An algorithm for the modular decomposition of hypergraphs

Paola Bonizzoni and Gianluca Della Vedova
Dipartimento di Scienze dell'Informazione
Università Degli Studi di Milano
Via Comelico 39/41, 20135 Milano – ITALY

e-mail {bonizzon,dellaved}@dsi.unimi.it

Abstract

We propose an $O(n^4)$ algorithm to build the modular decomposition tree of hypergraphs of dimension 3 and show how this algorithm can be generalized to compute in $O(n^{3k-5})$ time the decomposition of hypergraphs of any fixed dimension k.

© 1999 by Academic Press. This material has been published in Journal of Algorithms 32(2), pp. 65 – 85, 1999, the only definitive repository of the content that has been certified and accepted after peer review. Copyright and all rights therein are retained by Academic Press. This material may not be copied or reposted without explicit permission. This paper is available on IDEAL, http://www.idealibrary.com

---

Preliminary version of this paper appeared in "Proceedings of the 21^{st} Workshop on Graph-Theoretic Concepts in Computer Science," *Lecture Notes in Computer Science*, Vol. 1017, pp. 303-317, Springer-Verlag, Berlin/New York, 1995

1 Introduction

The modular decomposition of a graph gives a tree representation (or *partitive tree* [14]) of the graph by means of *modules*, subsets of vertices that share the common property of being adjacent to the same vertices outside the module. Such a tree can represent in $O(n)$ space all modules of a graph, even though they can be in exponential number w.r.t. n.

Given such a tree representation, it is possible to solve certain combinatorial problems on the graph, by designing an algorithm which derives the solution of the problem from those for the single components induced by the modules of the decomposition tree. This technique allows to design the most known efficient algorithms for computing the maximum weighted clique, the maximum independent set and the minimum number of cliques necessary to cover all the vertices of a graph and other combinatorial problems on particular classes of graphs [12].

These problems are NP-hard for general graphs, but they can be solved in polynomial time for restricted classes of graphs that admit a partitive tree satisfying certain required properties. In fact, while the modular decomposition allows to solve efficiently problems on graphs, on the other end it can be used to classify graphs for which such solution is possible, as by this form of representation structural properties of graphs can be analyzed.

The importance of this form of representation for graphs has been recently pointed out by the first linear-time algorithms [11, 2] for computing the modular decomposition, which are fundamental for solving problems on *comparability graphs* [9], i.e graphs obtained by eliminating the orientation in digraphs representing partial orders. Moreover the modular decomposition turns out to be of interest for other relational structures besides graphs. In fact, a generalization of it has been to *k-ary relations* in [14] and to *2-structures* in [6, 7], where a 2-structure on a domain D is a labeling of all antireflexive pairs on D. This second generalization of the decomposition reveals to be useful in solving other different problems in Computer Science [8], and has been studied widely. Efficient algorithms for the decomposition of 2-structures have been proposed in [4, 10].

Another generalization has been proposed with the notion of a *k-structure* [5], which is a labeling of all antireflexive k-tuples on a domain D, that is of the form $\{x_1, \cdots, x_k\}$ with $x_j \neq x_i$ for some $1 \leq i, j \leq k$. In [1], those k-structures that cannot be decomposed into nontrivial modules are analyzed. The notion of a k-structure is strictly related to that of a hypergraph, precisely the modular decomposition for hypergraphs is the more natural generalization of the one for graphs. This form of decomposition for general set systems, such as arbitrary clutters, and its possible applications, mainly in combinatorial optimization, has been analyzed and motivated in [13, 14], where the first polynomial algorithm for arbitrary clutters is discussed.

In this paper, we give a generalization of the notion of module which is different from the one for k-structures and allows us to define a context to generalize the modular decomposition to hypergraphs. Then, we propose a polynomial algorithm for it. An approach to the problem of developing an algorithm for the decomposition of hypergraphs is discussed in [10], where a framework for computing the modular decomposition of k-structures by an incremental algorithm is proposed.

Our algorithm is based on a recursive technique for computing the modular decomposition tree of hypergraphs of fixed dimension. We first propose an $O(n^4)$ algorithm for the modular decomposition of hypergraphs of dimension 3. Then we show how our algorithm generalizes to hypergraphs of higher dimension k, by using a recursive technique which computes the decomposition tree for a hypergraph of dimension k, from the one for hypergraphs of dimension $k-1$.

The general algorithm for k-hypergraphs requires $O(n^{3k-5})$ time.

2 Preliminaries

Let A, B be two sets. Then A, B *overlap* iff $A - B$, $B - A$ and $A \cap B$ are nonempty sets. The set A is a *k-subset* of a set V, if $A \subseteq V$ and $2 \leq |A| \leq k$. The symmetric difference of two sets A and B, denoted as $A \triangle B$, is the set $(A \cup B) - (A \cap B)$.

Definition 2.1 *A* k-hypergraph *is a pair $H = (V, E)$, where V is a finite set of elements, called* vertices *and $E \subset 2^V$ is such that if $e \in E$ then e is a k-subset of V. The elements of E are called the* hyperedges *of H.*

We will say that two vertices v and w are *i-adjacent* if there exists a hyperedge consisting of exactly i vertices and containing both v and w. For example, if $\{v_1, v_2, v_3, v_4\}$ is a hyperedge, then the vertex v_3 is 4-adjacent to every element in the set $\{v_1, v_2, v_4\}$ of vertices.

In the following, by hypergraph we mean a k-hypergraph for some fixed k.

Definition 2.2 *Let $H = (V, E)$ be a hypergraph and assume $X \subseteq V$. The* subhypergraph induced by X, *denoted as $H|X$ is the hypergraph $H' = (X, E \cap 2^X)$.*

The notion of modular decomposition tree for a hypergraph can be obtained by the following notion of partitive set family. A family F of sets on a domain D is a *partitive set family* [14] iff $D \in F$, each singleton subset of D is a member of F, and whenever X and Y are overlapping members of F, then $X \cap Y$, $X \cup Y$, $X - Y$, $Y - X$ are also members of F.

A *partitive set family* F has a tree representation, the *partitive tree* of F [14] which is constructed as follows: the members of the family that overlap no other, called *prime* sets, are the nodes of the tree and the containment relation on these sets gives the parent relation in the tree. Each internal node X of the partitive tree is labeled as:

- *q-complete* if the union of any subset of its children is a member of F,
- *q-primitive* if each of its children is a member of F, while no other union of a proper subset of its children is a member of F,
- *q-linear* if there is a linear order of all children of X such that the union U of a subset of its children is a member of F iff the elements in U are consecutive in such a linear order.

The following is a fundamental result on partitive trees [14] which will be used in the paper.

Lemma 2.1 *Let F be a partitive set family over the domain D. If a set $X \subseteq D$ is a member of F, then it is a node in the partitive tree of F or a union of children of an internal node of such a tree.*

A partitive set family has been defined for some relational structures, such as graphs, k-ary relations [14] and k-structures [5], by introducing the notion of module: the partitive tree associated to the family of modules of these structures is their *modular decomposition tree*. In the case of a graph $G = (V, E)$, a set M, with $M \subseteq V$, is a *module* of G if there does not exist a vertex v, with $v \in V - M$ such that v is adjacent to at least one vertex in M, but not to all vertices in M. The notion of *module* can be generalized to hypergraphs by using the following relation on sets.

Definition 2.3 *Let $H = (V, E)$ be a hypergraph and let A, B be k-subsets of V and $C \subseteq V$. The two sets A, B are in* C-relation*, which is denoted as $A =_C B$, iff $A \cap C \neq \emptyset$, $B \cap C \neq \emptyset$ and $A - C = B - C \neq \emptyset$.*

Two sets are in C-relation when they are identical outside C and the intersection with C of each of these sets is nonempty. It is immediate to verify that a C-relation is symmetric and transitive.

