{E}(H)$ be the edge such that $B' \cap C = B \cap C$ and with $|(B' \setminus C) \Delta (A \setminus C)|$ minimum.

Let $X' = (B' \setminus C) \Delta (A \setminus C)$, then there is no module Y of H such that $C \subsetneq Y$ but $X' \nsubseteq Y$.

Theorem 4. *If H is a simple hypergraph, for every set $W \subseteq V(H)$, Algorithm Modular-closure computes $Minmodule(W)$ in $O(n \cdot l)$. And its modular decomposition tree can be computed in $O(n^3 \cdot l)$).*

Algorithm 2. Procedure Comparison

```
Data: 2 lists $L', L''$ of the restriction to $V(G) \setminus C$ of the edges in $\mathcal{E}(H)$ that
      contain some $f \in \mathcal{E}(H(C))$. They are supposed to be lexicographically
      increasingly ordered using $\tau$
Result: X a set of vertices forced be contained in Minmodule(C)
1  if $L' = L''$ then
2      $X \leftarrow \emptyset$, STOP
3  else
4      Let $e \in L'$ and $f \in L''$ be the first lexicographically difference,
5      if $(e <_\tau f) or (e \neq \emptyset$ and $f = \emptyset)$ then
6          {% $e \notin L''$ is a edge-splitter %}
7          compute $f' \in L''$ that minimizes $|h(e)\Delta h(f')|$ with $f' \in L''$
           $X \leftarrow h(e)\Delta h(f')$
8      else
9          {% $(e >_\tau f) or (e = \emptyset$ and $f \neq \emptyset)$, i.e. $f \notin L'$ is a edge-splitter %}
10         compute $e' \in L'$ that minimizes $|h(f)\Delta h(e')|$ with $e' \in L'$,
11         $X \leftarrow h(f)\Delta h(e')$
12
13 {%Note that $e = \emptyset$, $f \neq \emptyset$ (resp. $e \neq \emptyset$, $f = \emptyset$) corresponds to the case
   $|L| < |Next(L)|$ (resp. $|L| > |Next(L)|$)%}
```

Proof. (i) **Correctness:** First we notice that C is a module of H iff all the lexicographically sorted lists L_i are equal. At each step of the lexicographic process a list can only be cut into parts, no lists are merged. If at some step of the algorithm two lists L_i, L_j are equal, and if afterwards they are cut into sublists via the refinement process, equality between sublists is preserved since the refinement act similarly on the lists. Thus the algorithm scan the lists form left to right using a single sweep and the following invariant: at each step of the while loop all the lists before the current list L are all equal to L.

Using the procedure Comparison either the lists are equal and then we proceed else using Property 4 we know that we can add this set of vertices. At the end of the algorithm either all lists are equals and $C \neq V(H)$ and therefore C is the non trivial minimal module containing W or $C = V(H)$ and there is no other module between W and $V(H)$.

(ii) **Complexity Analysis:** To implement the first step (line 1) we can use an ordered partition refinement technique on $\mathcal{E}(H)$ (see [13]) using the sets $N(x)$ for every $x \in V(H)$ as pivot sets. This provides a total ordering τ of $\mathcal{E}(H)$. This can be done in $O(n + m + l)$.

To compute $\mathcal{Q}(C)$, we can use the same ordered partition refinement technique using the sets $N(x)$ for every $x \in C$ as pivot sets we can compute the ordered partition of $\mathcal{E}(H)$. Starting from the partition $P_0 = \{\mathcal{E}(H)\}$, we refine this partition successively using $N(x_i)$ for every $x_i \in C$. Let us denote by P_f the partition obtained after this round of refinements. Each part of P_f can be ordered using τ, since partition refinement can maintain an initial ordering of its elements within the same complexity. So if we start with the initial ordering τ in the unique part of P_0. And the parts are lexicographically ordered with respect to their intersection to C. This can be done in $O(|C| + \Sigma_{x \in C}|N(x)|)$.

In fact after line 6 we can ignore the vertices of C, a similar remark holds when C is updated.

Now we have to check if all edges lists L_i are identical or not and stop at the first difference. Since the lists are ordered lexicographically using an ordering τ of the vertices, a simple scan of these ordered lists is enough to compute (Comparison procedure) of Algorithm 1.

