

From Parallel Comparability Graph Recognition to Parallel Modular Decomposition

Michel Morvan and Laurent Viennot
LITP/IBP Université Paris 7 Denis Diderot
Case 7014, 2, place Jussieu
F-75251, Paris Cedex 05.
e-mail: {morvan,lavie}@litp.ibp.fr

August 5, 1995

Abstract

A parallelization of the algorithm of Golubic for recognizing comparability graphs is proposed for the concurrent parallel random access machine (CRCW PRAM). Parallel algorithms for finding a transitive orientation and the modular decomposition of any undirected graph are deduced from an extension of the theory of Golubic toward modular decomposition. The algorithms for recognizing and transitively orienting comparability graphs run in $O(\log n)$ time using δm processors and the modular decomposition algorithm runs in $O(\log n)$ time using n^3 processors (n, m and δ respectively denote the number of vertices, the number of edges and the maximal degree of the undirected input graph).

Topics: algorithms and data structures, graph theory, parallel algorithms.

1. Introduction

In his book, Golubic [Gol85] developed an algorithmical theory of comparability graphs. Recently his transitive orientation algorithm has been improved separately by Cournier and Habib [CH94] and by McConnell and Spinrad [MS94]. They both work on the so called modular decomposition. The relations between comparability graphs and modular decomposition have been first introduced by Gallai [Gal67].

In this paper we focus on the same problem from the point of view of parallel algorithmics. This first leads us to a natural parallelization of the recognition algorithm of Golubic. For the transitive orientation problem, the parallelization is not straightforward and it has been necessary to find new ways to solve the problem. Surprisingly, this leads us to reconsider the maximal multiplexes introduced by Golubic to count the number of transitive orientations and to make the link between his results and modular decomposition. We give a constructive characterization of the maximal multiplexes and we show how they are closely related to modular decomposition.

Section 2 introduces the notations. Before proposing a parallel recognition algorithm in Section 3, we present in a row in Section 4 all the results of Golubic, that we will use. In Section 5 we propose an algorithmic approach of the maximal multiplexes of Golubic in order to deduce in Section 6 a parallel transitive orientation algorithm. Section 7 is devoted to modular decomposition. It includes a new theory linking the maximal multiplexes to modular decomposition and a parallel modular decomposition algorithm.

2. Definitions

In this paper, we will always suppose that the graphs considered are loopless and finite. A graph $G = (V, \mathcal{E})$ with set of *vertices* V and set of *edges* $\mathcal{E} \subseteq V^2$ will be considered *undirected* if it is *symmetric* which means that $\mathcal{E}^{-1} = \mathcal{E}$ where $\mathcal{E}^{-1} = \{ (a, b) \mid (b, a) \in \mathcal{E} \}$. We note $\widehat{\mathcal{E}} = \mathcal{E} \cup \mathcal{E}^{-1}$. An edge (a, b) will be noted ab , and an undirected edge is denoted \widehat{ab} . The edge ba is called the *inverse* of the edge ab . By extension, \mathcal{E}^{-1} is called the *inverse set* of \mathcal{E} .

An undirected graph $G = (V, \mathcal{E})$ is a *comparability graph* if it can be obtained from a strict order (V, \mathcal{F}) on its vertices by symmetrization. Formally, $G = (V, \mathcal{E})$ is a comparability graph if there exists an *orientation* \mathcal{F} of G satisfying: $\mathcal{F} \cap \mathcal{F}^{-1} = \emptyset$, $\mathcal{E} = \mathcal{F} + \mathcal{F}^{-1}$ (disjoint union) and \mathcal{F} is *transitively closed*: $\mathcal{F}^2 \subseteq \mathcal{F}$ where $\mathcal{F}^2 = \{ ac \mid \text{there exists } b \in V, ab, bc \in \mathcal{F} \}$. Such an orientation is called a *transitive orientation* of G .

If $\mathcal{A} \subseteq \mathcal{E}$ is a set of edges of a graph $G = (V, \mathcal{E})$, then we note $V_{\mathcal{A}}$ the set of vertices *spanned* by \mathcal{A} : $V_{\mathcal{A}} = \{ a \mid \text{there exists } b \in V, ab \in \mathcal{A} \text{ or } ba \in \mathcal{A} \}$.

3. The Theory of Golumbic

In this section we recall theoretic results of Golumbic we need. Golumbic introduced them to count the number of transitive orientations of a given comparability graph. Our search for a parallel algorithm has naturally led us to the same structures with a different point of view that we will develop later.

Let $G = (V, \mathcal{E})$ be an undirected graph. a, b, c is a *triangle* if the edges \widehat{ab} , \widehat{bc} and \widehat{ca} are in \mathcal{E} . Our main interest will be in a special sort of triangles. On the contrary, the theory of Golumbic is based on the situation where one undirected edge is missing. Suppose $\widehat{ab}, \widehat{bc} \in \mathcal{E}$ and $\widehat{ac} \notin \mathcal{E}$. It is impossible to orient \widehat{ab} and \widehat{bc} with ab and bc because the transitive edge ac will be missing. In this case, Golumbic says that ab *directly forces* cb (and ba directly forces bc). This is denoted by $ab \sim cb$.