Definition 2.4 *Let $H = (V, E)$ be a hypergraph and $M \subseteq V$. Then M is a* module *of H iff for every pair of k-subsets $A, B \subseteq V$, whenever $A =_M B$ then A is a hyperedge iff B is a hyperedge.*

A module is a subset of vertices which cannot be distinguished from outside of it, according to the following notion.

Definition 2.5 *Let $H = (V, E)$ be a hypergraph and X a nonempty subset of V, $Y \subseteq V$. Then a set X* distinguishes *a set Y iff for some A, B k-subsets of V, where $A =_Y B$ and $A - Y = B - Y = X$ it is $A \in E$ and $B \notin E$.*

Lemma 2.2 *Let $H = (V, E)$ be a hypergraph and Y, Z be two overlapping subsets of V, $X \subseteq V$. If Y, Z cannot be distinguished by X, then $Y \cup Z$ cannot be distinguished by X.*

Proof: Let A be a k-subset of V, with $A - (Y \cup Z) = X$ and $A \cap (Y \cup Z) \neq \emptyset$. Then $A =_{Y \cup Z} X \cup \{x\}$, for $x \in Y \cap Z$. By transitivity of the C-relation, the Lemma follows. □

Note that the notion of a module is the same for graphs as for hypergraphs: a set M of vertices is a module if it cannot be distinguished by sets of vertices outside M.

Clearly, by the previous definition 2.4, V, $\emptyset$ and the singleton subsets of V are modules of H: these are the *trivial* modules of a hypergraph.

The following Lemma has been proved in [14] for arbitrary set systems and a proof of a similar Lemma has been given for k-structures in [5].

Lemma 2.3 *The family of nonempty modules of a hypergraph is a partitive set family.*

Proof: Let M_1, M_2 be two overlapping modules of a hypergraph $H = (V, E)$. Then, we show that: i) $M_1 \cap M_2$, ii) $M_1 - M_2$, iii) $M_1 \cup M_2$ are all nonempty modules of the hypergraph.

i) Let A, B be two subsets of V such that $A =_{M_1 \cap M_2} B$. Since $A - (M_1 \cap M_2) = B - (M_1 \cap M_2) \neq \emptyset$, it follows that it must be $A - M_1 = B - M_1$ and $A - M_2 = B - M_2$, and clearly either $A - M_1$ or $A - M_2$ is not empty, otherwise A and B are contained in $M_1 \cap M_2$.

Assume that $A - M_1 = B - M_1 \neq \emptyset$. Then, $A =_{M_1} B$, as $A \cap M_1 \neq \emptyset$ and $B \cap M_1 \neq \emptyset$. Since M_1 is a module, it follows that $A \in E$ iff $B \in E$, which proves the statement i).

ii) Let A, B be two subsets of V such that $A =_{M_1 - M_2} B$. If A and B are not contained in M_1, then $A =_{M_1} B$, and since M_1 is a module it follows that $A \in E$ iff $B \in E$, which proves what required. Otherwise, assume that $A, B \subseteq M_1$. Since $A - (M_1 - M_2) \neq \emptyset$ and $B - (M_1 - M_2) \neq \emptyset$, it follows that A and B must have common elements with the set $M_1 \cap M_2 \neq \emptyset$ and $A - M_2 \neq \emptyset$ and $B - M_2 \neq \emptyset$. Then, pose $A_1 = A \cup \{x\}$ and $B_1 = B \cup \{x\}$, for $x \in M_2 - M_1$. Then, $A_1 =_{M_1} B_1$ and $A =_{M_2} A_1$, $B =_{M_2} B_1$. Since M_2 and M_2 are modules, by transitivity $A \in E$ iff $B \in E$, and hence property ii) follows.

iii) Immediate by Lemma 2.2 and definition of module.

□

The previous Lemma 2.3 allows to give the following definition.

Definition 2.6 *The* modular decomposition tree of a hypergraph H *is the partitive tree associated to its partitive set family.*

The modules in the partitive set family which are prime sets will be called *prime modules* of the decomposition tree.

By the definition of a partitive tree, an internal node of such a tree with more than two children is either q-primitive, q-complete or q-linear, while if an internal node has two children, we cannot distinguish between these three notions. We will assume that in the partitive tree of a hypergraph, every internal node with two children is labeled q-complete.

Let P be a partition of the vertices of H. A *system of representatives* of P is a set containing a vertex from each set in P. If each member of P is a module, then P is called a *congruence partition*. This partition induces a hypergraph, the *quotient hypergraph* H/P which is simply constructed as follows: the nodes of H/P are the sets in P, and for $X_1, \ldots, X_n$ sets in P, $\{X_1, \ldots, X_n\}$ is a hyperedge of H/P iff the set $\{x_1, \ldots, x_n\}$ with $x_i \in X_i$, for every i, $1 \leq i \leq n$ is a hyperedge of H. In fact, it can be easily shown that each system of representatives S of a congruence partition induces a subhypergraph of H, $H|S$, which is isomorphic to the quotient. It follows that we can identify the hypergraph H/P with the hypergraph $H|S$. The *representative* x of a module X, with $X \in P$, is the *image* of X in S, while X will be the *inverse image* of x in P.

Moreover, given an arbitrary module M of H, the image of M in S is the set $M \cap S$. Given M a module of H/P, its inverse image in H, will be the union of the inverse images of all elements of M in P.

The results in [5] on k-structures suggest interesting facts about properties of quotients and modules of hypergraphs. Just as for graphs and k-structures, we can easily prove that a *restriction rule* holds for hypergraphs: given a module M of a hypergraph $H = (V, E)$ and $Y \subseteq V$, then $M \cap Y$ is a module of the induced hypergraph $H|Y$.

Using the above rule, along the same lines of the proof in [7, 5], we can derive a *quotient rule* for hypergraphs:

Lemma 2.4 (quotient rule) *Let $H = (V, E)$ be a hypergraph and let $H|S$ be a quotient hypergraph, for S a system of representatives of a congruence partition of V into prime modules. Then:*

1. *given M a prime module of H, its image in S is a prime module of $H|S$,*
2. *given M a prime module of $H|S$, its inverse image is a prime module of H.*

Lemma 2.4 holds also in the case of arbitrary congruence partitions and modules. By the notion of a partitive tree for a hypergraph H, we obtain the notion of a *complete, primitive* and *linear* hypergraph as defined in the following.

Definition 2.7 *Let $H = (V, E)$ be a hypergraph. Then H is* complete *iff any subset of V is a module of H, H is* primitive *iff H admits only trivial modules and $|V| > 2$, while H is* linear *iff there exists a linear order of the set V, with $|V| > 2$, such that a subset A of V is a module iff all elements in A are consecutive in such a linear order.*

Just as for graphs and k-structures two results can be easily proved [5]: the first one is that $H = (V, E)$ is a complete hypergraph if and only if E is the empty set or E contains all k-subsets

of V, the second one is that a hypergraph cannot be linear. In fact, along the same lines of the proof of Lemma 2.3, it is easy to show that the family of overlapping modules is closed w.r.t. the symmetric difference. As pointed out in [14], for general relational structure, this property allows to prove that hypergraphs cannot be linear.

3 An algorithm for the modular decomposition

In this section we propose an algorithm for the modular decomposition of hypergraphs of dimension 3 that will be the basis for the general algorithm for hypergraphs of fixed dimension k, $k > 3$, described in section 5.

