Algorithmic Aspects of a Novel Modular Decomposition Theory

Binh-Minh Bui-Xuan, Michel Habib, Vincent Limouzy, Fabien de Montgolfier

▶ **To cite this version:**

Binh-Minh Bui-Xuan, Michel Habib, Vincent Limouzy, Fabien de Montgolfier. Algorithmic Aspects of a Novel Modular Decomposition Theory. 2006. hal-00111235v1

HAL Id: hal-00111235

https://hal.science/hal-00111235v1

Preprint submitted on 4 Nov 2006 (v1), last revised 20 Nov 2007 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Algorithmic Aspects of a Novel Modular Decomposition Theory

B.-M. Bui-Xuan [a] M. Habib [b] V. Limouzy [b] F. de Montgolfier [b]

[a] *LIRMM, CNRS and University Montpellier II, 161 rue Ada, 34392 Montpellier Cedex 5, France.* buixuan@lirmm.fr

[b] *LIAFA, CNRS and University Paris 7, 2 place Jussieu, Case 7014, 75251 Paris Cedex 05, France.* {habib,limouzy,fm}@liafa.jussieu.fr

Abstract

A new general decomposition theory inspired from modular graph decomposition is presented. This helps unifying modular decomposition on different structures, including (but not restricted to) graphs. Moreover, even in the case of graphs, the terminology "module" not only captures the classical graph modules but also allows to handle 2−connected components, star-cutsets, and other vertex subsets.

The main result is that most of the nice algorithmic tools developed for modular decomposition of graphs still apply efficiently on our generalisation of modules. Besides, when an essential axiom is satisfied, almost all the important properties can be retrieved. For this case, an algorithm given by Ehrenfeucht, Gabow, McConnell and Sullivan [16] is generalised and yields a very efficient solution to the associated decomposition problem.

hal-00111235, version 1 - 4 Nov 2006

1 Introduction

Modular decomposition has arisen in different contexts as a very natural operation on many discrete structures such as graphs, directed graphs, 2-structures, automaton, boolean functions, hypergraphs, and matroids. In graph theory, modular decomposition plays a central role. Not only modular graph decomposition yields a framework for the computation of all transitive orientations of a given comparability graph [19,21,28], but it also highly relates to common intervals of a set of permutations [2,6,14] and therefore has applications in bioinformatics. Besides, many graph classes such as cographs, P_4-sparse or P_4-tidy graphs are characterised by properties of their modules (see e.g. [4]). It is also worth noticing that well-known NP-hard problems such as colouring can be solved in polynomial, and often linear, time when the graph is "sufficiently" decomposable [30] using some application of the *divide and conquer*

paradigm. Finally, the decomposition is useful for graph drawing [32], compact encoding (e.g. with cographs [10] and $P4-$sparse graphs [26]), and precomputing for graph problems including recognition, decision, and combinatorics optimisations (see [30] or [4] for a survey). A central point of this theory relies on the decomposition theorem which presents a tree, so-called *modular decomposition tree*, as compact encoding of the family of modules of a graph. Then, computing efficiently this tree given the graph has been an important challenge of the past three decades [6,7,8,9,11,13,16,14,22,24,28,30].

On the other hand, several combinatorial algorithms are based on partition refinement techniques [31,23,24]. Many graph algorithms make intensive use of *vertex splitting*, the action of splitting parts according to the neighbourhood of a vertex. For instance, all known linear-time modular decomposition algorithms on graphs use this technique [6,8,11,13,16,24,23,28]. In bioinformatics also, the distinction of a set by an element, so-called *splitter*, seems to play an important role, e.g. in the efficient computation of the set of common intervals of two permutations [6,33].

An abstract notion of *splitter* is studied here and a formalism based on the concept of homogeneity is proposed. The resulting structures will be referred to as *homogeneous relations*. Our aim is a better understanding of existing modular decomposition algorithms by characterising the algebraic properties on which they rely. As a natural consequence, the new formalism unifies modular decomposition on graphs and on their common generalisations to directed graphs [27] and to $2-$structures [17]. Of course, the theory still applies on structures beyond the previous ones. Moreover, even in the case of graphs, our terminology "module" not only captures the classical graph modules but also allows to handle other vertex subsets, e.g. those similar to $2-$connected components, or to star-cutsets.

Our main result is that most of the nice algorithmic tools developed to compute the modular decomposition tree of a graph still apply efficiently in the general theory. For graph modules,to design efficient algorithms there actually are three main approaches, distinguishable by the use of properties of: the set of maximal modules excluding a vertex [16], a factoring permutation [6,8,22,24], or the visit order of some peculiar graph search such as the so-called LexBFS *lexicographic breadth-first search* [5,15,25]. Because of its specificity due to exotic graph searches, the use of the third approach in the new theory is forfeit. Still, we extend the two first approaches, and retrieve most of the common efficient computations.

However, as a consequence of their broadness, no obvious decomposition theorem, up to our knowledge, is available for arbitrary homogeneous relations, hence no modular decomposition tree necessarily is guaranteed. Indeed, though the modules of homogeneous relations inherit many interesting properties from

graph modules, they do not necessarily satisfy the following essential one. One can *shrink* a whole graph module M into one single vertex $m \in M$: if some vertex of M distinguishes two exterior vertices, then so does every vertex of M and so does m. Let us denote the property by the name of *modular quotient*. It actually is the basis of many divide-and-conquer paradigms derived from the modular graph decomposition framework, such as the computation of weighted maximal stable or clique set, and graph colouring [20,29]. This naturally motivates us to study homogeneous relations fulfilling the modular quotient property, hereafter denoted by *good homogeneous relations*. As expected, almost all important properties of modular graph decomposition, including the decomposition theorem, still hold for the latter relations. Eventually, we generalise an algorithm given in [16] to an $O(|X|^2)$ algorithm computing the modular decomposition tree of a given good homogeneous relation on X.

The paper is structured as follows. First the new combinatorial decomposition theory is detailed in Sections 2 and 3. Section 4 investigates the general algorithmic framework on arbitrary homogeneous relations. The subsequent Section 5 is devoted to good homogeneous relations. Finally, we close the paper with noteworthy outcomes.

2 Homogeneity, an abstraction of Adjacency

Throughout this section X is a finite set, and $\mathcal{P}(X)$ denotes the family of all subsets of X. A *reflectless* triple is $(x, y, z) \subseteq X^3$ with $x \neq y$ and $x \neq z$. This will be denoted by $(x|yz)$ instead of (x, y, z) since the first element plays a particular role. Let H be a relation over the reflectless triples of X. Given $x \in X$, we define H_x as the binary relation on $X \setminus \{x\}$ such that $H_x(y, z) \Leftrightarrow H(x|yz)$.

Definition 1 (Homogeneous Relation) *H is a* homogeneous relation on *X if, for all $x \in X$, H_x is an equivalence relation on $X \setminus \{x\}$ (i.e. it fulfills the symmetry, reflexivity and transitivity properties). Equivalently, such a relation can be seen as a mapping from each $x \in X$ to a partition of $X \setminus \{x\}$, namely the equivalence classes of H_x.*

Definition 2 (Module) *Let H be a homogeneous relation on X. A subset $M \subseteq X$ is a* module *of H if*

$$\forall m, m' \in M, \quad \forall x \in X \setminus M, \quad H(x|mm').$$

Remark: From the definition it is obvious that, given a module M, if $\neg H(x|mm')$ for some $m, m' \in M$ then $x \in M$.

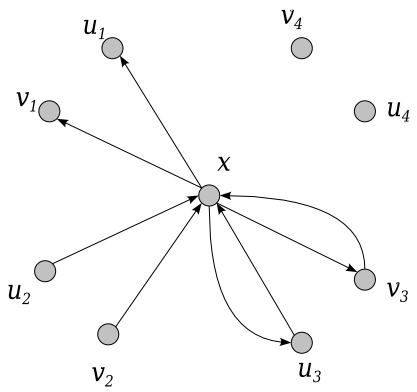

Fig. 1. The standard homogeneous relation H of this directed graph satisfies $H(x|u_iv_i)$ for all i, and $\neg H(x|u_iv_j)$ for all $i \neq j$.

If $\neg H(x|mm')$ we say that x *distinguishes* m from m', or x is a *splitter* of $\{m, m'\}$. A module M is *trivial* if $|M| \leq 1$ or $M = X$. The family of modules of H is denoted by $\mathcal{M}_H$, and $\mathcal{M}$ when no confusion occurs. H is *modular prime* if $\mathcal{M}_H$ is reduced to the trivial modules. For convenience, such a relation is also called *prime* when it clearly appears in the context that modules are involved. Homogeneity and distinction can be applied to graphs. Indeed, there is a natural homogeneous relation associated to graphs as follow.

Definition 3 (Standard Homogeneous Relation) *The* standard homogeneous relation $H(G)$ *of a directed graph* $G = (X, A)$ *is defined such that, for all* $x, u, v \in X$, $H(G)(x|uv)$ *is true if and only if the two following conditions hold:*
1. either both u *and* v *or none of them are in-neighbours of* x*, and*
2. either both u *and* v *or none of them are out-neighbours of* x*.*

Roughly, $H(x|uv)$ tells if x "sees" u and v the same way. Of course the above definition also holds for undirected graphs, tournaments, oriented graphs, and can also be extended to 2−structures (which roughly are edge-coloured complete directed graphs $G = (X, X^2)$, see e.g. [17] for further information). It follows straight from definition that

Proposition 1 *Let* G *be a graph, resp. tournament, oriented graph, directed graph,* 2−*structure. Modules of its standard homogeneous relation* $H(G)$ *are modules of* G *in the usual sense [19,30].*

Standard homogeneous relations highly refer to the notion of adjacency in graph theory. Notice that there are other homogeneous relations bound to a graph or to a 2−structure (e.g. in Section 6). Let us now give some first structural properties of homogeneous relations. Given $A \subseteq X$ one can define the induced relation $H[A]$ as H restricted to reflectless triples of A^3. If A is a module we have the following nice property:

Proposition 2 (Restriction) *Let H be a homogeneous relation, M a module of H, and $N \subseteq M$. Then, $N \in \mathcal{M}_{H[M]} \quad \Leftrightarrow \quad N \in \mathcal{M}_H$.*

Proof: That a module of H is a module of $H[M]$ is straightforward from definition. Conversely, if $N \subseteq M$ is not a module of H, then there is a splitter $s \in X \setminus N$ such that $\exists x, y \in N, \neg H(s|xy)$. However, s cannot belong to $X \setminus M$ since this would imply s is a splitter w.r.t. H of M. Therefore, $s \in M \setminus N$, and is a splitter w.r.t. $H[M]$ of N. Hence, N is not a module of $H[M]$. □

2.1 Lattice Structure

Let H be an arbitrary homogeneous relation over a finite set X. Let $\mathcal{M}$ denote the family of its modules. Two sets A and B *overlap* if $A \cap B$, $A \setminus B$ and $B \setminus A$ all are non-empty. It is denoted by $A \circledcirc B$.

