4th Lecture : Modular decomposition MPRI 2013–2014

Michel Habib
habib@liafa.univ-Paris-Diderot.fr
http://www.liafa.univ-Paris-Diderot.fr/~habib

Sophie Germain, 15 octobre 2013

Schedule

Examen le mardi 26 novembre de 9h à 12h,
Salle habituelle

A nice algorithm

A very simple algorithm to buid to recognize a cograph from a factoring permutation.
Analogous to Jarvis's algorithm for computing the convex hull of a set of points in the plane.

Algorithm 5: Recognition test

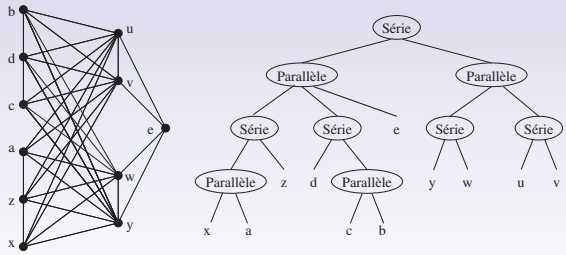
Input: Let $\sigma = x_1, \ldots, x_n$ be a permutation of the vertex set of a graph G, σ is represented as a doubly linked list.

Output: σ a list of vertices

begin
- Let x_0 and x_{n+1} be added to σ (these vertices are dummies which are not twins with any other vertex)
- Let z be the current vertex, initially $z \leftarrow x_1$
- Let $succ(z)$ (resp. $prec(z)$) be the vertex following (resp. preceding z) in σ
- **while** $z \neq x_{n+1}$ **do**
  - **if** *z and $prec(z)$ are twins (true or false) in $G(\sigma)$* **then**
    - remove $prec(z)$ from σ
  - **else**
    - **if** *z and $succ(z)$ are twins (true or false) in $G(\sigma)$* **then**
      - $z \leftarrow succ(z)$
      - remove $prec(z)$ from σ
    - **else** $z \leftarrow succ(z)$
- **if** $|\sigma - \{x_0, x_{n+1}\}| = 1$ **then return** G is a cograph **else return** $G(\sigma)$ contains a P_4

end

An example



Elimination scheme

G is a cograph iff it exists an ordering of the vertices s.t. x_i has a twin (false or true) in $G\{x_{i+1}, \ldots x_n\}$

Theorem

This algorithm finds an elimination scheme iff σ is a factoring permutation of a cograph.

Proof :

Main Invariant : For any $k \geq 1$,the subsequence $\sigma_k([z_0, z_k[)$ does not contain any twins vertices in $G(\sigma_k)$.
If the algorithms finds an elimination scheme characteristic of cographs (using twins), then we know that G is a cograph.

Exercises :

1. In case of failure, where is the P_4 ?
2. In case of success, how to derive the tree ?

A logarithmic process

Let $\mathcal{P}$ be a problem on a set $\mathcal{S}$ of data structures, and *Size* a function from $\mathcal{S}$ to $\mathbb{R}^+$. $\mathcal{H}$ is a *divide-and-conquer algorithm with respect to Size solving* $\mathcal{P}$ if :

- there exists a set $\mathcal{T} \subseteq \mathcal{S}$ of trivial inputs on which $\mathcal{H}$ solves $\mathcal{P}$ in $O(1)$ time ;
- any $S \in \mathcal{S}$ with $Size(S) \leq 1$ is a trivial input, namely $S \in \mathcal{T}$;
- for all $S \notin \mathcal{T}$, $\mathcal{H}(\mathcal{S})$
  - first divides S into some sub-instances $S_1, \ldots, S_k$ holding $Size(S_i) > 0$ for all i and holding $Size(S_1) + \cdots + Size(S_k) \leq Size(S)$,
  - then recurses with $\mathcal{H}(S_1)$, ..., $\mathcal{H}(S_k)$,
  - and finally combines the results in order to provide the output of $\mathcal{H}(\mathcal{S})$.

proposition

Let $\mathcal{H}$ be a divide-and-conquer algorithm, and α be such that, for all $S \in \mathcal{S} \setminus \mathcal{T}$,
$Div(S) + Com(S) \leq \alpha \times (Size(S) - \max_{i=1}^{k} Size(S_i))$, where $S_1, \ldots, S_k$ is the partition of S given by $\mathcal{H}(\mathcal{S})$. Then, for all input $S \in \mathcal{S}$, $\mathcal{H}(\mathcal{S})$ runs at most in $\alpha \times Size(S) \log Size(S)$ time. This bound is best possible.