We now give some preliminary definitions and properties used in the algorithm.

Definition 3.1 *Let $H = (V, E)$ be a hypergraph of dimension k and let $X \subseteq V$. Then X is a* clique *of H iff every k-subset of X is a hyperedge of H.*

As in definition 3.1, a set $X \subseteq V$, is called an *independent set* of H iff no k-subset of X is a hyperedge of H.

Given a hypergraph H, a module M excluding a vertex v of H, is simply a module of H not containing v.

Definition 3.2 *[4] Let X be a module of a hypergraph H. Then X is a* maximal module excluding v *iff $v \notin X$ and for each module Y of H such that $X \subset Y$, $v \in Y$.*

By $M(H, v)$ we denote the set of all maximal modules of H excluding v.

Lemma 3.1 *Let $H = (V, E)$ be a hypergraph and $v \in V$. Then $M(H, v)$ is a partition of the set $V - \{v\}$.*

Proof: Assume that a vertex w, with $w \neq v$ is not in any member of $M(H, v)$. This is not possible, as $\{w\}$ is a module of H excluding v (in fact, $\{w\}$ is a trivial module of H). Assume now that there are two members of $M(H, v)$ that contain the same vertex w. Since these modules overlap, their union is a module containing w and excluding v, but this contradicts the fact that $M(H, v)$ contains maximal modules excluding v. □

Let $H = (V, E)$ be a 3-hypergraph. In the following, given a subset X of V, we will say that X *is split by the vertex v* in V, if it is distinguished by the set $\{v\}$ or by some set $\{v, v_1\}$, for $v_1 \in V - X$.

Then let P be a partition of a subset V_1 of V and v a vertex in V_1. Let C be the set of P containing v. By $\Pi(v, P)$ we denote the set consisting of $\{C\}$ and all maximal sets X which are contained in a set of $P - \{C\}$ and X is not split by the vertex v. Note that by Lemma 2.2, $\Pi(v, P)$ is a partition which is a refinement of P.

The set $\Pi(v, P)$ is computed by using the notion of underlying graph.

Definition 3.3 *Let $H = (V, E)$ be a 3-hypergraph, and let P be a partition of V. Let v be a vertex of H, with $v \in C$, for $C \in P$. The* underlying graph of H w.r.t. v and P, *denoted as $G_H(v, P)$, is the graph with vertex set $V - \{v\}$ and set of edges $E_1 = \{(v_1, v_2) : \{v_1, v_2, v\} \in E, v_1 \notin C$ or $v_2 \notin C\}$.*

Example 3.1 Let $H = (V, E)$ be a hypergraph of dimension 3, with $V = \{v_1, \ldots, v_6\}$ and $E = \{\{v_1, v_2\}, \{v_1, v_6\}, \{v_2, v_6\}, \{v_1, v_2, v_6\}, \{v_3, v_4\}, \{v_4, v_5\}, \{v_3, v_5\}, \{v_3, v_4, v_5\}\}$. Let us consider the partition $P = \{\{v_2\}, \{v_1, v_3, v_6\}, \{v_4, v_5\}\}$ of V. The *underlying graph* $G_H(v_2, P)$ has vertex set $V - \{v_2\}$ and edge set $\{(v_1, v_6)\}$.

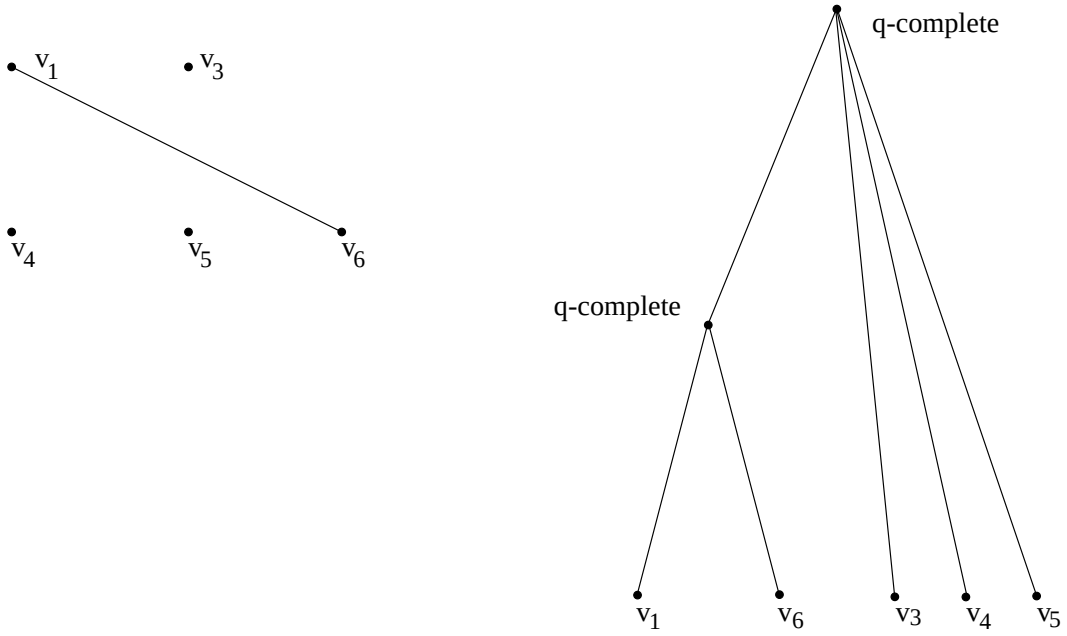

Figure 1: $G_H(v_2, P)$ and its modular decomposition tree

The general idea on which our algorithm for decomposing k-hypergraphs is based, derives by the following property that generalizes to hypergraphs a result shown in [4] for 2-structures. This result shows that every ancestor of a given vertex v in the decomposition tree is related to the maximal modules excluding v.

Lemma 3.2 *Let $H = (V, E)$ be a hypergraph, and let $v \in V$. Let U be a proper ancestor of $\{v\}$ in the decomposition tree of H, and let Z be the child of U in the tree containing v. Then*

1. *if U is q-complete, then the union of all children of U, except Z, is a maximal module of H excluding v;*
2. *if U is q-primitive, then each child X of U, with $X \neq Z$ is a maximal module of H excluding v.*

The set of all maximal modules of H excluding v is obtained by applying condition 1. and 2. to all ancestors of $\{v\}$ in the decomposition tree of H.

Proof: Let w be a generic vertex of H, $w \neq v$, let U be the least common ancestor of vertices v and w in the decomposition tree, and let Z and W be the children of U containing v and w, respectively. By Lemma 2.1, each module of H is obtained as a union of some children of a

node in the partitive tree of H. Then the maximal module of H excluding v and containing w is obtained as a union of children of U, such that this union contains W and excludes Z. But, if U is q-primitive, by definition of q-primitive it follows that W is the union that gives the maximal module excluding v and containing w. Similarly, if U is q-complete, then the union of all children of U, except Z is the maximal module excluding v and including w. □

Since the property described in Lemma 3.2 gives a necessary condition for a set to be an ancestor (q-primitive or q-complete) of a set of modules of H excluding a given vertex, we derive a procedure to determine a common ancestor for some nodes in the partitive tree, once that all maximal modules excluding each vertex in V are known. The use of such a property to compute the partitive tree of a structure is common with the approach proposed in [4]. But, we develop a different method to construct the partitive tree of hypergraphs. Moreover, new ideas are required here to compute the family of maximal modules excluding a vertex.

We describe our new approach for the case of 3-hypergraphs, while in section 5 we will generalize the approach proposed here to the case of k-hypergraphs.