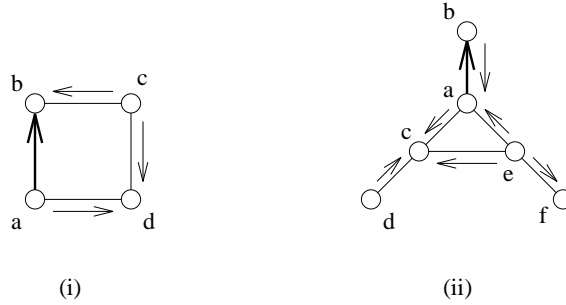


Figure 1. Examples of forcing. The arbitrary choice of ab for orienting \widehat{ab} forces the others indicated orientations. In (ii) we have $ab \sim ac \sim cd \sim ce \sim fe \sim ea \sim ba$. This brings a contradiction since we cannot orient \widehat{ab} with both ab and ba . Thus the graph is not a comparability graph.

This relation is symmetric and reflexive. The equivalence classes of its transitive closure \sim^* are called *implication classes* and form a partition of the edge set. It is said that ab *forces* $a'b'$ when $ab \sim^* a'b'$. Since $ab \sim^* a'b'$ if and only if $a'b' \sim^* ab$, if \mathcal{I} is an implication class, then so is \mathcal{I}^{-1} . $\widehat{\mathcal{I}}$ is then called a *color class*. The color classes form a partition of the set of undirected edges. See Figure 2 for an example.

Notice that in a comparability graph, no edge ab and its inverse ba are both in the same implication class (see Figure 1). Equivalently, each implication class \mathcal{I} is disjoint from \mathcal{I}^{-1} . The converse is also true as stated in the following theorem.

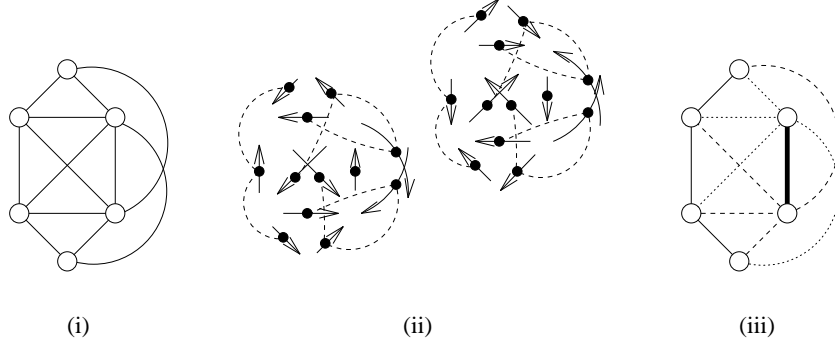


Figure 2. (i) An undirected graph $G = (V, \mathcal{E})$. (ii) The graph (\mathcal{E}, \sim) . The dash lines represent the direct forcing relation. (Remember that an undirected edge corresponds to two edges). The implication classes of G are the connected components of this graph. (iii) The color classes of G . Undirected edges with same line style are in the same color class.

Theorem 1 [Gol85].

- (i) If \mathcal{I} is an implication class, one of the two following cases occurs:
 - $\mathcal{I} \cap \mathcal{I}^{-1} = \emptyset$
 - $\mathcal{I} = \mathcal{I}^{-1}$.
- (ii) If $\mathcal{I} \cap \mathcal{I}^{-1} = \emptyset$ then $(V_{\widehat{\mathcal{I}}}, \widehat{\mathcal{I}})$ has exactly two transitive orientations: \mathcal{I} and \mathcal{I}^{-1} .
- (iii) An undirected graph G is a comparability graph if and only if $\mathcal{I} \cap \mathcal{I}^{-1} = \emptyset$ for each implication class \mathcal{I} .

Notice that the graph of Figure 2 is a comparability graph since each implication class is disjoint from its inverse.

Definition 2 [Gol85]. Let $G = (V, \mathcal{E})$ be an undirected graph. A complete subgraph $(V_{\mathcal{S}}, \mathcal{S})$ on $r + 1$ vertices is called a simplex of rank r if each undirected edge of \mathcal{S} is contained in a different color class of G . The multiplex generated by a simplex \mathcal{S} of rank r is defined to be the undirected subgraph $(V_{\mathcal{M}}, \mathcal{M})$, with $\mathcal{M} = \cup \mathcal{C}$, where the union is over all color classes \mathcal{C} satisfying $\mathcal{C} \cap \mathcal{S} \neq \emptyset$.

In regard to transitive orientation, a multiplex behaves like a simplex generating it which can be oriented by choosing a total order on its vertices. A simplex of rank 2 is called a *tricolored triangle*.

Golumbic has proved the following properties:

- (1) Simplices generating the same multiplex are isomorphic and hence have the same rank k . The multiplex they generate is said to have also rank k .
- (2) A multiplex is maximal (for inclusion) if and only if it is generated by a maximal simplex.
- (3) Two maximal multiplexes are either equal or disjoint.
- (4) If \mathcal{I} is an implication class such that $\mathcal{I} = \widehat{\mathcal{I}}$, then \mathcal{I} itself is a maximal multiplex of rank 1.

