

6th Course: Partition Refinement and Modular Decomposition

Michel Habib
habib@liafa.jussieu.fr
<http://www.liafa.jussieu.fr/~habib>

Chevaleret, october 2011

Schedule

- 1 Outcomes of the two lessons on treewidth
- 2 Partition Refinement

Several different viewpoints on tree decompositions

- Dynamic programming (k-trees)
- Chordal completion and generalization of maximal clique trees
- Graph rewriting systems and Logics (MSO and Courcelle's meta theorem)
- Phylogeny and tree of evolution
- Tools or parameters for inductive proofs (Roberston and Seymour)
- ...

We have dealt with important concepts:

- MSO and structural complexity
- Well quasi orders (wqo)
- Non constructive algorithms or existence theorems for algorithms.

Finding a minor in polynomial time

The problem of deciding whether a graph G contains H as a minor is NP-complete in general; for instance, if H is a cycle graph with the same number of vertices as G , then $H \leq_{\min} G$ if and only if G contains a Hamiltonian cycle.

However, when H is fixed, it can be solved in polynomial time.

More specifically, the running time for testing whether $H \leq_{\min} G$ in this case is $O(n^3)$, where n is the number of vertices in G and the big O notation hides a constant that depends superexponentially on $|H|$.

The skew-partition problem a pedagogical example

- 1 1985: This decomposition appears in the theory of perfect graphs V. Chvátal (a problem on minimal imperfect graphs). A skew partition is a partition of the vertex set of G in A and B , such that: $G(A)$ is not connected and $\overline{G(B)}$ is not connected. This notion is stable under complement.
- 2 1999: An algorithm in $O(n^{\log n})$ Feder, Hell, Klein, and Motwani (which “implies” not NP-complete).
- 3 2000: Proven to be in P , de Figuereido, Klein, Kohayakawa and Reed with an algorithm in $O(n^{101})$ (a kind of brute force algorithm).
- 4 2011: Best known algorithm so far works in $O(n^4 m)$ with reasonable constant and using some knowledge about skew partitions, Kennedy and Reed 2008.
- 5 ...

Classes of twin vertices

Definition

x and y are called **false twins**, (resp. **true twins**) if
 $N(x) = N(y)$ (resp. $N(x) \cup \{x\} = N(y) \cup \{y\}$)

Exercise

Propose a good algorithm to compute these classes

Algorithm Folklore

Data: $G = (V, E)$ a graph with n vertices and m edges

Result: The classes of false twin vertices

$Q \leftarrow \{V\}$

for Every $x \in V$ **do**

$Q \leftarrow \text{Refine}(Q, N(x))$

end

Partition Refinement

If $Q = \{C_1, \dots, C_k\}$

$\text{Refine}(Q, S) = \{C_1 \cap S, C_1 - S, \dots, C_k \cap S, C_k - S\}$

At the end, parts of Q have no splitter outside and therefore are modules.

Furthermore they have no splitter inside the part.

They are made up with false twins (non connected).

Complexity

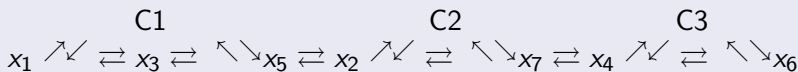
$$\sum_{x \in V} |N(x)| \in O(n + m)$$

Implementation

$$V = \{x_1, \dots, x_7\}$$

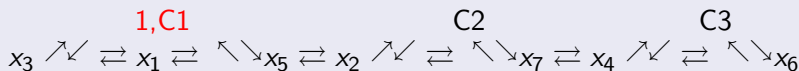
$$P = \{C1, C2, C3\} \text{ and } S = \{x_3, x_4, x_5\}$$

Data Structure

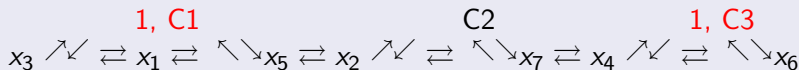


First Step

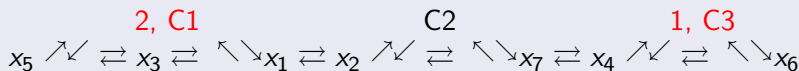
Processing x_3



Processing x_4



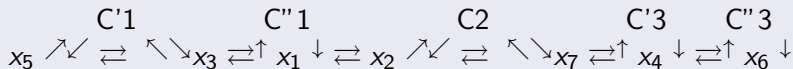
x_5



Second step

Maintain a list of the C_i 's that intersect S . List bounded by $|S|$.

Result



$Refine(P, S) = \{C'1, C''1, C2, C'3, C''3\}$

computed in $O(|S|)$

Exercises

- 1 Propose an implementation in $O(|S|)$ of Refine **stable** which preserves a given initial ordering of the vertices of the elements x_i 's. of the parts.
- 2 Propose an implementation in an array compatible with an initial ordering and within the same complexity.

Applications

Detection of multi-occurency in a list of subsets

Just consider the bipartite elements–subsets.

Recognition of a laminar family

Easy application

Computation of LexBFS

Easy application

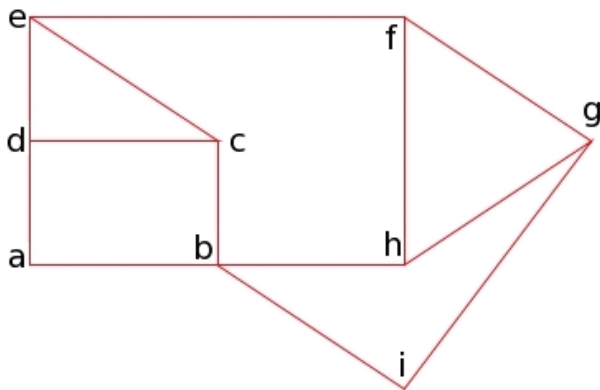


Figure: Graph $G = (V, E)$

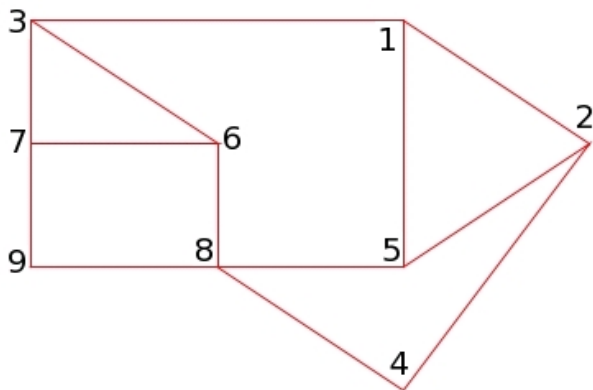


Figure: Application of
Lex-BFS on G

$\sigma(\alpha)$	α	$\mathcal{N}(\alpha)$	cells
			a b c d e f g h i
9	a	b d	b d — c e f g h i
8	b	c h i	d — c h i — e f g
7	d	c e	c — h i — e — f g
6	c	e	h i — e — f g
5	h	f g	i — e — f g
4	i	g	e — g — f
3	e	f	g — f
2	g	f	f
1	f		

With partition refinement no need to manage the labels !