The algorithm *ConstructTree(H)* that computes the decomposition tree of a hypergraph H of dimension 3 uses three basic procedures:

CreateNode computes an internal node X of the partitive tree of H such that X has only leaves as children,

Maxmodule(H, v) computes the maximal modules of H that exclude the vertex v, using the procedure *Partition*.

Partition(v, P) on input a vertex v and a partition P of a subset V_1 of V, Partition(v, P) computes $\Pi(v, P)$.

The hypergraph $H = (V, E)$ is represented with an adjacency matrix. Then, given X a k-subset of V, it is possible to determine in constant time if X is a hyperedge of H. All underlying graphs are represented by means of adjacency lists: this is required to obtain the linear time complexity bounds of the algorithms described in [11, 2, 3].

Let V be the vertex set of the hypergraph $H = (V, E)$ and let $W \subseteq V$: by $EM(W)$ we denote the set of all maximal modules of $H|W$ excluding a vertex v, for some $v \in W$; then to each set $X \in EM(W)$ is associated the set $\mathrm{Excl}(X)$ of the vertices v in W such that $X \in M(H, v)$. Moreover we assume that each vertex $v \in W$ has a pointer to the sets in $EM(W)$ containing v. By $\mathrm{Minc}(v)$ we denote the sets of minimum cardinality among all sets in $EM(W)$ that contain v. Finally, by $\mathrm{Minexcl}(v)$ we denote the set of vertices w in W such that $w \in \mathrm{Excl}(X)$, for some $X \in \mathrm{Minc}(v)$, i.e. $\mathrm{Minexcl}(v) = \bigcup_{X \in \mathrm{Minc}(v)} \mathrm{Excl}(X)$.

Assume that $|V| = n$. Note that each vertex v is in at most n members of $EM(V)$ and that $EM(V)$ contains at most n^2 sets. These facts imply that, given $EM(V)$, it is possible to compute in time $O(n^2)$ the sets $\mathrm{Minc}(v)$, $\mathrm{Minexcl}(v)$.

We now state the different basic procedures of the algorithm, and prove the correctness of each one.

Partition(v, P)

Require: A vertex v of the hypergraph $H = (V, E)$ of dimension 3 and a partition P of a subset V_1 of V

H_1 := subhypergraph induced by V_1 in H
C := the set in P to which v belongs
$G_{H_1}(v, P)$:= the underlying graph w.r.t. v and P
T := modular decomposition tree of $G_{H_1}(v, P)$
Compute a postorder traversal of the tree T, merging in a single set X all those leaves that are children of an internal node labeled q-complete, are contained in the same set in $P - \{C\}$ and either:

- X is a clique of $G_{H_1}(v, P)$ and each vertex in X is 2-adjacent to v in H_1, or
- X is an independent set of $G_{H_1}(v, P)$ and each element in X is not 2-adjacent to v in H_1

P_1 := the partition of the set $V_1 - C$ obtained in the postorder traversal of T
Return: $P_1 \cup \{C\}$

Example 3.2 Let $G_H(v_2, P)$ be the underlying graph of example 3.1. The set $\{v_1, v_6\}$ is a module and a clique of $G_H(v_2, P)$, while both v_1 and v_6 are 2-adjacent to v_2 in H. Consequently the partition returned by the call to Partition(v_2, P) is $\{\{v_1, v_6\}, \{v_2\}, \{v_3\}, \{v_4, v_5\}\}$.

Lemma 3.3 *Let $H = (V, E)$ be a 3-hypergraph, let P be a partition of V and $v \in V$. Let C be the set of P to which v belongs. Let X be a subset of V that is disjoint from C. Then X cannot be split by v iff X is a module of the underlying graph $G_H(v, P)$ and either:*

- *X is a clique in $G_H(v, P)$ and each vertex $x \in X$ is 2-adjacent to v in H, or*
- *X is an independent set in $G_H(v, P)$ and no vertex $x \in X$ is 2-adjacent to v in H.*

Proof: By definition of module and of underlying graph, it is immediate that a set $\{w, v\}$ cannot distinguish the set X of H, for $w \in V - X$, iff $\{w\}$ cannot distinguish X in $G_H(v, P)$, i.e. X is a module of $G_H(v, P)$.

Now, $\{v\}$ cannot distinguish X in H iff X is a clique (independent set) in $G_H(v, P)$ and each vertex $x \in X$ is (not) 2-adjacent to v in H. This fact follows from the definition of underlying graph and the fact that $\{v\}$ cannot distinguish X in H iff for every two 3-subsets A, B of V such that $A =_X B$, and $A - X = B - X = \{v\}$, then $A \in E$, $B \in E$. The Lemma directly follows.

□

Lemma 3.4 *Let $H = (V, E)$ be a 3-hypergraph, v a vertex of H and P a partition of a subset V_1 of V. Then Partition(v, P) computes $\Pi(v, P)$.*

Proof: Let H_1 be the subhypergraph of H induced by the set V_1. A set X of the final partition of $V_1 - C$ computed by Partition(v, P) is obtained by merging some children of a node labeled q-complete of the modular decomposition tree T of $G_{H_1}(v, P)$, while visiting the tree in postorder. Then, X is a module of the graph $G_{H_1}(v, P)$. Moreover, by the algorithm Partition, X is a clique (or an independent set) of the graph and consists of elements all 2-adjacent (or not 2-adjacent) to v in H_1. Then, by Lemma 3.3, X cannot be split by v in V_1. In the following, we prove that X is a maximal set which is contained in a set of $P - \{C\}$ and cannot be split by the vertex v in V_1. Assume to the contrary that there is a set Y which cannot be split by v and $X \subset Y$,

with Y contained in a set of $P - \{C\}$. By Lemma 3.3, Y is a module of $G_{H_1}(v, P)$ and must be a clique or an independent set of such a graph. It follows that each subset of Y is a module of $G_{H_1}(v, P)$. Hence, Y cannot contain a nontrivial prime module of the underlying graph. Then, by Lemmas 2.1, 2.3 Y cannot be obtained as union of internal nodes of the tree, but Y is union of leaves. It follows that all children of Y must be merged while visiting the tree. Since $Y \supset X$, X is not the largest set that can be merged by the algorithm, which is a contradiction. □

Maxmodule(H, v)
Require: A hypergraph $H = (V, E)$ of dimension 3 and a vertex $v \in V$.
  for each vertex $w \in V$ **do**
    Label w *old*
    Set$(w) := V$
  endfor
  Label v *new*
  $P := \{\{v\}, V - \{v\}\}$
  Repeat
    Let v_1 be a vertex labeled *new*
    $P_1 := \{A \in P : A \subseteq \text{Set}(v_1)\}$
    Set$(v_1) := C$, where $C \in P_1$, $v_1 \in C$
    $P_2 := \text{Partition}(v_1, P_1)$
    Label *old* the vertex v_1
    Label *new* all vertices contained in each set of P_2 which is not in P_1
    $P := (P - P_1) \cup P_2$
  until all vertices are labeled *old*
Return $(P - \{\{v\}\})$
; $P - \{\{v\}\}$ is the partition of $V - \{v\}$ into maximal modules of H excluding the vertex v

---

By Lemma 3.1, all maximal modules excluding v form, together with the set $\{v\}$, a partition of V. In fact, we will show in the following that Maxmodule must compute a partition of $V - \{v\}$ into modules.

Note that, inside a call to Maxmodule(H, v) the partitions that are given as arguments in different calls to Partition are not necessarily partitions of V, but they may be partitions of a subset of V: such partitions will be called *restricted partitions*. The partitions of V computed in Maxmodule will be called *complete partitions*. It is easy to note that Maxmodule constructs successive refinements of a complete partition.