Theorem 3 [Gol85]. Let $G = (V, \mathcal{E})$ be an undirected graph, and let $\mathcal{E} = \mathcal{M}_1 + \dots + \mathcal{M}_k$, where each \mathcal{M}_i is a maximal multiplex of \mathcal{E} .

- (i) If \mathcal{F} is a transitive orientation of G , then $\mathcal{F} \cap \mathcal{M}_i$ is a transitive orientation of \mathcal{M}_i .
- (ii) If $\mathcal{F}_1, \dots, \mathcal{F}_k$ are transitive orientations of $\mathcal{M}_1, \dots, \mathcal{M}_k$ respectively, then $\mathcal{F}_1 + \dots + \mathcal{F}_k$ is a transitive orientation of G .
- (iii) If G is a comparability graph and $r_i = \text{rank } \mathcal{M}_i$, then the number of transitive orientations of G is $\prod_{1 \leq i \leq k} (r_i + 1)!$.

Summarizing, Golumbic says that the maximal multiplexes partition the edges and act independently with respect to transitive orientation. Thus, every comparability graph behaves as if it was a disjoint collection of complete graphs.

The following lemma plays an important role in the theory of Golumbic. As we will need it in an undirected context, we give our undirected version of it. See Figure 3 for an illustration of the proof.

Lemma 4. (The Triangle Lemma). *Let \mathcal{A} and \mathcal{B} be color classes of an undirected graph $G = (V, \mathcal{E})$ with $\mathcal{A} \neq \mathcal{B}$ and having edges $\widehat{ac} \in \mathcal{B}$ and $\widehat{bc} \in \mathcal{A}$. Then the following is true:*

- (i) *The undirected edge \widehat{ab} exists in G , let \mathcal{C} be its color class.*
- (ii) *If $\mathcal{C} \neq \mathcal{A}$ and $\widehat{b'c'} \in \mathcal{A}$, then $\widehat{ac'} \in \mathcal{B}$ or $\widehat{ab'} \in \mathcal{B}$.*
- (iii) *If $\mathcal{C} \neq \mathcal{A}$ then no undirected edge in \mathcal{A} touches a .*
- (iv) *If $\widehat{b'c'} \in \mathcal{A}$ and $\widehat{d'c'} \in \mathcal{B}$ then $\widehat{a'b'} \in \mathcal{C}$.*

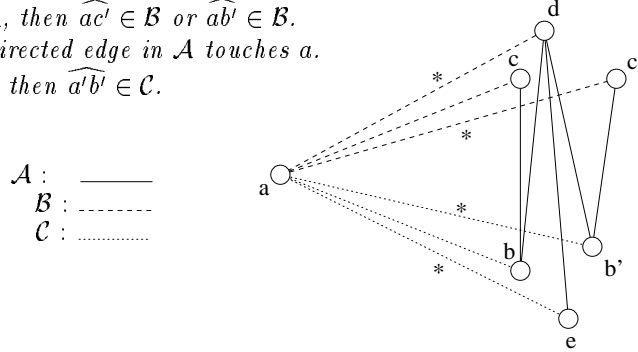


Figure 3. The Triangle Lemma. The existence and the color of the undirected edges marked with a * comes from the following hypothesis. It is supposed $\mathcal{B} \neq \mathcal{A}$ and $\widehat{bc}, \widehat{b'c'} \in \mathcal{A}$, and more precisely: $bc \sim bd \sim ed \sim b'd \sim b'c'$. From $\mathcal{C} \neq \mathcal{A}$, one can deduce the existence of \widehat{ad} . From $\widehat{cd} \notin \mathcal{E}$, one can deduce $\widehat{ad} \in \mathcal{B}$. The reasoning is analogous for the other marked edges.

4. Recognition Algorithm

The parallelization of Golumbic's algorithm is straightforward, and we only give its parallel version. It is based on theorem 1(iii).

In all our algorithms, n and m always denote the numbers of vertices and edges respectively of the input graph. δ denote the maximal degree of its vertices. The vertices are numbered and the edges are couples of numbers.

All our algorithms will be based on the following "elementary" parallel routines. The Cole Parallel Merge Sort [Col88] can sort p numbers in $O(\log p)$ time using p processors on EREW PRAM. Sum, products and conjunction of p elements can be implemented on EREW PRAM with parallel prefix computation in $O(\log p)$ time using $\frac{p}{\log p}$ processors. The connected components of a graph with p vertices and q edges can be computed [SV82] on CRCW PRAM in $O(\log p)$ time using $p + q$ processors. The complexities of our algorithms follow easily from the complexities of these routines.

In the following algorithm, we will need the degree $d(u)$ of each vertex u , it can be computed with a prefix sum on its adjacency list.

Algorithm 1.

Input: The edges of an undirected graph $G = (V, \mathcal{E})$ in both array of edges and sorted adjacency lists forms(*).

(*) This input form is not restrictive since the second structure is easily obtained by sorting lexicographically the first one, without increasing the complexity of the algorithm.

Output: Each edge’s implication class number and color class number. Returns true if G is a comparability graph.

Step 1. Compute the direct forcing relation \sim :

For $1 \leq e \leq m$ do
 $ij :=$ the e^{th} edge
 For $1 \leq s \leq d(i)$ do
 $k :=$ the s^{th} adjacent vertex of i
 Test (in time $O(\log \delta)$) if k is in the sorted list of successors of j
 If it is not, put in memory $ij \sim ik$ and $ji \sim ki$.