Proposition 3 *For all $A, B \in \mathcal{M}$, if $A \circledcirc B$, $(A \cap B) \in \mathcal{M}$ and $(A \cup B) \in \mathcal{M}$.*

Proof: the fact that $A \cap B$ is a module is obvious. We use the transitivity of H_x for all $x \notin A \cup B$ to prove $(A \cup B) \in \mathcal{M}$. □

Proposition 4 *If $\mathcal{M}$ denotes the family of modules of a homogeneous relation, and $\mathcal{M}' = \mathcal{M} \cup \{\emptyset\}$, then $(\mathcal{M}', \subseteq)$ is a lattice.*

Proof: Since $\emptyset \in \mathcal{M}'$, and thanks to Proposition 3, the intersection of two members A and B belonging to $\mathcal{M}'$ belongs to $\mathcal{M}'$. It is the infimum of A and B, since any member of $\mathcal{M}'$ that is a subset of both A and B is a subset of $A \cap B$. Let $\mathcal{N}$ be the family of all members of $\mathcal{M}'$ containing both A and B. It is non-empty for X is a member. Since $\mathcal{M}'$ is closed under intersection, $\mathcal{N}$ admits a unique smallest member (w.r.t. inclusion), which is the intersection of all its members, and is the supremum of A and B. □

This lattice is a sublattice of the boolean lattice (hypercube) on X. Moreover, if we consider $A \in \mathcal{M}$ such that $|A| \geq 1$, and $\mathcal{M}(A) = \{M \in \mathcal{M}_H \text{ and } M \supseteq A\}$, then $(\mathcal{M}(A), \subseteq)$ is a distributive lattice.

2.2 Modules as Roots of a Submodular Function

Submodular functions are combinatorial objects with powerful potential (see e.g. [18]). Theorem 1 below enables the application of this theory to homogeneous relations: the modules of any such relation coincide with the roots of a function which satisfies the submodular inequality on intersecting subsets.

Definition 4 *A set function $\mu : \mathcal{P}(X) \to \mathbb{R}$ is submodular if, for all sets $A, B \in \mathcal{P}(X)$, $\mu(A) + \mu(B) \geq \mu(A \cup B) + \mu(A \cap B)$ (see e.g. [18]).*

Theorem 1 *Let H be a homogeneous relation on X. Let $s(A)$ be the function counting the number of splitters of a non-empty subset $A \subseteq X$. Then, s follows the submodular inequality on intersecting subsets:*

$$s(A) + s(B) \geq s(A \cup B) + s(A \cap B) \text{ for all } A \cap B \neq \emptyset.$$

Proof: If $A \subseteq B$ or $B \subseteq A$, the inequality is trivial. If $A \circledcirc B$ then $A \neq \emptyset$ and $B \neq \emptyset$. Let $\mathcal{S}_A$ denote the set of splitters of A. If $\{X_1, \ldots, X_k\}$ is a partition of X, we note $X = \{X_1, \ldots, X_k\}$. Obviously, $\mathcal{S}_{A\cap B} = \{\mathcal{S}_{A\cap B} \setminus B,\ \mathcal{S}_{A\cap B} \cap B\}$. As $\mathcal{S}_A \cap A = \emptyset$, the partition $\mathcal{S}_{A\cup B} = \{\mathcal{S}_{A\cup B} \setminus \mathcal{S}_A,\ \mathcal{S}_{A\cup B} \cap \mathcal{S}_A\}$ can be reduced to $\mathcal{S}_{A\cup B} = \{\mathcal{S}_{A\cup B} \setminus \mathcal{S}_A,\ \mathcal{S}_A \setminus (A \cup B)\}$. Similarly, $\mathcal{S}_B = \{\mathcal{S}_B \setminus \mathcal{S}_{A\cap B},\ \mathcal{S}_{A\cap B} \setminus B\}$. Finally, $\mathcal{S}_A = \{\mathcal{S}_A \setminus B,\ (\mathcal{S}_A \cap B) \setminus \mathcal{S}_{A\cap B},\ (\mathcal{S}_A \cap B) \cap \mathcal{S}_{A\cap B}\}$ can be reduced to $\mathcal{S}_A = \{\mathcal{S}_A \setminus (A \cup B),\ (\mathcal{S}_A \cap B) \setminus \mathcal{S}_{A\cap B},\ \mathcal{S}_{A\cap B} \cap B\}$. Hence,

$$|\mathcal{S}_A| + |\mathcal{S}_B| - |\mathcal{S}_{A\cup B}| - |\mathcal{S}_{A\cap B}| = |(\mathcal{S}_A \cap B) \setminus \mathcal{S}_{A\cap B}| + |\mathcal{S}_B \setminus \mathcal{S}_{A\cap B}| - |\mathcal{S}_{A\cup B} \setminus \mathcal{S}_A|.$$

To achieve proving the theorem, we prove that $\mathcal{S}_{A\cup B} \setminus \mathcal{S}_A \subseteq \mathcal{S}_B \setminus \mathcal{S}_{A\cap B}$. Indeed, let $s \in \mathcal{S}_{A\cup B} \setminus \mathcal{S}_A$. Then, $s \notin A \cup B$ and $H(s|xy)$ for all $x, y \in A$. Now, suppose that $s \notin \mathcal{S}_B$. Since s does not belong to B, we deduce $H(s|xy)$ for all $x, y \in B$. Furthermore, as A and B overlap and thanks to the transitivity of H, we deduce $H(s|xy)$ for all $x, y \in A \cup B$ and $s \notin A \cup B$, which is by definition $s \notin \mathcal{S}_{A\cup B}$. Contradiction. Finally, supposing $s \in \mathcal{S}_{A\cap B}$ would imply $s \in \mathcal{S}_A$. □

In [33] a (restricted) version of this theorem is proved, and this submodularity property is used to propose a very nice algorithm which computes the set of common intervals of a set of permutations. This approach was generalised for modules of standard homogeneous relations of undirected graphs in [6]. It would be interesting to consider this idea on arbitrary homogeneous relations.

2.3 Strong Modules and Primality

In an arbitrary family $\mathcal{F}$ of subsets of X, a member $A \in \mathcal{F}$ is *strong* if it does not overlap any other member $B \in \mathcal{F}$. Those which are not strong are *weak*. If they belong to the family, X and the singletons $\{x\}$ $(x \in X)$ form the *trivial* strong members of $\mathcal{F}$. Otherwise we extend $\mathcal{F}$ with the trivial strong members.

The set inclusion orders the strong members of $\mathcal{F}$ into a tree, hereafter denoted by the *generalised decomposition tree of* $\mathcal{F}$. This could be seen as a quick proof that, in $\mathcal{F}$, there are at most $2|X| - 1$ strong members, and at most $|X| - 2$ non-trivial ones, since the tree has $|X|$ leaves and no degree 1 internal nodes. When $\mathcal{F}$ is *weakly partitive* (see definition in Section 3), this tree plays an

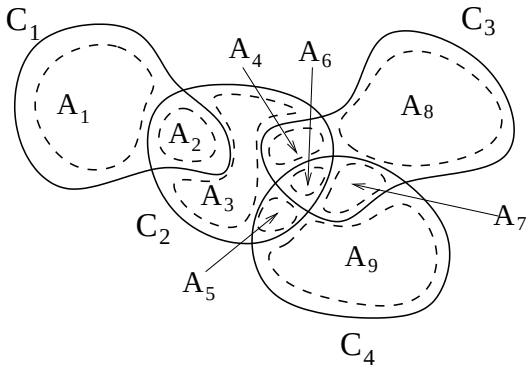

Fig. 2. The atoms $A_1, \ldots, A_9$ of the overlap class $\mathcal{C} = \{C_1, C_2, C_3, C_4\}$.

important role since it is an exact coding in $O(|X|)$ space of the possibly $2^{|X|}$ members of the family. It is then called the *decomposition tree of* $\mathcal{F}$.

The *parent* of a (possibly weak) member $M \in \mathcal{F}$ is the smallest strong member M_P properly containing M, and M is said to be a *child* of M_P. For instance, if M is strong then M_P is its parent in the generalised decomposition tree. A strong member is *prime* if all its children are strong, and *brittle* otherwise.

An *overlap class of* $\mathcal{F}$ is an equivalence class of the transitive closure of the overlap relation $\owedge$ on $\mathcal{F}$. Such a class is *trivial* if it contains only one member $A \in \mathcal{F}$. Then A is by definition a strong member of $\mathcal{F}$. The *support* of an overlap class $\mathcal{C} = \{C_1, \ldots, C_k\}$ is defined as $S(\mathcal{C}) = C_1 \cup \cdots \cup C_k$. An *atom* of the overlap class $\mathcal{C}$ is a maximal subset of $S(\mathcal{C})$ that does not overlap any C_i $(1 \leq i \leq k)$ (an illustration is given in Fig. 2). Notice that the atoms form a partition of $S(\mathcal{C})$. Besides, an atom of an overlap class belongs to the class if and only if this class is trivial. Furthermore, the support, resp. an atom, of an overlap class belongs to the family $\mathcal{F}$ if and only if it is a strong member of $\mathcal{F}$. Of course, a support, resp. an atom, does not necessarily belong to $\mathcal{F}$. However, in a *weakly partitive* family (see Section 3), all atoms and supports of overlap classes will belong by definition of partitivity to $\mathcal{F}$, hence are strong members of the family. It is an elementary result of finite set theory that

Proposition 5 *The following holds for any family $\mathcal{F}$ of subsets of a finite set X satisfying the closure under union of overlapping members.*
1. $A \subseteq X$ is a prime strong member of $\mathcal{F}$ if and only if $\{A\}$ is a trivial overlap class of $\mathcal{F}$.
2. $A \subseteq X$ is a brittle strong member of $\mathcal{F}$ if and only if it is the support of some non-trivial overlap class $\mathcal{C}_A$ of $\mathcal{F}$. In this case, weak children of A coincide with members of $\mathcal{C}_A$.