When C and $\mathcal{Q}(C)$ are updated, the algorithm goes on with the first part of the previous current list L. First means that if L has been split during the update we take its first part. Therefore in the worst case some list can be analyzed several times (at most n times) and therefore the overall complexity of the list scan is bounded by $O(n \cdot l)$.

When a difference is found between two lists we have to search for an edge that minimizes the symmetric distance with respect to the differentiating edge. Even though it can be done several times for a given edge, but every time we launch this search, at least one vertex will be added into C, thus at most search for n times. So the overall complexity of these searches is $O(n \cdot l)$.

Therefore the whole process is in $O(n+m+l+n \cdot l) = O(n \cdot l)$. Using Theorem 3 we obtain the decomposition tree in $O(n^3 \cdot l)$. □

Up to our knowledge, [20] states there is a polynomial time decomposition algorithm for clutters based on its $O(n^4 m^3)$ modular closure algorithm without precising the complexity, our algorithm is an improvement because our total decomposition time is already smaller than $O(n^4 m^3)$ s.

4.2 Decomposition into k-Subset Modules

Definition 9 [6]. *A subset $X \neq \emptyset$ is a **k-subset splitter** of the set C if there exist $A, B \subseteq V$ s.t. $2 \leq |A|, |B| \leq k$ and $A \cap C \neq \emptyset$, $B \cap C \neq \emptyset$ and $A \setminus C = B \setminus C = X$, $A \in \mathcal{E}(H)$ but $B \notin \mathcal{E}(H)$.*

Lemma 1. *Given a set $C \subsetneq V(H)$, any k-subset splitter of C is in the form of $H(e) \setminus C$ for some $e \in \mathcal{E}(H)$, $|H(e)| \leq k$.*

Such an edge will be called an **k-edge-splitter** of C. Let $D(k, h) = \Sigma_{i=1}^{i=k} \binom{h}{i}$ for $1 \leq k \leq h$, where $\binom{h}{i}$ denotes the binomial coefficient. All values of $D(k, h)$ strictly greater that $|\mathcal{E}(H)|$ will be set as **Out-of-Range**, a huge number.

Lemma 2. *For a simple hypergraph H, given a set $C \subsetneq V(H)$ and an edge $e \in \mathcal{E}(H)$ s.t. $|H(e)| \leq k$, $e \cap C \neq \emptyset$ and $X = e \setminus C \neq \emptyset$, let L be the list of edges in $\mathcal{E}(H)$ with size $\leq k$ and whose intersection with $V(H) \setminus C$ are identical to X and intersection with C is not empty, i.e. $L = \{e' \in \mathcal{E}(H) \mid e' \setminus C = X,\ e' \cap C \neq \emptyset$ and $|H(e')| \leq k\}$. If $|L| < D(k - |X|, |C|)$ then e is an edge-splitter of C.*

Proof. It is equivalent to check for such an e given above and $X = e \setminus C$, whether every non empty subset B of size $\leq k - |X|$ in C has $X \cup B \in \mathcal{E}(H)$. Since H is simple and there are no identical elements in L, a counting argument captures the condition. Moreover, the number of subsets checked this way is $\leq |\mathcal{E}(H)|$. □

Algorithm 3. k-subset modular-closure

```
1  Algorithm: k-subset modular-closure
   Data: $H$ a simple hypergraph, k an integer such that $1 \leq k \leq |V(H)|$ and
         $W \subsetneq V(H)$
   Result: The minimal k-module of $H$ that contains $W$
2  Compute all $D(k,h)$ for $|W| \leq h \leq |V(H)|$,
3  Compute a lexicographic ordering $\tau$ of $\mathcal{E}(H)$ with respect to some ordering of
   the vertices,
4  $C \leftarrow W$, $X \leftarrow W$,
5  while $X \neq \emptyset$ do
6      For every $e_i \in \mathcal{E}(H)$ overlap $C$, $|H(e_i)| \leq k$, create the lists $L_i$ of edges in
       $\mathcal{E}(H)$ with size $\leq k$ whose intersection with $V(H) \setminus C$ are identical to $e_i \setminus C$
       and of size $h$ and intersection with $C$ is not empty
7      if For some $i$, $|L_i| < D(k-h, |C|)$ then
8          $X \leftarrow X \cup (H(e_i) \setminus C)$,
9          if $V(G) = C \cup X$ then
10             % there is no non-trivial module between $W$ and $V(H)$ %
11             $RESULT \leftarrow V(H)$, STOP
12         else
13             %$X$ is a splitter for $C$%,
14             $C \leftarrow C \cup X$
15
16     else $X \leftarrow \emptyset$
17 $RESULT \leftarrow C$ %$C$ is a non trivial module containing $W$%
```