In the following, we will say that a vertex v_1 *separates two vertices* v_2 *and* v_3 *in a partition* P_1 computed by Maxmodule(H, v), if there is a call to Partition(v_1, P_1) inside Maxmodule(H, v) that computes a refinement of P_1 in which v_2 and v_3 are in two distinct sets, while in P_1, v_2 and v_3 are in the same set.

Lemma 3.5 *Let $H = (V, E)$ be a 3-hypergraph and v a vertex in V. Let $v_1, v_2 \in V - \{v\}$ be two vertices contained in two distinct sets of the partition P computed by* Maxmodule(H, v). *Let v_3 be a vertex that separates v_1 and v_2 in the partition P_1, where $C_{1,2}$ is the set of P_1 containing both v_1 and v_2. Then $C_{1,2}$ does not contain any module of H excluding v_3 and containing $\{v_1, v_2\}$.*

Proof: Assume that $P_2 = \text{Partition}(v_3, P_1)$, where by Lemma 3.4 $P_2 = \Pi(v_3, P_1)$. Assume to the contrary that there is a module M of H excluding v_3, contained in $C_{1,2}$ and $M \supseteq \{v_1, v_2\}$. By definition of module, $\{v_3\}$ and any set $\{v', v_3\}$ cannot distinguish M. Consequently, by construction of $\Pi(v_3, P_1)$, a maximal set that cannot be split by v_3 and which is in $C_{1,2} \in P_1$ must contain M and hence $\{v_1, v_2\}$, which contradicts the fact that v_1, v_2 are in two distinct sets of P_2. □

Lemma 3.6 *Let $H = (V, E)$ be a 3-hypergraph and $v \in V$. Let $< P_1, \ldots, P_n >$ be the sequence of partitions of V computed in successive steps by Maxmodule(H, v). Then, for every i with $1 \leq i \leq n$, each module of H excluding v is contained in a set of P_i.*

Proof: We prove the Lemma by induction on the index i. Clearly, the Lemma holds for $i = 1$, as $P_1 = \{\{v\}, \{V - \{v\}\}\}$. Now, let C_k and C_j be two distinct sets of the partition P_i, for $i > 1$, and assume to the contrary that there is a module M of H excluding v such that M contains a vertex $v_1 \in C_k$ and a vertex $v_2 \in C_j$. Let P_l be a partition such that v_1, v_2 are in a same set $C_{1,2}$ of P_l, and there is a vertex $v_3 \notin C_{1,2}$ that separates v_1, v_2 in P_l. Clearly, $l < i$ and hence, by induction M is contained in $C_{1,2}$. By Lemma 3.5, M is not a module of H excluding the vertex v_3, and since $v_3 \notin M$, it follows that M cannot be a module of H, which is a contradiction. □

Given a partition P of a set V_1 of vertices of a hypergraph and a vertex $v \in V_1$, let us denote by $S(v, P)$ the set of P containing v. By $V(P)$ we denote the set $\cup_{A \in P} A$.

Lemma 3.7 *Let $< P_1, \ldots, P_n >$ be the sequence of successive restricted partitions computed in Maxmodule(H, v). Then, for each i, $1 \leq i \leq n$, either $V(P_i) = V$ or $V(P_i)$ is a set of some P_j, for $j < i$.*

Proof: By construction of Maxmodule(H, v), there is a vertex v_1 such that $P_i = \{A \in P_i' : A \subseteq \text{Set}(v_1)\}$, where P_i' is a complete partition and $\text{Set}(v_1) = V$ or $\text{Set}(v_1)$ is a set C in a partition P_k, with $k < i$. If $\text{Set}(v_1) = V$, then the Lemma holds. Thus assume that $\text{Set}(v_1) \subset V$ and let P_k' be the complete partition such that $P_k' \supseteq P_k$. Then P_i' must be a refinement of P_k', by construction of Maxmodule. It follows directly that $C = V(P_i)$. □

Lemma 3.8 *Let P be the partition computed by Maxmodule(H, v) and let X be a set of P. Let v_1 be a vertex in V such that $v_1 \notin X$. Then there is a call to Partition(v_1, P_1) such that X is contained in a set of P_1 and X is disjoint from $S(v_1, P_1)$.*

Proof:

Immediate by construction of Maxmodule. □

Lemma 3.9 *Let H be a 3-hypergraph and $v \in V$. Let P be the partition of $V - \{v\}$ computed by* Maxmodule(H, v). *Then each element of P is a module of H.*

Proof: Assume to the contrary that there is a set $X \in P$, with $X \subset V$, that is not a module of H. This means that there is a set $\{v_1\}$ or a set $\{v_1, v_2\}$ that distinguishes X in H. We will show that this fact leads to a contradiction, thus implying that X must be a module of H.

By Lemma 3.8, there is a call to Partition(v_1, P_1), for some P_1, such that X is contained in a set C of P_1 and $S(v_1, P_1)$ is disjoint from C. Assume that $P_2 = \text{Partition}(v_1, P_1)$; by Lemma 3.4 $P_2 = \Pi(v_1, P_1)$. We consider the following two cases:

(i) assume that $\{v_1\}$ distinguishes X. Since $P_2 = \Pi(v_1, P_1)$, there is no set of P_2 that contains X, which contradicts the fact that X is in the final partition P.

(ii) Assume that $\{v_1, v_2\}$ distinguishes X. If $v_2 \in V(P_1)$, since $P_2 = \Pi(v_1, P_1)$, P_2 does not have any set containing X, which is still a contradiction. Now, assume that $v_2 \notin V(P_1)$. By Lemma 3.8, there is a call to Partition(v_2, P_3) such that $S(v_2, P_3)$ is disjoint from the set of P_3 containing X. Let $P_4 =$ Partition(v_2, P_3). By Lemma 3.4, $P_4 = \Pi(v_2, P_3)$. If $v_1 \in V(P_3)$, we obtain a contradiction as P_4, and hence the final partition, cannot have any set containing X. Thus, assume $v_1 \notin V(P_3)$. By Lemma 3.7, $V(P_1)$ and $V(P_3)$ are sets of two restricted partitions P_i, P_j, moreover $V(P_1)$ and $V(P_3)$ both contain X. Let $P_i' \supseteq P_i$ and $P_j' \supseteq P_j$, for P_i', P_j' two complete partitions computed by Maxmodule(H, v). Clearly, by construction of Maxmodule, one of the two partitions P_i' and P_j' is a refinement of the other. Since X is contained in $V(P_1)$ and $V(P_3)$, $v_1 \in V(P_1), v_2 \notin V(P_1)$ and $v_1 \notin V(P_3), v_2 \in V(P_3)$, we obtain a contradiction. □

Lemmas 3.6 and 3.9 prove the following fact.

Corollary 3.1 *The procedure Maxmodule(H, v) constructs the set of all maximal modules excluding v.*

CreateNode(EM)
Require: the set EM of all maximal modules of H excluding a vertex v.