Of course we apply all these notions to the family of modules of a homogeneous relation H. Let $Z(x, y)$ be the largest module of H containing x but y. $Z(x, y)$ is well defined since it is the union of all modules containing x but y, which is

a module thanks to Proposition 3, and is non-empty because $\{x\}$ is one such module. Let $\mathcal{Z}(H)$ be the family

$$\mathcal{Z}(H) = \{Z(x,y) \mid \ x,y \in X \ \wedge \ x \neq y\}.$$

Notice that $\mathcal{Z}(H)$ is not necessarily closed under union of overlapping members. An example of such $\mathcal{Z}(H)$ is as follows. If $X = \{a,b,c\}$, $H(a|bc)$, $H(b|ac)$, and $H(c|ab)$, then $\{a,b\} \in \mathcal{Z}(H)$, $\{a,c\} \in \mathcal{Z}(H)$, however $X \notin \mathcal{Z}(H)$.

Theorem 2 *All support and atoms of $\mathcal{Z}(H)$ that are modules of H are strong modules. A non-trivial strong module of H either is the support or an atom of some overlap class of $\mathcal{Z}(H)$.*

Proof: Let us prove the first claim of the theorem.

(1) The support of an overlap class of $\mathcal{Z}(H)$ is a module, since the family of modules is closed under the union of overlapping members (Proposition 3). If the support S of a given overlap class $\mathcal{C}$ is overlapped by another module, then it is overlapped by a module $A \notin \mathcal{Z}(H)$. Let x be an element of $A \setminus S$ and y an element of $S \setminus A$. $Z(x,y)$ contains A but not $\{y\}$ and thus overlaps S, so it must overlap at least one member of $\mathcal{C}$ and thus $Z(x,y) \in C$, a contradiction since $x \notin S$. So the support of an overlap class is a strong module.

(2) Let A be an atom of a given overlap class $\mathcal{C}$ of $\mathcal{Z}(H)$. If A is included in at least two members of $\mathcal{C}$, then A is exactly the intersection of all members of $\mathcal{C}$ which include A. Since the family of modules is closed under intersection of overlapping members (Proposition 3), A is a module. Notice that if A is included in only one member of $\mathcal{C}$, it may fail to be a module. Let us suppose that A is a module, and that it is overlapped by another module. Then it is overlapped by a module $B \notin \mathcal{Z}(H)$. Let x be an element of $B \setminus A$ and y an element of $A \setminus B$. $Z(x,y)$ contains B but not $\{y\}$ and thus overlaps A, so it overlaps all elements of $\mathcal{C}$ which include A and thus $Z(x,y) \in \mathcal{C}$, a contradiction since no atom may be overlapped by a member of overlap the class. So the atoms of an overlap class who are modules are strong modules.

Now, let us prove that if M is a non-trivial strong module then it is the support or an atom of some overlap class. We shall distinguish three cases. Let M_P be the strong parent of M (which exists since $M \neq X$).

(1) M is prime and M_P is prime. Then for all $x \in M$ and all $y \in M_P \setminus M$, $M = Z(x,y)$. As M is a strong module, it alone form a trivial overlap class of $\mathcal{Z}(H)$ and is equal to its support and to its unique atom.

(2) M is prime and M_P is brittle. Then for all $x \in M$ and all $y \in M_P \setminus M$, M is included in $Z(x,y)$. Notice that these $Z(x,y)$ belong all to a same

overlap class $\mathcal{C}$ of $\mathcal{Z}(H)$. Since M is a strong module of H, $M \subseteq S(\mathcal{C})$ cannot overlap any member of $\mathcal{C}$. Moreover, for all $M \subsetneq N \subseteq S(\mathcal{C})$, N would overlap $Z(x, y)$ with $x \in M$ and $y \in N \setminus M$. Hence, M is by definition an atom of $\mathcal{C}$.

(3) M is brittle. It is easy to notice that M has $k \geq 3$ strong children $M_1, \ldots, M_k$. Let us pick an element x_i in each M_i. Then for all i and j we consider $Z(x_i, x_j)$. Not all of them are strong modules (otherwise, M would be prime). Let us consider the *overlap graph* of these modules (the vertices are the modules, and there is an edge between overlapping modules). Each connected component of this graph is an overlap class. According to the first sentence of the theorem, the support of each overlap class is a strong module. If there are two overlap classes, the support of at least one is a strong module that is strictly between M and its sons M_i in the inclusion tree, since the overlap graph has at least one edge, a contradiction. So there must be only one overlap class, whose support is exactly M.

□

For an arbitrary homogeneous relation, Theorem 2 gives the basis for an $O(|X|^3)$ strong modules computing time, which is depicted in Section 4.5.

2.4 Particular Homogeneous Relations

We now survey some classes of homogeneous relations defined by added axioms, which, in practice, frequently occurs. For instance, the class of standard homogeneous relations (see Definition 3) has very specific properties, leading to efficient decomposition algorithms (see Section 5).

Definition 5 *A homogeneous relation H is said to be*

- ***weakly graphic*** *if $H(y|xz) \wedge H(z|xy) \Rightarrow H(x|yz)$ for all $x, y, z \in X$;*
- ***weakly digraphic*** *if $H(s|xy) \wedge H(t|xy) \wedge H(y|sx) \wedge H(y|tx) \Rightarrow H(x|st)$ for all $x, y, s, t \in X$;*
- ***modular quotient*** *if $H(x|st) \Leftrightarrow H(y|st)$ for all module M of H, for all $x, y \in M$, and $s, t \notin M$.*

Proposition 6 *A weakly graphic homogeneous relation is weakly digraphic. There are weakly graphic homogeneous relations that are not modular quotient. There are modular quotient homogeneous relations that are not weakly digraphic, hence not weakly graphic.*

Proof: If H is weakly graphic, $H(s|xy)$ and $H(y|sx)$ imply $H(x|sy)$. Likewise, $H(t|xy)$ and $H(y|tx)$ imply $H(x|ty)$. Then, $H(x|st)$ by transitivity of

H_x. Hence, H is weakly digraphic. Besides, let K be defined over $X_K = \{x, y, s, t\}$ as $K_x = \{\{y\}, \{s\}, \{t\}\}$, $K_y = \{\{x\}, \{s, t\}\}$, $K_s = \{\{x, y\}, \{t\}\}$, and $K_t = \{\{x, y\}, \{s\}\}$. Then, K is weakly graphic (exhaustive checking on all triplets) but modular quotient ($\neg H(x|st)$ and $H(y|st)$ for the module $\{x, y\}$). Finally, let L be defined over $X_L = \{x, y, s, t, z\}$ as $K_x = \{\{s\}\}, \{t, y, z\}$, $K_y = \{\{s, t, x\}, \{z\}\}$, $K_s = \{\{x, y\}, \{t, z\}\}$, $K_t = \{\{x, y\}, \{s, z\}\}$. and $K_z = \{\{x\}, \{s, t, y\}\}$. Then, L is modular quotient (L is modular prime) but weakly digraphic (x, y, s, t form a counter example). □

The modular quotient property plays an important role in modular decomposition algorithmics. Indeed, if H is modular quotient, elements in a module M of H uniformly perceive a set A not intersecting M: if one element of M distinguishes A then so do all. This, combined with the definition of a module, allows to shrink M into a single element, the quotient by M, or to pick a *representative element* from the module. Recursiveness can therefore be used when dealing with modules. The modular quotient and restriction (Proposition 2) properties were first used in modular decomposition of graphs and are useful for algorithmics [30]. In this paper, these relations will be qualified as *good homogeneous relations*, and Section 5 is devoted to their study. Let the *congruence w.r.t.* H of an element $x \in X$ stand for the number of equivalence classes of the relation H_x. Then, the *local congruence* of H is the maximum congruence of all elements of X. Homogeneous relations of congruence 2 plays a special role in graph theory as they include the class of standard homogeneous relations of undirected graphs and tournaments (see next section). Furthermore, those relations satisfy the following nice property.

Proposition 7 *Any weakly graphic homogeneous relation H of local congruence 2 is modular quotient.*

Proof: Suppose H weakly graphic and not modular quotient. Then, there exist x, y, s, t pairwise distinct such that $\{x, y\}$ is a module, $H(x|st)$, and $\neg H(y|st)$. Let us prove that we have both $\neg H(y|xs)$ and $\neg H(y|xt)$. Indeed, suppose w.l.o.g. that $H(y|xs)$. Then, the transitivity of H_y implies $\neg H(y|xt)$ (for we already have $\neg H(y|st)$). Besides, since $\{x, y\}$ is a module, $H(s|xy)$. The weakly graphic property implies $H(x|sy)$, and the transitivity of H_x yields $H(x|ty)$. But then we would have $H(x|ty)$, $H(t|xy)$ ($\{x, y\}$ module), and $\neg H(y|xt)$, which is a contradiction with being weakly graphic. Hence, $\neg H(y|st)$, $\neg H(y|xs)$, $\neg H(y|xt)$, and the congruence of y is at least 3. □

2.5 Standard Homogeneous Relations

Given a (directed) graph, and more generally a 2−structure, the associated standard homogeneous relation is defined in Definition 3. Such relations are

peculiar and satisfy the following fundamental property.

Proposition 8 *The standard homogeneous relation of a 2−structure is modular quotient. In particular, this result holds for graphs, tournaments, oriented graphs, and directed graphs.*

Proposition 8 has important algorithmic implications that will be detailed in Section 5. Now, the name of *weakly graphic* and *weakly digraphic* homogeneous relations used in the previous section is motivated by Proposition 9 below. A *symmetric* 2−structure refers to an edge-coloured clique (the clique is seen as an undirected graph, see e.g. [17] for further information).