Theorem 5. *For a simple hypergraph H and $A \subsetneq V(H)$, for any fixed integer $k \leq |V(H)|$, Algorithm 3 (k-subset modular-closure) can compute the minimal k-subset module that contains A in $O(n \cdot m \cdot l)$ time, which gives a $O(n^3 \cdot m \cdot l)$ decomposition algorithm.*

5 Using Orthogonality for Graph Modular Decomposition

5.1 Factorizing Permutations and Fractures

Given a permutation σ of a set V, let $\sigma(i)$ denote the i^{th} element of V. An interval $[l, r]$ of σ is a set of elements that follow consecutively $(1 \leq l \leq r \leq n)$.

Definition 10. *Let $\mathcal{F}$ be a family of subsets of V. A permutation σ of V is a **factorizing permutation** if every strong set of $\mathcal{F}$ is an interval of σ. Furthermore, it is **perfect** if every set of $\mathcal{F}$ is an interval.*

Factorizing permutations were defined in the context of modular decomposition [8] but can be generalized to any subset family, where a permutation can be obtained by a traversal of its decomposition tree. We adapt a definition from [8]:

Definition 11. *Let $G = (V, E)$ be a graph and σ a permutation of V. Let us consider a pair $\{\sigma(i), \sigma(i+1)\}$ of two consecutive elements. The* ***left fracture*** *(resp.* ***right fracture****) of i is the largest interval $[s, i]$ (resp. $[i+1, s]$) where $\sigma(s)$ is a splitter of $\{\sigma(i), \sigma(i+1)\}$. If that pair admits no splitter on its left (resp. right) then i has no left (resp. right) fracture. The* ***fracture family****, denoted $\mathcal{F}rac(\sigma)$, of a given permutation of the vertices of a graph is the set of all (left and right) fractures for all $1 \leq i < n$.*

Lemma 3. *Given a graph G and a permutation σ of its vertices, an interval I of σ is a module iff it does not overlap any fracture of $\mathcal{F}rac(\sigma)$.*

Lemma 4. *Let σ be a factorizing permutation of the modules family of G, F be a fracture of σ, and M the smallest module containing F. M is a strong module.*

Theorem 6. *Given a graph G, the family $\mathcal{M}$ of its modules and a factorizing permutation σ of $\mathcal{M}$ we have: $\mathcal{P}(\mathcal{M}) = \mathcal{F}rac(\sigma)^{\perp}$. Furthermore, if the graph is undirected then $\mathcal{M} = \mathcal{F}rac(\sigma)^{\perp}$.*

5.2 Modular Decomposition of Tournaments

We can apply the theorem above to undirected graphs where we get a new algorithm but no improvement of the existing algorithms, or to tournaments (orientation of the complete graph), and we get an simple (much simpler than the existing algorithm in [18]) and optimal modular decomposition algorithm. Among the families admitting a perfect factorizing permutation are the ***anti-symmetric-partitive families***, families where Axiom ii of Definition 1 is replaced with: $\forall A, B \in \mathcal{F}$ that overlap, $A \cap B, A \cup B, A \setminus B \in \mathcal{F}$ and $A \Delta B \notin \mathcal{F}$. Their decomposition tree is often called a ***PQ-tree***, whose nodes are labeled P (prime) and Q (having a linear ordering of the siblings so that any union of siblings that follow consecutively belong to the family, and no other union). Well-known antisymmetric-partitive families are the intervals of the real line (their intersection model being an interval graph), the common intervals of two permutations [2], or the modules of a tournament.