Proof :

by induction on $s = Size(S)$. If S is not trivial and $S_1, \ldots, S_k$ are such that $s_k = Size(S_k)$ is greater than any $s_i = Size(S_i)$, then

$$\begin{aligned} Div(S) \quad + \quad & Com(S) + \sum_{i=1}^{k} C(S_i) \leq \alpha \times \left(\sum_{i=1}^{k-1} s_i + \sum_{i=1}^{k} s_i \log s_i \right) \\ & \leq \alpha \times \left(\sum_{i=1}^{k-1} s_i + \sum_{i=1}^{k-1} s_i \log \frac{s}{2} + s_k \log s \right) \\ & \leq (\alpha \times s \log s) \end{aligned}$$

.

Remarks

The standard optimisation technique used in Merge sort results in the same bound. However, the size of the input given to Merge sort is granted to geometrically decrease (by half) as inductive levels grow, implying that the induction depth is lesser than $\log Size(S)$. On the other hand, this result still holds even when the induction depth is linear on $Size(S)$.
For a graph G, $Size(G) = |V(G)| + |E(G)|$

Historical notes

The big list of published algorithms for modular decomposition (N.B. Perhaps some items are missing ... please give me the missing references)

- Cowan, James, Stanton 1972 $O(n^4)$
- Maurer 1977 $O(n^4)$ directed graphs
- Blass 1978 $O(n^3)$
- Habib, Maurer 1979 $O(n^3)$
- Habib 1981 $O(n^3)$ directed graphs
- Corneil, Perl, Stewart 1981, $O(n+m)$ cograph recognition.
- Cunningham 1982 $O(n^3)$ directed graphs
- Buer, Mohring 1983 $O(n^3)$
- McConnell 1987 $O(n^3)$
- McConnell, Spinrad 1989 $O(n^2)$ incremental
- Adhar, Peng 1990 $O(log^2 n)$, $O(nm)$ proc. parallel, cographs, CRCW-PRAM

- Lin, Olariu 1991 $O(logn)$, $O(nm)$ proc. parallel, cographs, EREW-PRAM
- Spinrad 1992 $O(n + malpha(m, n))$
- Cournier, Habib 1993 $O(n + malpha(m, n))$
- Ehrenfeucht, Gabow, McConnell, Spinrad 1994 $O(n^3)$ 2-structures
- Ehrenfeucht, Harju, Rozenberg 1994 $O(n^2)$ 2-structures, incremental
- McConnell, Spinrad 1994 $O(n+m)$
- Cournier, Habib 1994 $O(n+m)$
- Bonizzoni, Della Vedova 1995 $O(n^{3k-5})$ Committee decomposition for hypergraphs
- Dahlhaus 1995 $O(log^2 n)$, $O(n+m)$ proc. parallel, cographs, CRCW-PRAM
- Dahlhaus 1995 $O(log^2 n)$, $O(n+m)$ proc. parallel, CRCW-PRAM

- Habib, Huchard, Sprinrad 1995 $O(n+m)$ inheritance graphs
- McConnell 1995 $O(n^2)$ 2-structures, incremental
- Capelle, Habib 1997 $O(n+m)$ if a factoring permutation is given
- Dahlhaus, Gustedt, McConnell 1997 $O(n+m)$
- Dahlhaus, Gustedt, McConnell 1999 $O(n+m)$ directed graphs
- Habib, Paul, Viennot 1999 $O(n + mlogn)$ via a factoring permutation
- McConnell, Spinrad 2000 $O(n + mlogn)$
- Habib, Paul 2001 $O(n+m)$ cographs via a factoring permutation
- Capelle, Habib, Montgolfier 2002 $O(n+m)$ directed graphs if a factoring permutation is provided.
- Shamir, Sharan 2003 $O(n+m)$ cographs, fully-dynamic
- McConnell, Montgolfier 2003 $O(n+m)$ directed graphs
- Habib, Montgolfier, Paul 2003 $O(n+m)$ computes a factoring permutation

- Simpler Linear-Time Modular Decomposition via Recursive Factorizing Permutation Tedder, Corneil, Habib, Paul, ICALP (1) 2008 : 634-646.

Why it is so important ?

[Jerry Spinrad' book 03]

The new [linear time] algorithm [MS99] is currently too complex to describe easily [...] The first $O(n^2)$ partitioning algorithms were similarly complex ; I hope and believe that in a number of years the linear algorithm can be simplified as well.

Applications of modular decomposition

- A very natural operation to define on discrete structures, searching regularities.
- A structure theory for comparability graphs
- A compact encoding using module contraction and if we keep at each prime node the structure of the prime graph.
- Divide and conquer paradigm can be applied to solve optimization problems. For example to test isomorphism.