Step 2. Compute the implication classes as the connected components of the graph (\mathcal{E}, \sim) .

Step 3. Check if G is a comparability graph:

For $1 \leq e \leq m$ do
 $ij :=$ the e^{th} edge
 Read (in time $O(\log n)$) the implication class number of ji
 Set the color class of ij to the greatest implication class number of ij and ji
 If ij and ji have same implication class number
 then set $A_e := \text{false}$
 else set $A_e := \text{true}$
 Return $\bigwedge_{1 \leq e \leq m} A_e$.

Theorem 5. *Algorithm 1 determines whether an undirected graph is a comparability graph. In both cases, it computes the implication and color classes of the graph. It runs on CRCW PRAM in $O(\log n)$ time using δm processors.*

This improves the result of Novick [Nov89b] where n^3 processors are used.

5. Extending The Theory of Golumbic

In sequential, a very simple modification of the recognition algorithm enables to compute a transitive orientation with the same complexity. Implication classes are computed one after another. By removing the edges of the last computed implication class and resuming the algorithm on the remaining graph, Golumbic obtains a “ G -decomposition” where each class can be oriented independently to get a transitive orientation of the whole graph.

Unfortunately, this doesn’t work in parallel and we have to find a new algorithmic approach to this problem. In fact, we will give a new vision of the maximal multiplexes. Moreover, we will show further how they are closely related to modular decomposition.

Recall the problem: how do color classes interact? Let \mathcal{A} and \mathcal{B} be distinct color classes of an undirected graph $G = (V, \mathcal{E})$. We say that \mathcal{A} touches \mathcal{B} if $V_{\mathcal{A}} \cap V_{\mathcal{B}} \neq \emptyset$. By applying twice the Triangle Lemma 4(iv) in each of the three cases: $\mathcal{C} = \mathcal{B}$, $\mathcal{C} = \mathcal{A}$ and $\mathcal{A}, \mathcal{B}, \mathcal{C}$ distinct, we can then show that there exists a unique color class \mathcal{C} such that for each $c \in V_{\mathcal{A}} \cap V_{\mathcal{B}}$, $\widehat{bc} \in \mathcal{A}$ and $\widehat{ac} \in \mathcal{B}$, the undirected edge \widehat{ab} (which must exist in G) is in \mathcal{C} . We then say that \mathcal{A} touches \mathcal{B} by \mathcal{C} .

If \mathcal{A} touches \mathcal{B} by \mathcal{C} , then \mathcal{C} touches \mathcal{A} by \mathcal{B} , and \mathcal{B} by \mathcal{A} . If $\mathcal{C} = \mathcal{A}$, then we say that \mathcal{A} covers \mathcal{B} . If $\mathcal{C} \neq \mathcal{A}$ and $\mathcal{C} \neq \mathcal{B}$, then we say that \mathcal{A} crosses \mathcal{B} (or \mathcal{A}, \mathcal{B} and \mathcal{C} cross each other). See Figure 4.

Theorem 6. *The maximal unions of color classes crossing one another are the maximal multiplexes.*

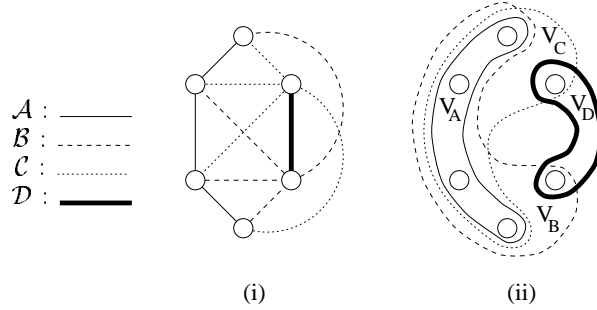


Figure 4. (i) The graph of Figure 2. Its color classes are $\mathcal{A}, \mathcal{B}, \mathcal{C}, \mathcal{D}$. (ii) Each color class \mathcal{B}, \mathcal{C} covers \mathcal{A} . The color classes $\mathcal{B}, \mathcal{C}, \mathcal{D}$ cross each other.

This new definition of the maximal multiplexes gives us a simple way to compute the maximal multiplexes and by the way the number of transitive orientations of a given comparability graph.

Notice that a maximal multiplex contains either only one color class or at least three.

Theorem 7. *Let \mathcal{M} be a maximal multiplex containing at least three color classes. The two following properties hold.*

- (i) *Each color class in \mathcal{A} induces a complete bipartite subgraph in the following sense. Let \mathcal{B} be a color class crossing \mathcal{A} by a color class \mathcal{C} . Then \mathcal{A} is the set of all the undirected edges joining a vertex in $V_{\mathcal{A}} \cap V_{\mathcal{B}}$ and a vertex in $V_{\mathcal{A}} \cap V_{\mathcal{C}}$. $V_{\mathcal{A}} \cap V_{\mathcal{B}}$ and $V_{\mathcal{A}} \cap V_{\mathcal{C}}$ are called supervertices of \mathcal{M} joined by the superedge \mathcal{A} . \mathcal{A} has then exactly two transitive orientations: the set of all the edges from $V_{\mathcal{A}} \cap V_{\mathcal{B}}$ to $V_{\mathcal{A}} \cap V_{\mathcal{C}}$ and its inverse set.*
- (ii) *The graph with the supervertices and the superedges of \mathcal{M} as vertices and edges is a complete graph, see Figure 5. Each maximal simplex generating \mathcal{M} can be obtained by picking a vertex in each supervertex of \mathcal{M} .*