Proposition 9 *The standard homogeneous relation of a directed graph, resp. a 2−structure, is weakly digraphic. The standard homogeneous relation of an undirected graph, resp. a symmetric 2−structure, is weakly graphic.*

We now investigate a converse question: given a homogeneous relation H over a finite set X, does it exist an undirected graph, or a tournament, admitting H as standard homogeneous relation? H is defined as a ***graphic*** homogeneous relation if its local congruence is atmost 2 and if $H[\{a, b, c\}]$ has exactly 0 or 2 elements of congruence 2 for all triple $\{a, b, c\}$. H is ***tournamental*** if its local congruence is atmost 2 and if $H[\{a, b, c\}]$ has exactly 1 or 3 elements of congruence 2 for all triple $\{a, b, c\}$.

Theorem 3 *H is the standard homogeneous relation of an undirected graph if and only if it is graphic. H is the standard homogeneous relation of a tournament if and only if it is tournamental.*

Proof: It is straightforward to check that the standard homogeneous relation of any graph, resp. tournament, is graphic, resp. tournamental. The converse for graphs can be proved as follows. Let H be a graphic homogeneous relation of local congruence atmost 2 over a finite set X, and $x \in X$. Let C_x be one of the equivalence classes of H_x. We define the matrix M as: $M(x, y) = 1$ if $y \in C_x$ and $M(x, y) = 0$ otherwise; for all $x' \neq x$, $M(x', y) = 1$ if $y \in C_{x'}$ and $M(x, y) = 0$ otherwise, where $C_{x'}$ is the equivalence class of $H_{x'}$ containing x. Suppose M not symmetric. Then, there exists $y \neq z$ both distinct to x such that $M(y, z) = 1$ and $M(z, y) = 0$. But then $H[\{x, y, z\}]$ would have exactly 1 or 3 elements of congruence 2. Therefore, M is an $\{0, 1\}$ symmetric matrix and can be seen as the adjacency matrix of some undirected graph G. It is then straightforward to verify that H is the standard homogeneous relation of G. The proof for tournaments is similar. We use the characterisation that the adjacency matrix of a tournament is a $\{-1, 1\}$ anti-symmetric matrix since there are no non-edges and no double arcs. □

Corollary 1 *It can be tested in $O(|X|^3)$ time if a homogeneous relation H admits a graph G or a tournament T such that $H(G) = H$ or $H(T) = H$.*

Proof: First check if all element x has congruence atmost 2. Then check for all triples the corresponding property of the restricted relation. □

Notice that, if a graphic, resp. tournamental, relation H is given as $|X|$ sets of equivalence classes of H_x (cf Section 4.1), then, the adjacency list representation of the corresponding graph, resp. tournament, can be built in $O(|X|^2)$ time. Indeed, for graphs one just has to decide which class of the first vertex $v \in X$ represents its neighbourhood. Then, for any other vertex u, the class containing v will be its neighbourhood if u is a neighbour of v, and its non-neighbourhood otherwise. Simply remove the "non-neighbourhood" classes (in $O(|X|)$ time each): the other class in each case is the vertex's adjacency list. A similar construction can be performed for tournaments in the same $O(|X|^2)$ worst case time.

Remark: Extending Theorem 3 to symmetric 2−structures is quite straightforward. It would be interesting to characterise the standard homogeneous relations of directed graphs, and 2−structures.

3 Partitivity and Decomposition Theorem

A generalisation of modular decomposition, known from [9], less general than homogeneous relations but more powerful, is the *partitives families*. The *symmetric difference* of two sets A and B, denoted by $A\Delta B$, is $(A \setminus B) \cup (B \setminus A)$.

Definition 6 *A family $\mathcal{F} \subseteq \mathcal{P}(X)$ is* weakly partitive *if it contains X and the singletons $\{x\}$ for all $x \in X$, and is closed under union, intersection and difference of overlapping members, i.e.*

$$A \in \mathcal{F} \ \wedge \ B \in \mathcal{F} \ \wedge \ A \mathbin{@} B \ \Rightarrow \ A \cap B \in \mathcal{F} \ \wedge \ A \cup B \in \mathcal{F} \ \wedge \ A \setminus B \in \mathcal{F}.$$

Furthermore a weakly partitive family $\mathcal{F}$ is partitive *if it is also closed under symmetric difference of overlapping members:*

$$A \in \mathcal{F} \ \wedge \ B \in \mathcal{F} \ \wedge \ A \mathbin{@} B \ \Rightarrow \ A\Delta B \in \mathcal{F}.$$

Let $\mathcal{F}$ be a weakly partitive family over X. As mentioned before, strong members of $\mathcal{F}$ can be ordered by inclusion into a tree, so-called generalised decomposition tree (see Section 2.3). In this tree, the child, under the usual parental notion in trees, of an internal node M is by definition a strong member of $\mathcal{F}$, which is also a strong child of the strong member $M \in \mathcal{F}$, in the sense of Section 2.3. Besides, a weak child of the node M will refer to the definition of Section 2.3. Let us define three types of strong members of $\mathcal{F}$, namely three types of nodes of the tree:

- *prime* nodes which have no weak children,

- *degenerate* nodes: any union of strong children of the node belongs to $\mathcal{F}$,
- *linear* nodes: there is an ordering of the strong children of the node such that a union of them belongs to $\mathcal{F}$ if and only if they follow consecutively in this ordering.

Theorem 4 ***[9]*** *In a partitive family, there are only prime and degenerate nodes. In a weakly partitive family, there are only prime, degenerate, and linear nodes.*

The generalised decomposition tree hence is an $O(|X|)$ space coding of the family: it is sufficient to type the nodes into complete, linear or prime, and to order the children of the linear nodes. It is then called the *decomposition tree* of the family. From this tree, all weak members of $\mathcal{F}$ can be outputted by making simple combinations of the strong children of brittle (degenerate or linear) nodes. Now, the following property states that modules of some homogeneous relations are a proper generalisation of (weakly) partitive families.

Proposition 10 *The modules of a weakly graphic, resp. weakly digraphic, homogeneous relation H form a partitive, resp. weakly partitive, family.*

Proof: Proposition 3 gives the closure by intersection and union of overlapping members. Let $A \in \mathcal{M}_H$ and $B \in \mathcal{M}_H$ be two overlapping modules of H. Suppose that there is a splitter s of $A \setminus B$: there are $x, y \in A \setminus B$ such that $\neg H(s|xy)$. Moreover, $s \in A \cap B$ otherwise it would be a splitter of A. Finally, since $A \circledcirc B$, there exists an element $t \in B \setminus A$. We have: $H(x|st)$ and $H(y|st)$ and $H(t|sx)$ and $H(t|sy)$ and $H(t|xy)$. In other words, H is not *weakly digraphic*. Hence, the family of modules of a weakly digraphic homogeneous relation is weakly partitive. Besides, suppose that z is a splitter of $A \Delta B$. Then, $z \in A \cap B$ and there exists $x \in A$ and $y \in B$ such that $\neg H(s|xy)$. Since $H(x|yz)$ and $H(y|xz)$, H is not *weakly graphic*. Hence, the family of modules of a weakly graphic homogeneous relation is partitive. □

As a result, the modules of a standard homogeneous relation form a weakly partitive family because such a relation always is weakly digraphic (cf Section 2.5). More generally, we will prove in Proposition 15 that the modules of any homogeneous relation that satisfies the modular quotient property (cf Section 2.4), so-called good homogeneous relation, form a weakly partitive family. Recall that a weakly digraphic homogeneous relation is not necessarily modular quotient (cf Proposition 6).

4 Algorithms for Arbitrary Homogeneous Relations

This section considers a given homogeneous relation H over a ground set X, and builds tools for computing the generalised modular decomposition tree of H. The best performance to compute this tree in the general case will be given in $O(|X|^3)$ time in Section 4.5. Notice that the decomposition Theorem 4 does not necessarily hold in this section.

4.1 Data Structures

According to Definition 1, a homogeneous relation H can be represented in $O(|X|^2)$ space by an $n \times n$ matrix A of values in $[\![1, n]\!]$ as follows. If $X = \{x_1, \ldots, x_n\}$, each equivalence class of the relation H_{x_i} will be assigned a distinct number from 1 to n. Then, the cell $A_{i,j}$ has value k if and only if x_j belongs to the equivalence class of H_{x_i} having the value k. This representation allows to test in $O(1)$ time whether $H(x_i|x_px_q)$ by checking if $A_{i,p} = A_{i,q}$. However, retrieving an equivalence class requires an $O(|X|)$ worst case time.

Another alternative is to use the list representation: each element $x \in X$ will be associated to a list of equivalence classes of the relation H_x. This list is allowed to ignore one class C_x among the equivalence classes of H_x, for instance the largest one. Thus, the total used space is $O(n+m)$, with $n = |X|$ and $m = \sum_{x \in X}(n - |C_x|)$. Though this representation allows access in $O(1)$ to an equivalence class of H_x for any element x, testing if $H(x|yz)$ would require $O(n - |C_x|)$.

Notice that for a homogeneous relation, it is straightforward to construct in $O(|X|^2)$ time a list representation given any matrix representation, and conversely.

N.B. Without further specification, all algorithms presented in this paper take matrix representations as input.

4.2 Smallest Module Containing a Subset

Let S be a non-empty subset of X. As $\mathcal{M}_H$ is closed under intersection, there is a unique smallest module containing S, the intersection of all modules containing S, denoted henceforth by $SM(S)$.