- A very basic graph algorithmic problem (similar to graph isomorphism problem).
- A better understanding of graph algorithms and their data structures.

Minimal Modules

Minimal module containing a set

For every $A \subseteq V$ there exists a unique minimal module containing A

Proof :

Since the module family is partitive and therefore closed under $\cap$.

Splitters

Definition

A splitter for a $A \subseteq V$, is a vertex $z \notin A$
s.t. $\exists x, y \in A$ with $zx \in E$ and $zy \notin E$.

Modules

$A \subseteq V$ is a module iff A does not have any splitter.

Usefull lemma

If z is a splitter for a $A \subseteq V$, then any module containing A must also contain z.

Submodularity

Let us denote by $s(A)$ the number of splitters of a set A, then s is a submodular function.

Definition

A function is submodular if
$\forall A, B \subseteq E$
$f(A \cup B) + f(A \cap B) \leq f(A) + f(B)$

This is the basic idea of Uno and Yagura's algorithm for the modular decomposition of permutation graphs in $O(n)$.

Bottom-up Techniques

Sketch of the algorithm

For each pair of vertices $x, y \in V$
Compute the minimal module $m(x, y)$ containing x and y.

Closure with splitters

While there exists a splitter add it to the set.

Complexity

$O(n^2.(n + m))$

Primality testing

One can derive a primality test since if there exists a non trivial module, it contains at least two vertices.

For some problems Bottom-Up techniques are the best known.

Origins : Golumbic, Kaplan, Shamir 1995

Input : $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ 2 undirected graphs such that $E_1 \subseteq E_2$ and Π be a graph property.
Results : a sandwich graph $G_s = (V, E_s)$ satisfying property Π and such that $E_1 \subseteq E_s \subseteq E_2$.
Edges of E_1 are forced, those of E_2 are optional ones, but those of $E_3 = \overline{E_2}$ are forbidden.

Unfortunately most cases are NP-complete, as for example of Π

- G_s being comparability, chordal, strongly chordal, ...

Only few polynomial cases

- cographs Golumbic, Kaplan, Shamir (1995)
- sandwich module Cerioli, Everett, de Figueiredo, Klein (1998)

...

Natural question

Find efficient algorithms for these polynomial cases.

Sandwich module problem

Input : $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ 2 undirected graphs such that $E_1 \subseteq E_2$.
Result : a sandwich graph $G_s = (V, E_s)$ having a non trivial module and such that $E_1 \subseteq E_s \subseteq E_2$.

Minimal Sandwich Module

Splitter

For a subset $A \subseteq V$, a splitter is a vertex $z \notin A$
s.t. $\exists x, y \in A$ with $zx \in E_1$ and $zy \notin E_2$ (or equivalently $zy \in E_3$)
A splitter is also called **bias vertex**.

Algorithm

The computation of a minimal sandwich module can be done in $O(n^2.(n + m_1 + m_3))$.

Hard to do better with this idea, using a bottom up approach.

Brute Force Algorithm

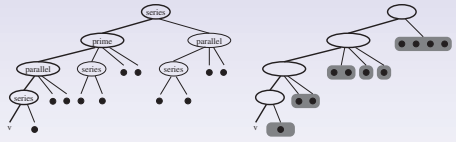
Using the decomposition theorem, we only have to compute at most n times some connected components of G or its complement.
$O(n.(n + m))$ complexity.

Three explored directions

- Ehrenfeucht et al approach
- Using Factoring Permutation
- Using LexBFS (as for cographs) Next lecture.

Ehrenfeucht et al approach

$\mathcal{M}(G, v)$ is composed by $\{v\}$ and the maximal modules of G that do not contain v.



Principle of the Ehrenfeucht et al.'s algorithm

1. Compute $\mathcal{M}(G, v)$
2. Compute $MD(G_{/\mathcal{M}(G,v)})$
3. For each $\mathcal{X} \in \mathcal{M}(G, v)$ compute $MD(G[\mathcal{X}])$

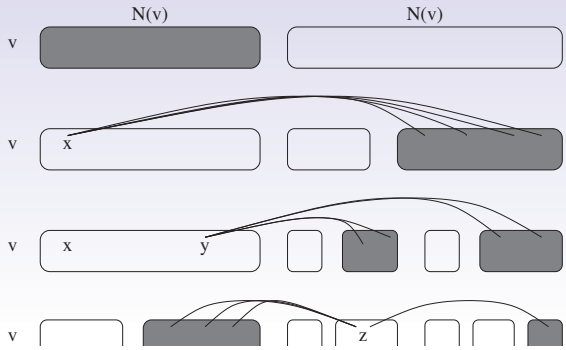
Computing $\mathcal{M}(G, v)$ via Partition Refinement

Splitter again

If z is a splitter of $A \subseteq V$ then any strong module contained in A is either contained in $N(z) \cap A$ or in $A - N(z)$.