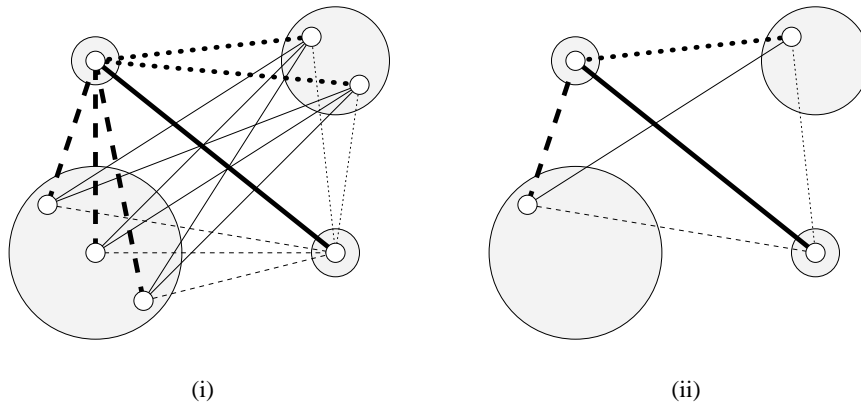


Figure 5. (i) The structure of complete graph of a maximal multiplex. The big disks figure the supervertices. (ii) A simplex generating (i). It is isomorphic to the complete graph on the supervertices.

The simplices generating \mathcal{M} are simply subgraphs isomorphic to the complete graph on the supervertices of \mathcal{M} . This isomorphism is in the heart of modular decomposition. We will develop this further.

Let us now focus on the covering relation. We already know from its definition that it is an order. The following theorem extends this relation to maximal multiplexes.

Theorem 8.

- (i) If a color class \mathcal{A} covers a color class \mathcal{B} in a maximal multiplex \mathcal{M} , then it covers each color class in \mathcal{M} (and thus is not in \mathcal{M}). We will then say that the maximal multiplex \mathcal{N} containing \mathcal{A} covers \mathcal{M} .
- (ii) If two distinct maximal multiplexes \mathcal{M} and \mathcal{N} touch each other, i.e. $V_{\mathcal{M}} \cap V_{\mathcal{N}} \neq \emptyset$, then one covers the other.
- (iii) A maximal multiplex \mathcal{M} covers a maximal multiplex \mathcal{N} if and only if $V_{\mathcal{M}} \supseteq V_{\mathcal{N}}$.
- (iv) The covering relation over the maximal multiplexes of a connected undirected graph is a tree order.

6. Algorithms

We can now give parallel algorithms for computing transitive orientation, the maximal multiplexes and the number of transitive orientations.

6.1 Computing A Transitive Orientation

Surprisingly, we can find a transitive orientation corresponding to the orientation of specific simplices without computing either them or the maximal multiplexes. The maximal simplices chosen are implicitly those obtained by picking the vertex with minimal number in each supervertex. Each simplex is oriented according to the total order given by the numbers of its vertices. We simply compute the union of the edges of these oriented simplices (one edge in each color class). See Figure 6 for an example.

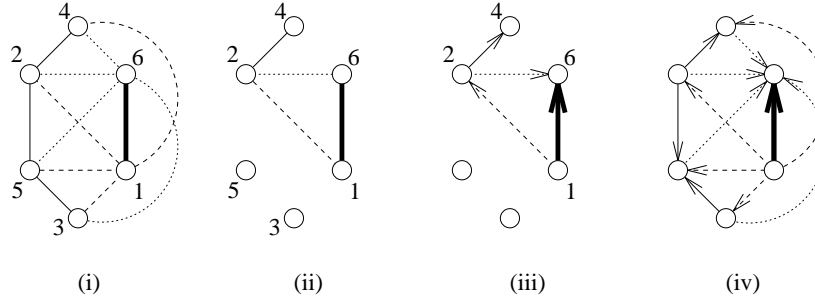


Figure 6. (i) The comparability graph of Figure 2 with numbered vertices. (ii) The union of the implicitly chosen simplices. (iii) The union of the oriented simplices. (iv) The corresponding orientation of the graph.

Algorithm 2.

Input: A comparability graph, its color and implication classes, given by the list of its edges, their color class numbers and implication class numbers.

Output: A transitive orientation of the graph.

Step 1. Select an edge in each color class:

Sort the edges by color class number.

Sort the edges lexicographically in each block of same color class number.

For each edge ab of color class number c and implication class number i do

If ab is the first edge of color c in the sorted list then set $I(c) := i$

Step 2. Orient the whole graph:

For each edge ab of color class number c and implication class number i do
 If $I(c) = i$ then mark ab . { This concurrent read can be done with $\log m$ exclusive reads. }
 { The marked edges form a transitive orientation. }

Theorem 9. *Algorithm 2 determines a transitive orientation of any undirected graph whose implication and color classes are given. It runs on EREW PRAM in $O(\log n)$ time using m processors.*

The correctness of the algorithm follows from Theorem 3(ii).