Theorem 7. *Let G be a tournament. The modular decomposition tree of G can be computed in $O(n^2)$ time.*

6 Conclusions

In this paper, using a general framework for decomposition of partitive families or tools of orthogonality, we have proposed 3 polynomial algorithms to compute hypergraph modular decomposition trees under 3 different definitions of modules. Our general framework yields a $O(n^3 \cdot l)$ algorithm for the standard decomposition of hypergraphs, and a $O(n^3 \cdot m \cdot l)$ time for k-subset modules, an improvement to the previously known non-polynomial $O(n^{3k-5})$ time algorithm [6], where k denotes the maximal size of an edge. Since our approach is brute force, there may exists linear time (in $O(l)$) algorithms for the standard hypergraph decomposition, as for graphs [14]. One would have to develop new hypergraph algorithms, for example one that computes in linear time some factoring permutation which always exists for every partitive family and use some orthogonality.

Conjecture 1. Simple hypergraphs admit a linear time $O(l)$ modular decomposition algorithm.

References

1. Berge, C.: Graphes et hypergraphes. Dunod, Paris (1970)
2. Bergeron, A., Chauve, C., de Montgolfier, F., Raffinot, M.: Computing common intervals of K permutations, with applications to modular decomposition of graphs. SIAM J. Discrete Math. **22**(3), 1022–1039 (2008)
3. Billera, L.J.: Clutter decomposition and monotonic boolean functions. Ann. N.-Y. Acad. Sci. **175**, 41–48 (1970)
4. Billera, L.J.: On the composition and decomposition of clutters. J. Combin. Theory, Ser. B **11**(3), 234–245 (1971)
5. Bioch, J.C.: The complexity of modular decomposition of boolean functions. Discrete Appl. Math. **149**(1–3), 1–13 (2005)
6. Bonizzoni, P., Vedova, G.D.: An algorithm for the modular decomposition of hypergraphs. J. Algorithms **32**(2), 65–86 (1999)
7. Borassi, M., Crescenzi, P., Habib, M.: Into the square: on the complexity of some quadratic-time solvable problems. Electr. Notes Theor. Comput. Sci. **322**, 51–67 (2016)
8. Capelle, C., Habib, M., de Montgolfier, F.: Graph decompositions and factorizing permutations. Discrete Math. Theor. Comput. Sci. **5**(1), 55–70 (2002)
9. Charbit, P., Habib, M., Limouzy, V., de Montgolfier, F., Raffinot, M., Rao, M.: A note on computing set overlap classes. Inf. Process. Lett. **108**(4), 186–191 (2008)
10. Chein, M., Habib, M., Maurer, M.C.: Partitive hypergraphs. Discrete Math. **37**(1), 35–50 (1981)
11. Courcelle, B.: A monadic second-order definition of the structure of convex hypergraphs. Inf. Comput. **178**(2), 391–411 (2002)
12. Dahlhaus, E.: Parallel algorithms for hierarchical clustering and applications to split decomposition and parity graph recognition. J. Algorithms **36**(2), 205–240 (2000)
13. Habib, M., McConnell, R.M., Paul, C., Viennot, L.: Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. Theor. Comput. Sci. **234**(1–2), 59–84 (2000)
14. Habib, M., Paul, C.: A survey of the algorithmic aspects of modular decomposition. Comput. Sci. Rev. **4**(1), 41–59 (2010)
15. James, L.O., Stanton, R.G., Cowan, D.D.: Graph decomposition for undirected graphs. In: Proceedings of the 3rd Southeastern International Conference on Combinatorics, Graph Theory, and Computing, pp. 281–290 (1972)
16. McConnell, R.M.: A certifying algorithm for the consecutive-ones property. In: SODA, Proceedings of the Fifteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp. 768–777 (2004)

17. McConnell, R.M., de Montgolfier, F.: Algebraic operations on PQ trees and modular decomposition trees. In: Kratsch, D. (ed.) WG 2005. LNCS, vol. 3787, pp. 421–432. Springer, Heidelberg (2005). https://doi.org/10.1007/11604686_37
18. McConnell, R.M., de Montgolfier, F.: Linear-time modular decomposition of directed graphs. Discrete Appl. Math. **145**(2), 198–209 (2005)
19. Möhring, R., Radermacher, F.: Substitution decomposition for discrete structures and connections with combinatorial optimization. In: Proceedings of the Workshop on Algebraic Structures in Operations Research, pp. 257–355 (1984)
20. Möhring, R.H.: Algorithmic aspects of the substitution decomposition in optimization over relations, set systems and boolean functions. Ann. Oper. Res. **4**, 195–225 (1985)