Theorem 5 *Algorithm 1 computes $SM(S)$ in $O(|X|.|SM(S)|) = O(|X|^2)$ time.*

Algorithm 1: Smallest Module containing S

Let x be an element of S, $M := \{x\}$ and $F := S \setminus \{x\}$
while F *is not empty* **do**
  pick an element y in F ; $F := F \setminus \{y\}$; $M := M \cup \{y\}$
  for *every element* $z \notin (M \cup F)$ **do**
    if $\neg H(z|x, y)$ **then** $F := F \cup \{z\}$
output M (now equals to $SM(S)$)

Proof: Time complexity is obvious as the **while** loop runs $|M| - 1$ times and the **for** loop $|X|$ times. The algorithm maintains the invariant that every splitter of M is in F. When M is replaced by $M \cup \{y\}$, using transitivity of the relation H_x, every splitter for $M \cup \{y\}$ either distinguishes x from y, or already is in F. The algorithm ends therefore on a module that contains S, and thus we have $SM(S) \subseteq M$. If $M \neq SM(S)$ let s be the first element of $M \setminus SM(S)$ added to F (eventually added to M). It distinguished two elements x and y from $SM(S)$, contradicting its homogeneity. So $SM(S) = M$. □

4.3 Maximal Modules Excluding an Element

Proposition 11 *Let x be an element of X. As $\mathcal{M}_H$ is closed under union of intersecting subsets, there is a unique partition of $X \setminus \{x\}$ into $S_1, \ldots, S_k$ such that every S_i is a module of H and is maximal w.r.t. inclusion in $\mathcal{M}_H$.*

We call $MaxM(x)$ this partition of maximal modules excluding x, and propose a partition refining algorithm for its computation. It is straight from definition that

Lemma 1 *Every module (especially the maximal ones) excluding x is included in some equivalence class of H_x.*

Therefore our algorithm starts with the partition $P = \{H_x^1, \ldots, H_x^k\}$ of equivalence classes of H_x. Then the partition is refined (parts are split) using the following rule. Let y be an element, called the *pivot*, and Y the part of P containing y.

Rule 1 *split every part A of P, except for Y, into $A \cap H_y^1$,...,$A \cap H_y^l$*

Notice that a part is broken if and only if its splitters include y.

Lemma 2 *Starting from the partition $P_0 = \{H_x^1..H_x^k\}$, the application of Rule 1 (for any pivot in any order) until no part can be actually split, produces $MaxM(x)$.*

Proof: The refining process ends when no pivot can split a part, i.e when every

part is a module. Let us suppose one of these modules M is not maximal w.r.t. inclusion: it is included in a module M', itself included in an equivalence class of H_x. Let us consider the pivot y that first broke M'. It cannot be out of M', as M' is module, nor within M', as a pivot does not break its own part. But M' was broken, contradiction. □

Let us now implement this lemma into an efficient algorithm. Let P_i be the partition after the ith application of Rule 1, y be a given vertex used as pivot, and Y_i the part of P_i containing y. We say that a part B of P_j descends from a part A of P_i if $i < j$ and $A \subset B$. Clearly, after y is chosen as pivot at step i, y does not distinguish any part of P_i excepted Y_i. If y is chosen as pivot after, at step $j > i$, y may only split the parts of P_{j-1} that descend from Y_i. Only these parts have to be examined for implementing Rule 1. But Y_j itself has not to be examined.

Let us suppose that, for a part A, we can split it in $O(|A|)$ time when applying Rule 1 with pivot y. Then the time spent at step j is $O(|Y_i| - |Y_j|)$, the sum of the size of the parts that descend from Y_i save Y_j. The time of all splittings with y as pivot is $O(|X|)$, leading to an $O(|X|^2)$ time complexity. This is implemented in Algorithm 2.

Algorithm 2: Maximal Modules excluding x

```
for every group G do
    for every part C of G do
        Compute the set Z of elements of in G but not in C
        for every element y of C do
            Partition Z according to the equivalence classes of H_y
            Add each partition set to the refining sets pool
Set the group boundaries to the parts boundaries (from P_{i-1} to P_i)
for each refining set R of the pool do
    Remove R from the pool and then refine P_i using R
```

Let us suppose that the parts are implemented as a linked list [24], and the new parts created after splitting an old one replace it and follow consecutively in the list. Then for each pivot y two pointers, one on the first part that descend from Y_i and the second to the last part, are enough to tell the parts to be examined. A simple sweep between the pointers, omitting Y_j, gives them. We call all classes descending from a previous one a *group*.

Now let us show how a part A can be split in $O(|A|)$ time. It is a classical trick of partition refining [31,24,23]. If the equivalence classes of H_y are numbered from 1 to k, then A can be bucket sorted in $O(|A| + k)$ time, then each bucket gives a new part that descend from A. If $|A| < k$, we have to renumber the used equivalence class of H_y from 1 to $k' \leq |A|$ before bucket sorting. A first sweep

on A marks the used equivalence class numbers. A second sweep unmarks an used number the first time it is seen, and replace it by the new number (an incremented counter) which is less than $|A|$. The vector of equivalence class numbers is initialised once in $O(k)$ time.

The last point is the ordering in which pivots are taken. Using all elements as pivots, and repeating this $|X|$ time, i.e. $|X|^2$ applications of Rule 1, is enough. A clever choice is to use y only if Y_i has been split, keeping a queue of "active" pivots. Let us define a measure that will be used later for complexity analysis.

Definition 7 *Let $\mathcal{P}$ be a partition of X. $Q(\mathcal{P})$ be the number of pairs $\{x, y\}$ such that x and y are not in the same part of $\mathcal{P}$.*

$Q(\mathcal{P})$ is between 1 (for the trivial partition $\{X\}$) and $\frac{|X|(|X|-1)}{2}$ (for the trivial partition into singletons).

Theorem 6 *$MaxM(x)$ can be computed in $\Theta(Q(MaxM(x))) = O(|X|^2)$ time.*

Proof: For the correctness proof, one just has to check that the algorithm above implements correctly Lemma 2. For the time complexity issues, notice that, for each pivot y, an element z is placed in Z only once. But it is placed in Z only if y and z are not in the same part. At each step, refining Z according to the equivalence classes of H_y, and then refining using all sets generated by y, takes $O(|Z|)$ time. Hence the algorithm takes $\Theta(Q(MaxM(x))$ time. □

4.4 Modular Primality test

We recall that H is *modular prime* if all its modules are trivial (see Section 2).

Theorem 7 *Let S be a non-empty subset of X. One can test in $O(|X|^2)$ time if H is modular prime.*

Proof: If $|X| \leq 2$ the answer is yes. Otherwise let x and y be two elements of X. In $O(|X|^2)$ time, the algorithm of Section 4.3 outputs the maximal modules excluding x. If one of them is non-trivial the answer is no. Otherwise all non-trivial modules contain x. In $O(|X|^2)$ time, the algorithm of Section 4.3 outputs the maximal modules excluding y. If one of them is non-trivial the answer is no. Otherwise all non-trivial modules contain x and y. Then, Algorithm 1 is used with $S = \{x, y\}$, in $O(|X|^2)$ time. The answer is yes if and only if $SM(\{x, y\}) = X$. □

4.5 Strong Modules Enumeration

Theorem 2 straightforwardly leads to an algorithm:

Theorem 8 *The strong modules of a homogeneous relation H on X can be computed in $O(|X|^3)$ time.*

Proof: First compute $MaxM(x)$ for all $x \in X$. All these sets together exactly form the family $\mathcal{Z}(H)$ defined in Theorem 2. It can be done in $O(|X|^3)$ time using the algorithm of Section 4.3 $|X|$ times. The size of this family (sum of the cardinals of every subsets) is $O(|X|^2)$ since they form $|X|$ partitions. Using Dahlhaus algorithm [12] the overlap components can be found in time linear on the size of the family, namely $O(|X|^2)$. According to Proposition 5 there are at most $|X|$ non-trivial overlap classes.

For each class it is easy to compute its support, and in $O(|X|^2)$ time easy to compute its atoms. For instance, consider the vector of parts of the overlap class containing a given element: the atoms are the elements with the same vector. Sorting the list of elements of the supports $O(|X|)$ times, one time per part, gives the elements with the same vector, thus the atoms.

Then the $O(|X|^2)$ supports and atoms must be sorted by inclusion order into the inclusion tree of the strong modules. It can be done in $O(|X|^3)$ time using the same sorting technique.

Eventually, "bad" atoms – those that are not strong modules – must be removed from the tree. According to first sentence of Theorem 2 the atoms that are modules are strong modules. We just have to perform $O(|X|)$ test on all nodes of the tree to test which of them are modules, which can be done in $O(|X|^2)$ time for each. □

4.6 Computation of the Generalised Modular Decomposition Tree given a Factoring Permutation

The notion of a factoring permutation in the case of graphs [7] was introduced to give an alternative for computing the modular decomposition tree of a graph [6,8,22,24] without the precomputing of maximal modules excluding some vertex x. It can be extended to homogeneous relation as follows.

Definition 8 (Factoring Permutation) *A* factoring permutation *of a homogeneous relation refers to a depth-first search's visit order of the leaves of the generalised modular decomposition tree of the relation.*

We here address the problem of, given a homogeneous relation H over a finite set X *and* a factoring permutation σ, computing the generalised modular decomposition tree of H. Of course the algorithm of Section 4.5 answers to this question. However, this section will depict a more efficient $O(|X|^2)$ solution. Actually, the name of factoring permutations mainly is motivated by the following characterisation. Without loss of generality, we denote the elements of X by $X = \{1, 2, \ldots, n\}$.

Proposition 12 *If σ is a factoring permutation of a homogeneous relation H over a finite set X, then every strong module of H is an interval of σ, namely it is of the form $\{\sigma(i), \sigma(i+1), \ldots, \sigma(j)\}$.*

Roughly, to compute all strong modules of H, it suffices to find among the intervals of σ those that are strong modules. Let I_{ij} denote the $\sigma-$interval $I_{ij} = \{\sigma(i), \sigma(i+1), \ldots, \sigma(j)\}$, and $\mathcal{S}_{ij}$ the splitter set of I_{ij}.

Proposition 13 $\mathcal{S}_{ij} = \mathcal{S}_{(i+1)j} \cup \mathcal{S}_{i(i+1)} \setminus \{\sigma(i)\}$.