6.2 Computing The Maximal Multiplexes

We could compute the maximal multiplexes similarly to the implication classes using Theorem 6. But we can get a better complexity by working on the simplices thanks to the following property.

Theorem 10. *Let $G = (V, \mathcal{E})$ be an undirected graph. Let (V, \mathcal{S}) be a union of maximal simplices of G such that exactly one simplex is in each maximal multiplex of G . The maximal simplices are the biconnected components of (V, \mathcal{S}) .*

We consider that an undirected graph is biconnected if it is connected and still connected after removing any single vertex. Equivalently, an undirected graph is biconnected if for any pair of distinct edges, there exists a simple cycle containing both of them.

Notice that such a graph (V, \mathcal{S}) is computed in Algorithm 2.

The algorithm is simple: compute such a graph (V, \mathcal{S}) as in Algorithm 2 and find its biconnected components.

Algorithm 3.

Input: Any undirected graph and its color classes given by the list of its edges and their color class numbers.

Output: For each color class, the maximal multiplex containing it.

Step 1. Compute a union of maximal simplices:

Sort the edges by color class number.

Sort the edges lexicographically in each block of same color class number.

For each edge ab of color class number c do

If ab is the first edge of color c in the sorted list then set $E(c) := \widehat{ab}$.

Compute the biconnected components of the graph given by the edge list E .

For each edge ab of color class number c , set its maximal multiplex number to the biconnected component number of $E(c)$.

Step 2. Compute the number of transitive orientations:

If the graph is not a comparability graph then it has 0 transitive orientation

Else

Sort the list E of color classes according to their multiplex numbers.

With a prefix sum, compute the number $N(m)$ of color classes in each maximal multiplex of number m .

The number of vertices of a simplex generating a maximal multiplex of number m

is given by $V(m) := \frac{1 + \sqrt{8N(m) + 1}}{2}$. { We have $N(m) = V(m)(V(m) - 1)/2$ since a simplex is a complete graph. }

With a prefix product, compute the number of transitive orientations which is $\prod_m V(m)!$ according to Theorem 3(iii).

The biconnected components of a graph with p vertices and q edges can be computed [Tar85] on CRCW PRAM in $O(\log p)$ time using $p + q$ processors. We thus deduce:

Theorem 11. *Algorithm 3 computes the maximal multiplexes and the number of transitive orientations of any undirected graph whose color and implication classes are given. It runs on CRCW PRAM in $O(\log n)$ time using $n + m$ processors.*

The correctness of the algorithm follows from Theorems 10 and 3(iii).

7. Modular Decomposition

In this section we show how the structure of maximal multiplex is closely related to modular decomposition. Moreover we will give a parallel modular decomposition algorithm based on the computation of the maximal multiplexes.

7.1 Definition

Let us first introduce quickly the modular decomposition, see [Moh89] for an overview.

Let $G = (V', \mathcal{E}')$, $G_1 = (V_1, \mathcal{E}_1), \dots, G_k = (V_k, \mathcal{E}_k)$ be undirected graphs with disjoint vertex sets. Let a_1, \dots, a_k be distinct vertices of G . The *composition graph* $G_{(G_1, \dots, G_k)}^{(a_1, \dots, a_k)} = (V, \mathcal{E})$ is the graph resulting from substituting each a_i of G by G_i , $i = 1, \dots, k$. More formally:

$$V = (V' - \{a_1, \dots, a_k\}) \cup V_1 \cup \dots \cup V_k$$

and $\mathcal{E} = \mathcal{E}_1 \cup \dots \cup \mathcal{E}_k \cup \bigcup_{i \neq j} \{ab \mid a \in V_i, b \in V_j \mid a_i a_j \in \mathcal{E}'\}.$

The edges in $\bigcup_{i \neq j} \{ab \mid a \in V_i, b \in V_j \mid a_i a_j \in \mathcal{E}'\}$ are called *internal* edges of the composition. The composition is *proper* if $1 < |V_i| < |V|$ for some i . A graph $G = (V, \mathcal{E})$ is *decomposable* if it can be obtained by proper composition. Otherwise it is said to be *prime*. A subset M of V is called an *homogeneous* set or a *module* if $\widehat{ab}_0 \in \mathcal{E}$ for some $a \in V - M$ and $m_0 \in M$ implies that $\widehat{am} \in \mathcal{E}$ for all $m \in M$. $G = (V, \mathcal{E})$ is *decomposable* if and only if it has a proper module M (i.e. a module with $1 < |M| < |V|$). Then $G = H(G_M^a)$ where G_M is the subgraph of G induced by M , and where H is obtained from G by replacing M by just one vertex a .

The following basic decomposition theorem is due to [Gal67].

Theorem 12. *For each decomposable graph $G = (V, \mathcal{E})$, one of the following cases occurs:*

- (i) $G = H(G_1^{a_1}, \dots, G_k^{a_k})$, where H is an independent graph (i.e. with no edge). Then G is obtained by parallel composition of G_1, \dots, G_k .
- (ii) $G = H(G_1^{a_1}, \dots, G_k^{a_k})$, where H is a complete graph. Then G is obtained by series composition.
- (iii) $G = H(G_1^{a_1}, \dots, G_k^{a_k})$, where H is a uniquely determined prime graph. Then G is obtained by prime type composition.