Let v be a node that minimizes |Minexcl(v)|
$X := \{v\} \cup$ Minexcl(v)
if each member Y of Minc(v) has |Excl(Y)| = 1 **then**

Label X q-complete

else

Label X q-primitive

endif

Return: X
; X is a node of the modular decomposition tree of H whose children are all leaves

---

Example 3.3 Let H be the hypergraph of example 3.1 whose partitive tree is given in Figure 2. Then,

$M(H, v_1) = \{\{v_2, v_6\}, \{v_3, v_4, v_5\}\}$, Minexcl$(v_1) = \{v_2, v_6\}$
$M(H, v_2) = \{\{v_1, v_6\}, \{v_3, v_4, v_5\}\}$, Minexcl$(v_2) = \{v_1, v_6\}$
$M(H, v_3) = \{\{v_1, v_2, v_6\}, \{v_4, v_5\}\}$, Minexcl$(v_3) = \{v_4, v_5\}$
$M(H, v_4) = \{\{v_1, v_2, v_6\}, \{v_3, v_5\}\}$, Minexcl$(v_4) = \{v_3, v_5\}$
$M(H, v_5) = \{\{v_1, v_2, v_6\}, \{v_3, v_4\}\}$, Minexcl$(v_5) = \{v_3, v_4\}$
$M(H, v_6) = \{\{v_1, v_2\}, \{v_3, v_4, v_5\}\}$, Minexcl$(v_6) = \{v_1, v_2\}$

Let us examine the result of a call to CreateNode($EM(V)$). Since Minexcl(v_1) is of minimum size among all sets Minexcl(v), for some $v \in V$, and Minc$(v_1) = \{\{v_1, v_6\}, \{v_1, v_2\}\}$, CreateNode($EM(V)$) computes $\{v_1, v_2, v_6\}$ as the parent of v_1 in the modular decomposition tree of H. Moreover such internal node is labeled q-complete, as |Excl$(\{v_1, v_6\})$| = |Excl$(\{v_1, v_2\})$| = 1.

In the following we will prove the correctness of CreateNode.

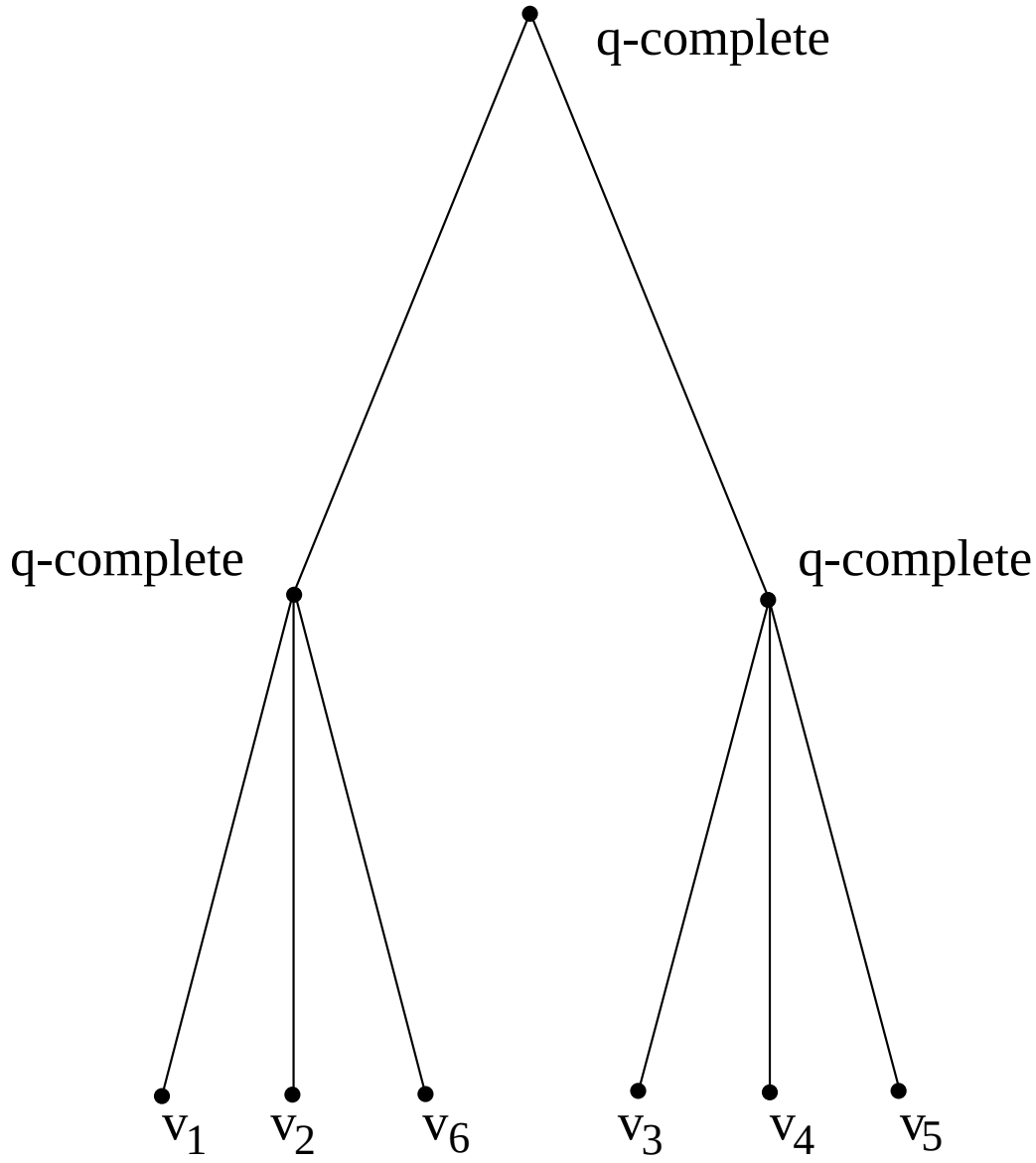

Figure 2: Partitive tree of H

Lemma 3.10 *Let $H = (V, E)$ be a hypergraph and v a vertex in V that minimizes $|Minexcl(v)|$. Let U be the parent of $\{v\}$ in the modular decomposition tree of H. Then, all children of U are leaves and $U = \{v\} \cup Minexcl(v)$.*

Proof: Assume to the contrary that a child of U is an internal node of the tree T and let Z_1 be a child of U of maximum cardinality. By Lemma 3.2, $U - Z_1$ contains the maximal module of H excluding each vertex in Z_1 and including v. Since Z_1 is of maximum cardinality, there is some set $Y \in \text{Minc}(v)$, such that $Y \subseteq U - Z_1$, and hence $\text{Minexcl}(v) \supseteq Z_1$. Now, let z be an element in Z_1. Clearly, by Lemma 3.2, for each $X \in \text{Minc}(z)$, $X \subseteq Z_1$ and hence $\text{Minexcl}(z) \subset Z_1$. But, this leads to contradict the fact that $|\text{ Minexcl}(v)|$ is minimum. Consequently, all children of U

must be leaves. By Lemma 3.2, each child $w \neq v$ of U is in Excl(Y), for some $Y \in$ Minc(v), and hence $U - \{v\} \subseteq$ Minexcl(v). Now, for each element z not in U, given Z a maximal module of H excluding z and including v, by Lemma 3.2, $Z \supseteq U$, consequently $Z \notin$ Minc(v) and $z \notin$ Minexcl(v). It follows that $U - \{v\} \supseteq$ Minexcl(v), thus obtaining that $U = \{v\} \cup$ Minexcl(v). □

By Lemma 3.10 and Lemma 3.2 it is immediate that:

Lemma 3.11 *Let $H = (V, E)$ be a hypergraph. The set X computed by CreateNode($EM(V)$) is an internal node of the decomposition tree T of H such that all children of X are leaves.*

Update(EM, W)
Require: the set EM of all maximal modules of $H = (V, E)$ excluding a vertex v, and W a system of representatives for a congruence partition of V into prime modules.

for each $Y \in EM$ **do**

$Y_1 :=$ Excl(Y) $\cap W$
$Y := Y \cap W$
Excl(Y) := Y_1

endfor

Modify consequently all sets Minc(v), Minexcl(v)
Return: EM
; EM is the set of all maximal modules excluding a vertex of the induced hypergraph $H|W$

By the quotient rule, we can show that Update($EM(V), W$) correctly computes the set $EM(W)$.