Proof: that $\mathcal{S}_{(i+1)j} \cup \mathcal{S}_{i(i+1)} \setminus \{\sigma(i)\} \subseteq \mathcal{S}_{ij}$ is straight from definition of a splitter. Conversely, let $x \notin I_{ij}$ be such that $x \notin \mathcal{S}_{(i+1)j} \cup \mathcal{S}_{i(i+1)}$. Then, by the transitivity property of H_x, we obtain $H(x|yz)$ for all $y, z \in I_{ij}$, or in other words $x \notin \mathcal{S}_{ij}$. Hence, $\mathcal{S}_{ij} \subseteq \mathcal{S}_{(i+1)j} \cup \mathcal{S}_{i(i+1)}$. We use the fact that $\sigma(i) \notin \mathcal{S}_{ij}$ to conclude. □

This leads to a naive $O(|X|^3)$ solution to this section's question: for all interval I_{ij}, compute $\mathcal{S}_{i(i+1)}$, then $\mathcal{S}_{ij}$ using the previously computed $\mathcal{S}_{(i+1)j}$ and Proposition 13, eventually test if $\mathcal{S}_{ij}$ is empty. Let us now improve this idea. The interval I_{ij} is said to be *right-free* if it does not have a splitter on the right in the order σ, namely for all $k > j$, $\sigma(k)$ does not belong to $\mathcal{S}_{ij}$. Obviously, if I_{ij} is a strong module, I_{ij} is right-free. However, a much more interesting viewpoint is as follows. If I_{ij} is not right-free, then there will be no $i' \leq i$ such that $I_{i'j}$ is a strong module. Furthermore,

Proposition 14 *If $j_1 < \cdots < j_k$ are such that any I_{ij_q} $(1 \leq q \leq k)$ is right-free, then $\mathcal{S}_{ij_1} \subseteq \cdots \subseteq \mathcal{S}_{ij_k}$.*

Proof: All splitters of these intervals stand on the left of $\sigma(i)$ in the order σ. Hence, a splitter s of I_{ij_q} can not belong to $I_{ij_{q+1}}$, and will belong to $\mathcal{S}_{ij_{q+1}}$. □

Roughly, if in some iteration step $1 \leq i \leq n$, we only store some right-free intervals in a list $RF = (I_{ij_1}, \ldots, I_{ij_k})$, then all their corresponding splitters can easily be stored by differences in a list $\Delta S = (\Delta_{j_1}, \ldots, \Delta_{j_k})$, where $\Delta_{j_1} = \mathcal{S}_{ij_1}$ and $\Delta_{j_q} = \mathcal{S}_{ij_q} \setminus \mathcal{S}_{ij_{q-1}}$ $(q \geq 2)$. Under this convention, an interval I_{ij_q} of the collection is a module if and only if the q^{th} first members of ΔS are empty: $\Delta_{j_1} = \cdots = \Delta_{j_q} = \emptyset$.

From iteration step i to $(i-1)$, the collection of intervals will extend from $RF = (I_{ij_1}, \ldots, I_{ij_k})$ to $RF = (I_{(i-1)(i-1)}, I_{(i-1)j_1}, \ldots, I_{(i-1)j_k})$, and the list ΔS will be updated accordingly using Proposition 13. Also, if for some j_q, the extension of I_{ij_q} to $I_{(i-1)j_q}$ introduce a splitter $\sigma(k)$ such that $k > j_q$, then we remove this interval from RF for it no more is right-free and j_q will have no chance to be the right boundary of an unvisited strong module. We come to Algorithm 3. For convenience, each interval I_{ij_q} will be represented by its right boundary: we shall use $RF = (j_1, \ldots, j_k)$.

Algorithm 3: Generalised modular decomposition tree computation from a factoring permutation

---

Input: a homogeneous relation H over a finite set X, and a factoring permutation σ of H
Output: the generalised modular decomposition tree $\mathcal{T}$ of H
$RF \leftarrow ()$ and $\Delta S \leftarrow ()$ and $M \leftarrow \emptyset$
Create a dummy $y = \sigma(n+1)$ such that $H(s|xy)$ for all $s \in X$ and $x = \sigma(n)$
for $i = n$ *downto* 1 **do**
  $x \leftarrow \sigma(i)$ and $y \leftarrow \sigma(i+1)$
  if x *belongs to some member of* ΔS **then** remove it
  for *every* $s = \sigma(l)$ *with* $l < i$ *and* $\neg H(s|xy)$ **do**
    Add s to the first member of ΔS
  Find $s = \sigma(r)$ such that $\neg H(s|xy)$ and r maximum
  while *the first member* j *of* RF *satisfies* $j < r$ **do**
    Remove j from RF
    Let Fst and Snd be the first and second members of ΔS
    $Snd \leftarrow Snd \cup Fst$ and remove Fst from ΔS
  $RF \leftarrow (i, RF)$ and $\Delta S \leftarrow (\emptyset, \Delta S)$
  Let S, resp. j, be the first member of ΔS, resp. RF
  while $S = \emptyset$ **do**
    $M \leftarrow \{I_{ij}\} \cup M$
    Let S, resp. j, be its next member in ΔS, resp. RF
Remove the weak members of M
Construct $\mathcal{T}$, the inclusion order of members of M
Output $\mathcal{T}$

Invariant 1 *For all* $1 \leq i \leq n$, *let* $RF_i = (j_1, \ldots, j_k)$ *and* $\Delta S_i = (\Delta_1, \ldots, \Delta_k)$ *be the values of* RF *and* ΔS, *at the end of the first loop "**for**" in Algorithm 3. Then,*

- *for all member* j *of* RF_i, *the interval* I_{ij} *is right-free;*
- *for all* $1 \leq q \leq k$, $\mathcal{S}_{ij_q} = \Delta_1 \cup \cdots \cup \Delta_q$.

Algorithm 3 correctness directly follows from Invariant 1. As for complexity issues, it is quite straightforward to check that the computing time of all loops is in $O(n^2)$. After those loops, removing weak members of the list M can be

done in linear time on $|M|$ using the lexical member ordering of M: I_{ij} is before $I_{i'j'}$ in M if and only if $i \leq i'$ or $(i = i') \wedge (j \leq j')$. Notice that $|M|$ is less than the number of intervals of σ, which is in $O(n^2)$. Likewise, the time spent for ordering by inclusion the remaining members of M is linear on their number using the lexical property. Whence, the global computing time of Algorithm 3 is $O(n^2)$.

Theorem 9 *Given a factoring permutation σ of a homogeneous relation H over a finite set X, one can compute the generalised modular decomposition tree of H in $O(|X|^2)$ time.*

Factoring permutations can be get in $O(|X|^2)$ time in many cases, especially with standard homogeneous relations of

- inheritance graphs: a linear extension gives a factoring permutation [15];
- chordal graphs: the cardinality lexicographic breadth first search of the graph yields a factoring permutation [25];
- tournaments: a very simple partition refining algorithm (greedily choose x and partition the class containing x into $N^-(x), \{x\}, N^+(x)$) computes a factoring permutation [27];
- undirected graphs: more sophisticated algorithms run in $O(m \log n)$ time [24] or $O(n + m)$ time [22].

5 Good Homogeneous Relation Decomposition Algorithm

The good homogeneous relations refer to homogeneous relations fulfilling the modular quotient property (cf Section 2.4). For instance, standard homogeneous relations are good (Proposition 8). Their study is motivated by, among others, the following essential property.

Proposition 15 *The modules of a good homogeneous relation form a weakly partitive family.*

Proof: Proposition 3 gives the closure under intersection and union of overlapping members. We just have to check that, for two modules A and B of H, if $A \Cap B$ then $A \setminus B$ is a module. Let us suppose $A \setminus B$ has a splitter s. As A is a module, $s \in A \cap B$. Let x and y be two elements of $A \setminus B$ such that $\neg H(s|xy)$. As $A \Cap B$ there exists $t \in B \setminus A$. Since B is a module, the modular quotient property gives $\neg H(t|xy)$. But then A no more is a module. □

Let H be a good homogeneous relation over a finite set X. We address the problem of computing the modular decomposition tree of H, namely the inclusion order of strong modules of H. Here again, the algorithm of Section 4.5

can be used to give a solution to this question in $O(|X|^3)$ time. However, this section will give a more efficient $O(|X|^2)$ time solution, which is inspired from Ehrenfeucht et al. works [16].

Definition 9 *A* super-modular-decomposition-tree *(SMDT for short) of a good relation H on X is a tree*

- *where the leaf-set is X*
- *such that each node of the tree is a module of H*
- *such that each strong module of H is a node of the tree.*

The idea of the algorithm is to compute the left branch ("caterpillar") of a super modular decomposition tree of H, going from the root X to a arbitrary element x (see Fig. 3). Then, the algorithm recurses to compute the "legs" of the caterpillar, and appends them to the caterpillar. Algorithm 4 captures this idea. Eventually, the SMDT is cast into the modular decomposition tree.

Proposition 16 *Algorithm 4 computes a super modular decomposition tree*

Proof: Obviously all nodes output are modules. We just have to check that the tree contains all strong module. This is true indeed, because, for a strong module M, the first element $x \in M$ taken for the $x-$branch (see the definition below) at some recursive step outputs $M \in \mathcal{B}(x)$. The goodness of the the relation gives that, when the algorithm is applied recursively on $H[N]$ and when N is a module, the module M of $H[N]$ output is exactly the module M of H. □

We are now to give a solution to each step of Algorithm 4, and prove their correctness.

5.1 Strong modules containing x

Definition 10 ($x-$branch) *The $x-$branch of a good homogeneous relation H over X is the set $\mathcal{B}(x)$ of all strong modules containing the element $x \in X$, ordered by inclusion. In other words, it is the path from the root to leaf x of*

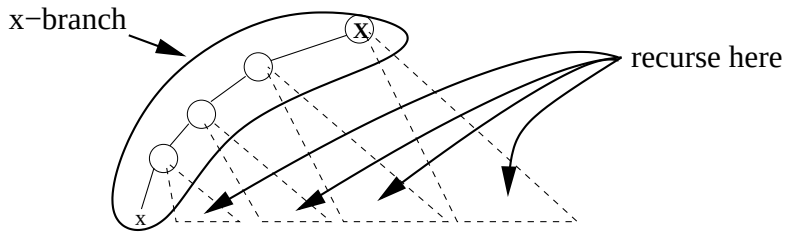

Fig. 3. A recursive approach to compute a super modular decomposition tree.