These three *mutually exclusive* cases are used to represent any partial order in a *decomposition tree* T . The root of T is V , the leaves are the vertices of G , and the sons of interior nodes are the vertex sets of the graphs G_i in the respective composition (parallel, series or prime type) of the graph associated with the node. The tree is unique if H is taken as large as possible in (i) and (ii). This tree is called the *canonical decomposition tree*. Then the tree nodes correspond to the

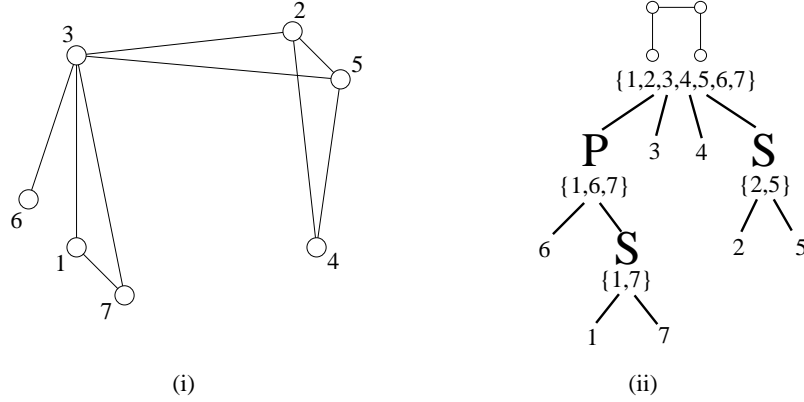


Figure 7. (i) A graph. (ii) Its canonical decomposition tree.

so-called *strong modules*, *i.e.* those modules that do not properly overlap with any other module. A node of the tree is labeled with P (respectively S) if it is a parallel (respectively series) node, and the corresponding graph H if it is a prime type node. See Figure 7.

7.2 Maximal Multiplexes and Modular Decomposition

We can now explain the link between maximal multiplexes and modular decomposition in the following theorem. (i) is the only point appearing in [Gol85].

Theorem 13. *Let $G = (V, \mathcal{E})$ be an undirected graph.*

- (i) [Gol85] *For all color class \mathcal{A} , $V_{\mathcal{A}}$ is a module.*
- (ii) *Let M be a module and \widehat{ab} be an undirected edge such that $a, b \in M$. If \mathcal{A} is the color class containing \widehat{ab} , then $V_{\mathcal{A}} \subseteq M$.*
- (iii) *For all maximal multiplex \mathcal{M} , $V_{\mathcal{M}}$ is a strong module.*
- (iv) *Let M be a strong module and \widehat{ab} be an undirected edge such that $a, b \in M$. If \mathcal{M} is the maximal multiplex containing \widehat{ab} , then $V_{\mathcal{M}} \subseteq M$.*

We can now see that Theorem 7 is simply a decomposition theorem for the series nodes. We can generalize this as follows (see also Figure 8).

Theorem 14. *The canonical decomposition tree of any undirected graph $G = (V, \mathcal{E})$ verifies the following properties.*

- (i) *Every non parallel interior node is a strong module $V_{\mathcal{M}}$ where \mathcal{M} is a maximal multiplex. Moreover \mathcal{M} is the set of all the internal edges of the composition.*
- (ii) *A node corresponding to a module \mathcal{M} is an ancestor of a node corresponding to a module \mathcal{N} if and only if \mathcal{M} covers \mathcal{N} , or equivalently $V_{\mathcal{M}} \supseteq V_{\mathcal{N}}$.*

These results lead us to develop the following decomposition algorithm.

7.3 A Parallel Modular Decomposition Algorithm

We can now give a parallel modular decomposition algorithm for any undirected graph. It comes almost straight forward from the theoretical results we have obtained. It yields to the same time bound as the algorithm proposed by Novick [Nov89a]. This result will be improved in a paper by Dalhaus [Dal95] but we had no chance to read it yet. We give our algorithm as a conclusion to the theoretical approach that we have developed.

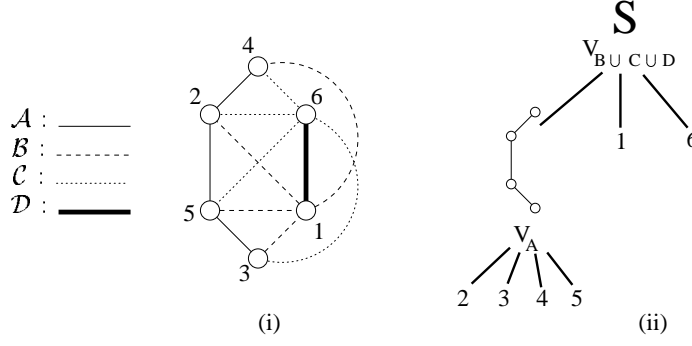


Figure 8. (i) The graph of Figure 2. Its maximal multiplexes are \mathcal{A} and $\mathcal{B} \cup \mathcal{C} \cup \mathcal{D}$.
(ii) Its canonical decomposition tree.