Lemma 3.12 *Let $H = (V, E)$ be a hypergraph. Let P be a congruence partition of V into prime modules and W a system of representatives for P. Then, given EM = Update($EM(V), W$), $EM = EM(W)$ and Update correctly computes the sets Excl(Y), for each $Y \in EM(W)$.*

Proof: Let M be a module in $EM(W)$. By Lemma 3.2, M is either a prime module of $H|W$ or $M = A - B$, for A, B two prime modules of $H|W$ and A is the parent of B in the partitive tree of $H|W$. First, assume that $M = A - B$. By the quotient rule, the inverse images $I(A)$, $I(B)$ of A, B are prime modules of H and $I(A)$ is the parent of $I(B)$ in the modular decomposition tree of H. Then, by Lemma 3.2, $I(A) - I(B) \in EM(V)$. Moreover, $(I(A) - I(B)) \cap W = M$. Now, assume that M is a prime module. As above we can show that $M = I(M) \cap W$ and $I(M) \in EM(V)$, for $I(M)$ the inverse image of M. It follows, by the previous two cases that $EM(W) \subseteq EM$. Now, let $M \in EM(V)$. By the Lemma 3.2 and by the quotient rule, as above we can show that $M \cap W \in EM(W)$. It follows that $EM \subseteq EM(W)$, which finally proves that $EM = EM(W)$.

We now show that given $X' \in EM(W)$, where $X' = X \cap W$, for $X \in EM(V)$, then Excl(X') =Excl(X) $\cap W$, which concludes the proof of the Lemma.

By Lemma 3.2, Excl(X) $= U - X$, where U is the smallest prime module of H containing X. Since, by the quotient rule $U \cap W$ is the smallest prime module of $H|W$ containing X', it follows by Lemma 3.2 that Excl(X') $= (U - X) \cap W =$ Excl(X) $\cap W$. □

Tree(EM, W)
Require: The set EM of all maximal modules of a hypergraph $H|W$, and a subset W of V,

if $|W| = 1$ **then**

Return the trivial tree consisting of the leaf W

else

$X :=$ CreateNode(EM)
Select a member x of X
$W' := W - X \cup \{x\}$
$EM :=$ Update(EM, W')
T:= Tree(EM, W')
Let Leaf be the leaf of T that corresponds to x in T
Label Leaf with the label of X
Add the members of X as children of Leaf
Add $X - \{x\}$ to Leaf and to all ancestors of Leaf in T

endif

Return: T
; T is the modular decomposition tree of $H|W$

Lemma 3.13 *Let $H = (V, E)$ be a hypergraph. Then Tree($EM(V), V$) computes the modular decomposition tree of H.*

Proof: The proof is by induction on the size of V. In fact, the tree T returned by Tree($EM(V), V$) is obtained by the tree T_1 computed by Tree(EM_1, W), where W is a system of representatives for the congruence partition P into prime modules, $P = \{X\} \cup \{\{x_i\} : x_i \in V - X\}$, and $X =$ CreateNode($EM(V)$). By Lemma 3.12, $EM_1 = EM(W)$, and hence by induction T_1 is the modular decomposition tree of the quotient hypergraph $H|W$. Then, by the quotient rule, and by Lemma 3.11 it is immediate to verify that the tree T must be the modular decomposition tree of H. □

We give in the following the algorithm for the modular decomposition of a hypergraph.

ConstructTree(H)
Require: A hypergraph $H = (V, E)$ of dimension 3.

for each vertex $v \in V$ **do**

Compute Maxmodule(v)

; compute $EM(V)$
endfor
for each set $Y \in EM(V)$ **do**

Compute Excl(Y)

endfor
for each vertex $v \in V$ **do**

Compute Minc(v), Minexcl(v)

endfor

$T :=$ Tree($EM(V), V$)

Return: T
; T is the modular decomposition tree of H

Corollary 3.2 *Let $H = (V, E)$ be a 3-hypergraph. Then the procedure ConstructTree(H) correctly computes the modular decomposition tree of H.*

4 On the complexity

Let $H = (V, E)$ be a 3-hypergraph, where $|V| = n$. Since all edges of the hypergraph are represented with an adjacency matrix it is possible to determine in constant time if a k-subset (hence a 3-subset) of V is a hyperedge of H or not.

Let P be a partition of a subset V_1 of V, and let H_1 be the subhypergraph of H induced by V_1. The complexity of the procedure Partition(v, P) is mainly determined by the construction of the underlying graph $G_{H_1}(v, P)$ and its modular decomposition. Now, the time required to construct the underlying graph $G_{H_1}(v, P)$ is at most $O(n+\nu(v, P))$, where $\nu(v, P)$ is the number of pairs of vertices in V_1 that are not both contained in C_v, where C_v is the set of P such that $v \in C_v$. In fact, it is required to determine if each 3-subset of vertices in P containing v and not contained in C_v is a hyperedge or not. Clearly, the number of edges of $G_{H_1}(v, P)$ is at most $\nu(v, P)$. Then, the decomposition tree of $G_{H_1}(v, P)$ can be computed in time $O(n + \nu(v, P))$ by using one of the linear time algorithms described in [11, 2, 3]. Note that, once the modular decomposition tree of $G_{H_1}(v, P)$ is computed inside the call to Partition(v, P), the cost of visiting such a tree to compute $\Pi_H(v, P)$ is $O(n)$. In fact, this step requires to find internal nodes of such a tree which are labeled q-complete. The children of such a node induce a quotient graph which is either a clique or an independent set: it is possible to determine in constant time which one of the two cases applies.

The procedure Maxmodule(H, v) may call Partition(v_i, P) when $v_i \in V$ is labeled *new*. A vertex v_i can be labeled *new* each time a call to Partition divides the set of P to which v_i belongs. So it is obvious that there can be at most n calls to Partition for a given vertex v_i. Let us determine the total cost of the calls to Partition (inside a call to Maxmodule(H, v)) with a given vertex w as argument.

Consider two calls to Partition, the first one with argument (w, P_1) and the second one with (w, P_2). By construction, the 3-subsets examined to construct the graphs G_1 and G_2, which are the underlying graphs of H w.r.t. w, P_1 and w, P_2, respectively, are distinct. In fact each edge of G_1 is such that at least one of the two endpoints is not in the set C_w of P_1 which contains w, while each edge of G_2 has both endpoints in C_w. It follows that the time bound for computing all underlying graphs (and hence their decomposition tree) w.r.t. a vertex w inside a call to Maxmodule(H, v) is: $\sum_{P_i \in \mathcal{P}} O(n + \nu(w, P_i)) = O(n^2)$, where $\mathcal{P}$ is the set of all partitions that are arguments with w of a call to Partition inside Maxmodule(H, v). Since Maxmodule(H, v) may call Partition with all vertices as arguments, the time complexity of Maxmodule(H, v) is $O(n^3)$.

Now let us determine the time required to compute ConstructTree(H). The cost of ConstructTree is given by the sum of the cost of computing $EM(V)$, the cost to initialize a pointer to each set in $EM(V)$ containing a vertex v and to determine Minc(v), Minexcl(v), for each vertex v, the cost to compute Excl(Y) for each set Y in $EM(V)$, the cost of Tree$(EM(V), V)$.

To compute $EM(V)$ requires $O(n^4)$ time, since for each vertex v in V, Maxmodule(H, v) must be determined.