Algorithm 4: Super Modular Decomposition Tree of a Good Homogeneous Relation

Input: a good homogeneous relation H over a finite set X
Output: a super modular decomposition tree $\mathcal{T}$ of H
Let x be an element of X
Compute $MaxM(x)$, the maximal modules not containing x
Order $MaxM(x) = M_1..M_k$ such that for each $B_j \in \mathcal{B}$, $1 \leq j \leq l$, there exists $f(j)$ such that $B_j = \{x\} \uplus M_1 \uplus M_2 ... \uplus M_{f(j)}$
Initialise $\mathcal{T}$ to be the $x-$branch
for *every* M_i $(1 \leq i < k)$ **do**
  Compute recursively the modular decomposition tree $\mathcal{T}_i$ of $H[M_i]$
  Append $\mathcal{T}_i$ to the node B_j of $\mathcal{T}$ such that $j \leq i < f(j)$
Output $\mathcal{T}$

the modular decomposition tree of the relation.

The tool to construct the strong modules *containing* x is the construction of the maximal modules *not containing* x. Section 4.3 defined the set $MaxM(x) = \{M_1, \ldots, M_k\}$ of maximal modules excluding x, which is a partition of $X \setminus \{x\}$ by Proposition 11. Let us examine the relationship between $MaxM(x)$ and $\mathcal{B}(x)$

Proposition 17 *The modules of* $MaxM(x)$ *can be ordered from 1 to* k *in such a way that*

$$\text{for each } B \in \mathcal{B}(x), \text{ there exists } f \text{ such that } B = \{x\} \uplus M_1 \uplus M_2 ... \uplus M_f$$

Proof: For a module $B \in \mathcal{B}(x)$, the maximal modules not containing B form a partition of X. Of course each module of $MaxM(x)$ is included (or equal to) one of the modules of this partition. So a module of $MaxM(x)$ can not overlap a module $B \in \mathcal{B}(x)$. For constructing the ordering, just number the modules of $\mathcal{B}(x)$ from $B_0 = \{x\}$ to $B_l = X$ using inclusion order. Then number the modules of $MaxM(x)$ included in B_1 from 1 to $f(1)$, the modules of $MaxM(x)$ included in B_2 but not in B_1 from $f(1)+1$ to $f(2)$, and generally the modules included in B_i but not in B_{i-1} from $f(i-1)+1$ to $f(i)$. □

A consequence is that, if we order the elements of the $x-$branch from $B_0 = \{x\}$ to $B_l = X$ in increasing inclusion order, then for all $1 \leq i < l$ $B_{i+1} \setminus B_i$ is equal to some elements of $MaxM(x)$ that follow consecutively in the ordering above.

Proposition 18 *Let* $B_i \in \mathcal{B}(x)$ *be a non-leaf strong module containing* x, $C_i^1 ... C_i^{g(i)}$ *be its children in the modular decomposition tree and* j *such that* $C_i^j = B_{i+1}$ *is the child containing* x. *If* B_i *is* linear *we suppose the children are ordered according to the linear ordering.*

- *If* B_i *is* prime *then for all* $k \neq j$ $C_i^k \in MaxM(x)$
- *If* B_i *is* linear *then* $\bigcup_{k=1}^{k=j-1} C_i^k \in MaxM(x)$ *and* $\bigcup_{k=j+1}^{k=g(i)} C_i^k \in MaxM(x)$
- *If* B_i *is* complete *then* $\bigcup_{k \neq j} C_i^k \in MaxM(x)$

There are no more elements in $MaxM(x)$ *than those described above.*

Proof: Obvious □

5.2 Quotient relation

Now let us construct a quotient relation. For all $M_i \in MaxM(x)$ let $e_i \in M_i$ be a *representative element* of M_i (an arbitrary element). The *quotient relation* of H by $MaxM(x)$, denoted $H(x)$, is the relation

$$H(x) = H[\{x, e_1, \ldots, e_k\}]$$

Proposition 19 *The quotient relation of* H *by* $MaxM(x)$ *does not depend on the choice of the representative element of each* M_i

Proof: This is because the relation H is good. □

Proposition 20 *Every non-trivial module of* $H(x)$ *contains* x*.*

Proof: Suppose there is a non-trivial module $\cup_{i \in I}\{e_i\}$ of $H(x)$ that excludes x. Then, $|I| \geq 2$, and $\cup_{i \in I} M_i$ is a module of H that excludes x, larger than an element of $MaxM(x)$, a contradiction. □

For $M_i \in MaxM(x)$, let $S(M_i) \in \mathcal{B}(x)$ be the smallest module of $\mathcal{B}(x)$ containing M_i. Using the notations of Proposition 17 if $S(M_i) = B_j$ then $i \leq j < j(i)$. Proposition 18 gives the relationships between M_i and $S(M_i)$ with respect to $S(M_i)$ type (complete, linear or prime). We say that $e_i \in M_i$ is a *P-element* (resp. *L-element*, *C-element*) if $S(M_i)$ is prime (resp. linear, complete). Two elements $e_i \in M_i$ and $e_j \in M_j$ are *companion* one of each other if $S(M_i) = S(M_j)$. Proposition 18 tells that e_i has zero companion if $S(M_i)$ is complete, zero or one if $S(M_i)$ is linear and at least one if $S(M_i)$ is prime.

5.3 Forcing graph

Definition 11 (Forcing Graph) *Keeping the above notations, the directed* forcing graph $G(x) = (V, A)$ *is defined as* $V = \{e_1, \ldots, e_k\}$*; and an arc*

$(e_i, e_j) \in A$ *exists if and only if* $\neg H(e_j|x, e_i)$.

Proposition 21 *Let y be a vertex of $G(x)$ and $N^*(y)$ the descendants of y in $G(x)$ (including y itself). $N^*(y) \cup \{x\}$ is the smallest module of $H(x)$ containing y.*

Proof: First notice that all nontrivial modules of $H(x)$ contain x. Then, if the forcing graph has an edge (e_i, e_j) then any nontrivial module of $H(x)$ containing e_i also contains e_j. All descendants of y in $G(x)$ are thus in any module containing y (and x).

Now we shall prove that for any set A of vertices of $G(x)$ with no outgoing arc, $A \cup \{x\}$ is a module of $H(x)$. Indeed, for all $u \in A$ and all $v \notin A$ we have $H(v|x, u)$. As H is a transitive relation, then for all $u, u' \in A$ $H(v|u, u')$ and thus $A \cup \{x\}$ is a module. So $N^*(y) \cup \{x\}$ is a module of $H(x)$. □

Let C be a strongly connected component (SCC for short) of $G(x)$. The above proposition gives that all vertices of C are companions. Furthermore we have:

Proposition 22 *A non-trivial strongly connected components of $G(x)$ is formed by companion P-elements. Conversely a maximal set of companion P-elements is strongly connected.*

Proof: According to Proposition 18 there are no companions C-elements and at most two companions L-elements. But clearly there is no arc between them. So a SCC with at least two vertices contains companion P-elements. According to Proposition 21 if companions P elements were split into two (or more) SCC C and D, then there would be either a module of $H(x)$ containing C but not D, or a module of $H(x)$ containing D but not C. In both case, the smallest module of $H(x)$ containing $C \cup D$ can not be prime. □

Proposition 23 *Two companions L-elements are false twins (they are the same neighbourhood are there is no arc between them). Conversely the pairs of false twins are exactly the companions L-elements.*

Proof: Let e and e' be two companions L-elements. The smallest module M of $H(x)$ containing $\{e, e'\}$ is thus a linear module $\{e\} \cup M' \cup \{e'\}$ where M' is the strong module son of M is the modular decomposition tree of $H(x)$. Of course $x \in M'$. Both $\{e\} \cup M'$ and $M' \cup \{e'\}$ are modules, and the descendants of e are exactly the descendants of e' and are M', according to Proposition 21. Furthermore since H is good, e and e' are twins. □

According to the Propositions 21, 22 and 23 we have:

Proposition 24 *Any linear extension (topological sort) of $G(x)$ orders $MaxM(x)$*

into the ordering of Proposition 17.

Proposition 25 *The $x-$branch of H can be computed in $O(Q(MaxM(x)))$ time*

Proof: Remind that $Q(\mathcal{P})$ is the number of pairs $\{x, y\}$ whose vertices are not in the same part of a partition $\mathcal{P}$ (Definition 7). Let k be the number of parts of $MaxM(x)$. obviously $k^2 = O(Q(MaxM(x)))$ and $Q(MaxM(x)) = O(|X|^2)$. The algorithm is

- The maximal modules $MaxM(x)$ excluding x can be computed in time $O(Q(MaxM(x)))$, according to Theorem 6, using the algorithm of Section 4.3.
- Then, the vertices of the forcing graph are determined arbitrarily: for all $1 \leq i \leq k$ let $e_i \in M_i$.
- Then, constructing the forcing graph $G(x)$ in $O(k^2)$ time is obvious
- Then the topological sort $G(x)$ in $O(k^2)$ time is also easy.
- At least Proposition 17 tells how the ordering of $MaxM(x)$ allow to construct $\mathcal{B}(x)$. Notice that all companion vertices appear consecutively in the topological sort and are all regrouped to form $B_{i+1} \setminus B_i$.