Computation of $\mathcal{M}(G, v)$

$\Rightarrow O(n + m \log n)$ time using vertex partitioning algorithm.

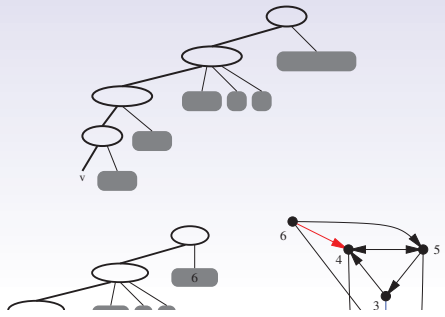


1. Particular partition refinement rule :
   Do not refine its part
   Just to maintain the invariant :
   Modular partition $\leq$ Current partition
2. To obtain a *logn*
   Avoid the biggest part

How to reconstruct the modular decomposition tree from the partition $\mathcal{M}(G, v)$?
The most difficult step in many algorithms.

Computation of $MD(G_{/\mathcal{M}(G,v)})$

- The modules of $G_{/\mathcal{M}(G,v)}$ are linearly nested :
  any non-trivial module contains v
- The *forcing graph* $\mathcal{F}(G, v)$ has edge $\overrightarrow{xy}$ iff y separates x and v



- The strong connected components of the *forcing graph* $\mathcal{F}(G, v)$ provides the modules of of $G_{/\mathcal{M}(G,v)}$.
- Recurse on each module.

Complexity

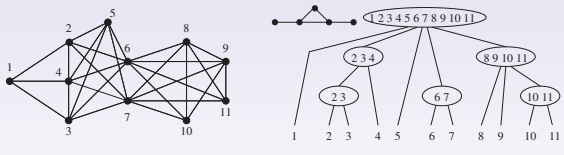
- [Ehrenfeucht et al.'94] gives a $O(n^2)$ complexity. It is quite tricky to efficiently compute the *forcing graph* $\mathcal{F}(G, v)$.
- [MS00] gives a very simple $O(n + m \log n)$ algorithm based on vertex partitioning.
- [DGM'01] proposes a $O(n + m.\alpha(n, m))$ and a more complicated $O(n + m)$ implementation.

Other algorithms

- [CH94] and [MS94] present the first linear algorithms.
- [MS99] present a new linear time algorithm which extends to transitive orientation.

Factoring permutations

The set of strong modules is nested into an inclusion tree (called the **modular decomposition tree** $MD(G)$ of G).



A factoring permutation is simply a left-right ordering of the leaves of a plane drawing of $MD(G)$.

Consequence : it always exists factoring permutations.
There are easier to compute that the modular decomposition tree.

Invariant

Any strong module is a factor of the partition.

Splitter interpretation

Starting with the partition $\{N(x), \{x\}, \overline{N(x)}\}$, we maintain the following invariant :
It exists a factoring permutation smaller than the current partition.

Recognition of "geometric" graph classes

Geometric in a very wide meaning, it could be :

- Embedding with some condition (planar, outerplanar ...) with polylines, with convex bodies or some generalization (ρ-convex, i.e. each edge is a polyline with at most ρ segments).
- Embedding with limited crossings
- Intersection graphs of some geometric objects (interval, chordal, permutation, trapezoids, ...)

A hierarchy of models

1. Undirected graphs (graphes non orientés)
2. Tournaments (Tournois), sometimes 2-circuits are allowed.
3. Signed graphs (Graphes signés) each edge is labelled + or - (for example friend or enemy)
4. Oriented graphs (Graphes orientés), each edge is given a unique direction (no 2-circuits)
   An interesting subclass are the DAG Directed Acyclic Graphs (graphes sans circuit), for which the transitive closure is a partial order (ordre partiel)
5. Directed graphs or digraphs (Graphes dirigés)

http ://math.nie.edu.sg/fmdong/Research/articles/beautiful
Second Neighbourhoods Conjecture
P.D. Seymour 1990
Every digraph without 2-circuits has a vertex with at least as many second neighbours as first neighbours.
Second neighbours, $SN(x)$ is the set of vertices at exact distance 2 of x.
Therefore we are looking for x such that $|SN(x)| \geq |N(x)|$.