Now, Tree$(EM(V), V)$ requires at most n recursive calls to itself, where each single call has a cost $O(n^3)$; the procedures Update, CreateNode require $O(n^3)$, $O(n^2)$ time, respectively. In fact, since there are at most $O(n^2)$ sets in $EM(V)$, it is easy to verify that the procedure Update requires at most $O(n^3)$. Moreover, computing Minc(v) and Minexcl(v) requires at most $O(n^2)$ time, for each vertex v. It follows that the time complexity for computing the modular decomposition tree of a hypergraph of dimension 3 is $O(n^4)$.

5 The algorithm for hypergraphs of dimension k

The ConstructTree(H) algorithm shows how to construct a partitive tree of a hypergraph of fixed dimension, once all its maximal modules excluding a given vertex v, for each vertex v, are known. In the previous section we have shown how to compute the set $M(H, v)$ for 3-hypergraphs, by using the procedure Partition. Thus, we can generalize the ConstructTree(H) procedure to the case of $k > 3$, by specifying a Maxmodule procedure for k-hypergraphs. Now, $M(H, v)$ is a partition of $V - \{v\}$ computed in successive steps, where at each step, such partition is refined into sets that cannot be distinguished from outside by $\{w\}$ or by every sets X, such that $w \in X$. In 3-hypergraphs, the sets that cannot be distinguished are determined by analyzing the 3-adjacency and 2-adjacency to some vertices. Note that the 3-adjacency to a given vertex is determined by analyzing the 2-adjacency relation in the underlying graph, more precisely by traversing the modular decomposition tree of such a graph. This approach suggests that the analysis of the k-adjacency relation for $k > 3$, can be reduced to the traversal of the modular decomposition tree of $(k-1)$-hypergraphs, hence we can decompose a hypergraph recursively on its dimension.

Definition 5.1 *Let $H = (V, E)$ be a k-hypergraph, P a partition of V and v a vertex of V such that $v \in C$, for $C \in P$. The* underlying hypergraph of H w.r.t. a vertex v and a partition P of V, *is the hypergraph $H(v, P) = (V', E')$, where $V' = V - \{v\}$ and $E' = \{X \subseteq V' : \{v\} \cup X \in E, X - C \neq \emptyset\}$.*

Clearly, $H(v, P)$ is a hypergraph of dimension $k-1$. The procedure Maxmodule(H, v) given in section 3 computes $M(H, v)$, for k-hypergraphs, whenever the procedure Partition correctly return the set $\Pi_k(v, P)$, where $\Pi_k(v, P)$, is a refinement $P_1 \cup \{C\}$ of P, for C the set of P containing v, where each set $A \in P_1$ is a maximal set that cannot be distinguished by any subset X of $V - A$, with $1 \leq |X| \leq k-1$, $v \in X$. In fact, all lemmas of section 3 used to prove the correctness of Maxmodule can be generalized to k-hypergraphs.

Now, the set $\Pi_k(v, P)$ is obtained by computing the decomposition tree of the underlying hypergraph $H(v, P)$ and each maximal set A that is a module of $H(v, P)$ and either is a clique of such underlying hypergraph, if A contains only vertices 2-adjacent to v in H, or an independent set of $H(v, P)$, if A contains no vertices 2-adjacent to v in H. In fact, if a set A contained in a set of $P - \{C\}$ is a clique of the hypergraph $H(v, P)$ and it consists of vertices 2-adjacent to v, then A cannot be distinguished by $\{v\}$: for each nonempty subset Y of A, $|Y| \leq k-1$, by definition of underlying hypergraph, $Y \cup \{v\}$ is an edge of H. Moreover, if A is a module of $H(v, P)$, then for every subset X of $V - A$, with $1 \leq |X| \leq k-2$, $X \cup \{v\}$ cannot distinguish A in H.

We now describe more in detail the procedure Partition; we assume that the procedure ConstructTree has one more argument, *DIM*, which is the dimension of the hypergraph.

Partition(v, P)

Require: A vertex $v \in V$ and a partition P of V_1, where $V_1 \subseteq V$ and $v \in C$, with $C \in P$

H_1 is the underlying hypergraph $H(v, P)$ w.r.t. v and P

$T :=$ ConstructTree$(H_1, DIM - 1)$

Compute a postorder traversal of the tree T, merging in a single set X all those leaves that are children of an internal node labeled q-complete, are contained in the same set in $P - \{C\}$ and either:

- X is a clique of the underlying hypergraph H_1 and each vertex in X is 2-adjacent to v in H, or
- X is an independent set of H_1 and each element in X is not 2-adjacent to v in H

$P_1 :=$ the partition of the set $V_1 - C$, obtained in the postorder traversal of T

Return: $P_1 \cup \{C\}$

Now, ConstructTree(H, DIM) calls n times Maxmodule and each call to Maxmodule uses at most n^2 calls to Partition, where Partition requires the decomposition of a hypergraph of dimension $DIM - 1$. It follows that decomposing a k-hypergraph can require the decomposition of at most n^3 hypergraphs of dimension $k - 1$. Then the time complexity of the algorithm is $O(n^{3k-5})$, which is obtained solving a simple recurrence equation.

Acknowledgment

This work has been supported by MURST 40% *Algoritmi e strutture di calcolo* and ASMICS 2 N^o 6317. The authors are grateful to the anonymous referees for their helpful comments.

References

[1] P. Bonizzoni. A tight lower bound for primitivity in k-structures. In S. Abiteboul and E. Shamir, editors, *Automata Languages and Programming: 21st International Colloquium, ICALP'94*, volume 820 of *LNCS*, pages 556–567, Berlin, 1994. Springer-Verlag.

[2] A. Cournier and M. Habib. A new linear algorithm for modular decomposition. In S. Tison, editor, *Trees in algebra and programming: 19th international colloquium, CAAP'94*, volume 787 of *LNCS*, pages 68–84, Berlin, 1994. Springer-Verlag.

[3] E. Dahlhaus, J. Gustedt, and R.M. McConnell. Efficient and practical modular decomposition. In *Proceedings of the Eighth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA'97*, pages 26–35, 1997.

[4] 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 2-structures and modular decomposition of graphs. *J. Algorithms*, 16:283–294, 1994.

[5] A. Ehrenfeucht and R.M. McConnell. A k-structure generalization of the theory of 2-structures. *Theoret. Comput. Sci.*, 132:209–227, 1994.

[6] A. Ehrenfeucht and G. Rozenberg. Theory of 2-structures, part 1: Clans, basic subclasses, and morphisms. *Theoret. Comput. Sci.*, 70:277–303, 1990.

[7] A. Ehrenfeucht and G. Rozenberg. Theory of 2-structures, part 2: Representations through labeled tree families. *Theoret. Comput. Sci.*, 70:305–342, 1990.

[8] A. Engelfriet, T. Harju, A. Proskurowsky, and G. Rozenberg. Characterization and complexity of uniformly nonprimitive labeled 2-structures. *Theoret. Comput. Sci.*, 154:247–282, 1996.

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

[10] R.M. McConnell. An $O(n^2)$ incremental algorithm for modular decomposition of graphs and 2-structures. *Algorithmica*, 14:229–248, 1995.

[11] R.M. McConnell and J.P. Spinrad. Linear-time modular decomposition of undirected graphs and efficient transitive orientation of comparability graphs. In *5th ACM-SIAM Symposium on Discrete Algorithms, SODA'94*, pages 536–545, 1994.

[12] R.H. Möhring. Algorithmic aspects of comparability graphs and interval graphs. In I. Rival, editor, *Graphs and Orders*, pages 41–101. D. Reidel, Boston, 1985.

[13] 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, 1985/6.

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