□

We thus have:

Theorem 10 *Algorithm 4 computes a super modular decomposition tree in $O(|X|^2)$ time.*

Proof: This is a direct application of Propositions 16 and 25. We just have to show that the sum of all $O(Q(MaxM(x)))$ time computations is $O(|X|^2)$. This is true because $Q(MaxM(x))$ is the number of pairs $\{x, y\}$ belonging to two elements of $MaxM(x)$. As the algorithm is recursively launched on a module of $MaxM(x)$, each pair $\{x, y\}$ is counted once, in the recursive call of its least common ancestor of the SMDT finally output. □

5.4 Testing for weak modules and typing the nodes

Now, constructing recursively $x-$branches, we can build a super modular decomposition tree. This tree however is not the modular decomposition tree of H since:

- Its nodes are not typed complete, linear or prime
- It contains all strong modules but may also contain weak modules

Definition 12 *Let N be a node of a SMDT of H, with sons $S_1..S_k$, and $e_i \in S_i$ be an arbitrary element. The* quotient of H by N *is $H[\{e_1, ..., e_k\}]$.*

Proposition 26 *The quotient relation of a node N of a SMDT is either*

- *type P: with no non-trivial module*
- *type L: the elements can be linearly ordered in such a way that the modules of the quotient relations are exactly the interval of the relation*
- *type C: every subset is a module*

If N has k sons, a trivial $O(k^2)$ time algorithm can test the type and order the elements if needed. A classical (and easy to prove) result is that

Proposition 27 *Let T be a tree with n leaves and no node with only one child. Then*

$$\sum_{N \text{ node of } T} degree(N)^2 = O(n^2)$$

We can therefore perform quadratic-time computations on each node of a SMDT. A first application of Proposition 27 is

Proposition 28 *Let H be a good relation on X. It take $O(|X|^2)$ time to compute the quotient relations for all nodes of a SMDT of H.*

Proof: A bottom-up sweep, keeping one representative per child, builds the representatives. Each quotient relation can then be computed in size linear in its size, i.e. $O(k^2)$. □

A second application of Proposition 27 together with Proposition 26 gives that the typing of the nodes of a SMDT takes $O(|X|^2)$ time. Note that we abusively consider that weak modules have a type. Then we can look for the weak modules, and cast the SMDT into the genuine modular decomposition tree, using:

Proposition 29 *Let H be a good relation on X, and N be a node of a SMDT, and F be its father in the SMDT. F has another son A. If F is linear then take A that immediately precedes or follow N in the linear ordering. Take an element $a \in A$. If N is non-trivial it has at least two sons B and C. If N is linear then take B its first child and C its last child. Finally take $b \in B$ and $c \in C$.*

N is a weak module iff $\{a, b\}$ or $\{a, c\}$ is a module of $H[\{a, b, c\}]$.

Proof: If $\{a, b\}$ or $\{a, c\}$ is a module N is obviously weak. Conversely if N is weak then it is overlapped by a module N'. N and N' have thus the same father F in the modular decomposition tree. If F is complete, any union of a

son of F included in N plus one not included N' overlaps N. As $a \in N$ and $b \in (F \setminus N)$ we get the result. And if F is linear (any other arc is excluded), then either the first son of F included in N plus the preceding one in the linear order, overlaps N, and $\{a, b\}$ is a module; or the last son of F included in N plus the following one in the linear order, overlap N, and $\{b, c\}$ is a module. □

This proposition, together with a third application of Proposition 27, gives that the weak modules can be removed from a SMDT in $O(|X|^2)$ time. We finally have

Proposition 30 *A Super Modular Decomposition Tree of H can be cast into the modular decomposition tree of H in $O(|X|^2)$ time.*

And, together with Theorem 10 we have:

Theorem 11 *The modular decomposition tree of a good relation H over X can be built in $O(|X|^2)$ time.*

6 Outcomes

Let us examine in the sequel some of the applications of this homogeneity theory to modular decomposition of graphs and 2-structures, and to other graph relations.

From Proposition 9 and Section 3, the modules of an undirected graph and of a symmetric 2-structure form a partitive family, while the modules of a directed graph just form a weakly partitive family. All know properties of modular decomposition [30] can be derived from this result. An $O(n^2)$ modular decomposition algorithm can also be derived from Section 5 algorithm. It runs in optimal time for relations given as matrices (like an adjacency matrix), but it is less efficient than the existing algorithms for graphs stored using adjacency lists [6,8,11,13,16,24,23,28].

In a graph we can consider different homogeneous relations, for instance the relation *"there exists a path from vertex x to vertex y avoiding the vertex s"*, or a more general relation *"there exists a path from x to y avoiding the neighbourhood of s"*. It is easy to see that these two relations fulfill the basic axioms (symmetry, reflexivity and transitivity). In the first case, the strong modules form a partition (into the 2-vertex-connected components, minus the articulation points). The second relation is related to decomposition into star cutsets.

Another interesting relation is $D_k(s|xy)$ if $d(s, x) \leq k$ and $d(s, y) \leq k$, where

$d(x, y)$ denotes the distance between x and y. The case $k = 1$ corresponds to modular decomposition. It is worth investigating this general case.

7 Conclusion

We hope that this homogeneity theory will have many other applications and will be useful to decompose automata [1] and boolean functions [3]. Obviously, the algorithmic framework presented here can be optimised in each particular application, as it has been done for modular graph decomposition [6,8,11,13,16,24,23,28].

Acknowledgements: We are grateful to J. Gustedt for a helpful discussion and his interesting remarks.

References

[1] C. Allauzen and M. Mohri. Efficient algorithms for testing the twins property. *Journal of Automata, Languages and Combinatorics*, 8(2):117–144, 2003.

[2] Anne Bergeron, Cedric Chauve, Fabien de Montgolfier, and Mathieu Raffinot. Computing common intervals of permutations, with applications to modular decomposition of graphs. In *13th Annual European Symposium on Algorithms (ESA'05)*, volume 3669 of *LNCS*, pages 779–790, 2005.

[3] J. Bioch. The complexity of modular decomposition of boolean functions. *Discrete Applied Mathematics*, 149(1-3):1–13, 2005.

[4] A. Brandstadt, V.B. Le, and J.P. Spinrad. *Graph Classes: A Survey.* SIAM Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, 1999.

[5] A. Bretscher, D. G. Corneil, M. Habib, and C.Paul. A Simple linear time LexBFS cograph recognition algorithm. In *29th International Workshop on Graph-Theoretic Concepts in Computer Science (WG'03)*, volume 2880 of *LNCS*, pages 119–130, 2003.

[6] B.-M. Bui Xuan, M. Habib, and C. Paul. Revisiting T. Uno and M. Yagiura's Algorithm. In *16th International Symposium of Algorithms and Computation (ISAAC'05)*, volume 3827 of *LNCS*, pages 146–155, 2005.

[7] C. Capelle. *Décomposition de Graphes et Permutations Factorisantes.* PhD thesis, Université Montpellier II, 1997.

[8] C. Capelle, M. Habib, and F. de Montgolfier. Graph decomposition and factorizing permutations. *Discrete Mathematics and Theoretical Computer Science*, 5(1):55–70, 2002.

[9] M. Chein, M. Habib, and M.C. Maurer. Partitive hypergraphs. *Discrete Mathematics*, 37(1):35–50, 1981.

[10] D. G. Corneil, H. Lerchs, and L. K. Stewart. Complement reducible graphs. *Discrete Applied Mathematics*, 3:163–174, 1981.

[11] A. Cournier and M. Habib. A new linear algorithm for modular decomposition. In *Trees in algebra and programming (CAAP'94)*, volume 787 of *LNCS*, 1994.

[12] E. Dahlhaus. Parallel algorithms for hierarchical clustering, and applications to split decomposition and parity graph recognition. *Journal of Algorithms*, 36(2):205–240, 2000.

[13] E. Dahlhaus, J. Gustedt, and R.M. McConnell. Efficient and practical algorithms for sequential modular decomposition. *Journal of Algorithms*, 41(2):360–387, 2001.

[14] F. de Montgolfier. *Décomposition modulaire des graphes. Théorie, extensions et algorithmes*. PhD thesis, Université Montpellier II, 2003.

[15] R. Ducournau and M. Habib. La multiplicité de l'héritage dans les langages à objects. *Technique et Science Informatique*, 8(1):41–62, 1989.

[16] A. Ehrenfeucht, H. Gabow, R. McConnell, and S. Sullivan. An $O(n^2)$ Divide-and-Conquer Algorithm for the Prime Tree Decomposition of Two-Structures and Modular Decomposition of Graphs. *Journal of Algorithms*, 16:283–294, 1994.

[17] A. Ehrenfeucht and G. Rozenberg. Theory of 2-structures. *Theoretical Computer Science*, 3(70):277–342, 1990.

[18] S. Fujishige. *Submodular Functions and Optimization*. North-Holland, 1991.

[19] Tibor Gallai. Transitiv orientierbare Graphen. *Acta Mathematica Academiae Scientiarum Hungaricae*, 18:25–66, 1967.

[20] V. Giakoumakis and I. Rusu. Weighted parameters in $(P_5, \overline{P_5})$-free graphs. *Discrete Applied Mathematics*, 80:255–261, 1997.

[21] M.C. Golumbic. Algorithmic graph theory and perfect graphs. In *Annals of Discrete Mathematics*, volume 57. Elsevier, second edition, 2004.

[22] M. Habib, F. de Montgolfier, and C. Paul. A simple linear-time modular decomposition algorithm. In *9th Scandinavian Workshop on Algorithm Theory (SWAT'04)*, volume 3111 of *LNCS*, pages 187–198, 2004.

[23] M. Habib, R. McConnell, C. Paul, and L. Viennot. Lex-BFS and partition refinement, with applications to transitive orientation, interval graph recognition and consecutive ones testing. *Theoretical Computer Science*, 234:59–84, 2000.

[24] M. Habib, C. Paul, and L. Viennot. Partition refinement techniques: An interesting algorithmic tool kit. *International Journal of Foundations of Computer Science*, 10(2):147–170, 1999.

[25] W.-L. Hsu and T.-M. Ma. Substitution decomposition on chordal graphs and applications. In *2nd International Symposium on Algorithms (ISA'91)*, volume 557 of *LNCS*, pages 52–60, 1991.

[26] B. Jamison and S. Olariu. A unique tree representation for P_4-sparse graphs. *Discrete Applied Mathematics*, 35:115–129, 1992.

[27] R.M. McConnell and F. de Montgolfier. Linear-time modular decomposition of directed graphs. *Discrete Applied Mathematics*, 145(2):189–209, 2005.

[28] R.M. McConnell and J.P. Spinrad. Modular decomposition and transitive orientation. *Discrete Mathematics*, 201:189–241, 1999. Extended abstract at *SODA'94*.

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

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

[31] Robert Paige and Robert E. Tarjan. Three partition refinement algorithms. *SIAM Journal on Computing*, 16(6):973–989, 1987.

[32] Fwu-Shan Shieh and Carolyn L. McCreary. Directed graphs drawing by clan-based decomposition. In Franz-Josef Brandenburg, editor, *Graph Drawing*, LNCS, pages 472–482, 1995.

[33] T. Uno and M. Yagiura. Fast algorithms to enumerate all common intervals of two permutations. *Algorithmica*, 26(2):290–309, 2000.