Notice that an undirected graph has at most $n - 1$ maximal multiplexes since its canonical decomposition tree has n leaves and thus less than $n - 1$ interior nodes.

Algorithm 4.

Input: An undirected graph G . Its complementary graph \overline{G} . The maximal multiplexes of G and \overline{G} .

Output: The canonical decomposition tree of G .

Step 1. Computing the nodes of the tree:

For all edge ab of maximal multiplex number m of G or \overline{G} , sort lexicographically the n^2 triples (a, m, b) .

Calculate with a prefix computation the list L of all the couples (a, m) appearing at the beginning of a triple.

Sort L anti-lexicographically to obtain $V_{\mathcal{M}}$ for each m where \mathcal{M} is the maximal multiplex numbered m .

For each maximal multiplex \mathcal{M} of G or \overline{G} calculate $|V_{\mathcal{M}}|$ and $\min(V_{\mathcal{M}})$ with prefix computations.

Sort the triples $T(\mathcal{M}) = (|V_{\mathcal{M}}|, \min(V_{\mathcal{M}}), \text{number of } \mathcal{M})$ lexicographically.

If two consecutive triples $T(\mathcal{M})$ and $T(\mathcal{N})$ have same first two components, then \mathcal{M} and \mathcal{N} correspond to the same prime type node.

{ If two nodes have a common vertex, then one is the ancestor of the other and spans strictly more vertices. }

The maximal multiplexes of G (respectively \overline{G}) attached to no maximal multiplex of \overline{G} (respectively G) correspond to the series (respectively parallel) nodes.

Step 2. Computing the father of each node:

Read in the lexicographically sorted list L the sorted lists $N(a)$ of all the nodes containing each vertex a .

For each node \mathcal{N} do

 Pick an edge ab in a corresponding maximal multiplex of G or \overline{G} .

{ As ab is an internal edge of the corresponding decomposition, a descendant of \mathcal{N} cannot contain both a and b . Thus the list $A(\mathcal{N})$ of the ancestors of \mathcal{N} is given by $N(a) \cap N(b) - \{\mathcal{N}\}$. }

 Compute $A(\mathcal{N})$ by merging $N(a)$ and $N(b)$.

 Calculate with a prefix computation the father \mathcal{M} of \mathcal{N} which is the element of $A(\mathcal{N})$ with minimal $|V_{\mathcal{M}}|$.

For each leaf a , set the father \mathcal{M} of a to the element of $N(a)$ with minimal $|V_{\mathcal{M}}|$.

Theorem 15. *Algorithm 4 determines the canonical decomposition tree of an undirected graph when the complementary graph and the maximal multiplexes of the two graphs are given. It runs on EREW PRAM in $O(\log n)$ time using n^2 processors.*

8. Conclusion

Let us recall the results obtained so far. Algorithm 1 computes the color and implication classes of any undirected graph and determines whether it is a comparability graph. Algorithm 2 and Algorithm 3 respectively compute a transitive orientation and the maximal multiplexes of a graph whose color and implication classes are given. Algorithm 4 compute the canonical decomposition tree of a graph when its complementary graph and the maximal multiplexes of the two graphs are given.

Combining these algorithms, we get a transitive orientation algorithm and a modular decomposition algorithm. They both run on CRCW PRAM in $O(\log n)$ time. They respectively use δm and n^3 processors.

References

- [CH94] A. Cournier and M. Habib. A new linear algorithm of modular decomposition. In *Trees in algebra and programming—CAAP 94* (Edinburgh) Lecture Notes in Computer Science, volume 787, pages 68–84, Berlin, 1994. Springer.
- [Col88] Richard Cole. Parallel merge sort. *SIAM J. Comput.*, 17(4), August 1988.
- [Dal95] Elias Dalhaus. Efficient parallel modular decomposition. In *WG '95 21st International Workshop on Graph-Theoretic Concepts in Computer Science*. M. Nagl, 1995. To appear in Lecture Notes in Computer Science.
- [Gal67] Tibor Gallai. Transitiv orientierbare graphen. *Acta Math. Acad. Scient. Hung. Tom.*, 18:25–66, 1967.
- [Gol85] M. C. Golumbic. *Algorithmic Graph Theory and Perfect Graphs*. Academic Press, New York, 1985.
- [Moh89] Rolf H. Mohring. Computationally tractable classes of ordered sets. In I. Rival, editor, *Algorithms and Order*, pages 105–193. Kluwer Acad. Publ., Dordrecht, 1989.
- [MS94] M.R. McConnell and J. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proceedings of the Fifth Annual ACM-SIAM Symposium on Discrete Algorithms* (Arlington, VA), pages 536–545, New York, 1994. ACM.
- [Nov89a] Mark B. Novick. Fast parallel algorithms for the modular decomposition. Technical Report 89-1016, Cornell University, 1989.
- [Nov89b] Mark B. Novick. Logarithmic time parallel algorithms for recognizing comparability and interval graphs. Technical Report 89-1015, Cornell University, 1989.
- [SV82] Y. Shiloah and U. Vishkin. An $O(\log n)$ parallel connectivity algorithm. *J. Algorithms*, 3:57–67, 1982.
- [Tar85] R.E. Tarjan. An efficient parallel biconnectivity algorithm. *SIAM J. Computing*, 14:862–874, 1985.