Partition refinement and graph decomposition

Michel Habib, Christophe Paul, Laurent Viennot

Partition refinement and graph decomposition

Michel HABIB* Christophe PAUL* Laurent VIENNOT†

This lecture presents some new applications to graph decomposition of a very simple vertex partitioning procedure. Such a procedure allows us to put in the same framework a broad range of applications. The most famous may be the quick-sort.

But the first to use the partition refinement technique may be Hopcroft [7, 1] to design an algorithm for the *deterministic automata minimization*. Later Paigue and Tarjan [11] proposed a partition refinement based algorithm for the *coarsest relational partition* that can be viewed as a generalization of the automata minimization problem. Two other applications are also developed: the *lexicographic string sort* and the *doubly lexical ordering of boolean matrices* [8, 13].

Spinrad [12] was the first to investigate the graph partitioning field. In the last years simplifications and new developments of graph algorithms are based on the partition refinement techniques [9, 10, 5, 6]. Generalizing vertex partitioning to clique partitioning, one can produce a very simple linear interval graph recognition algorithm and a 1 consecutiveness test for boolean matrices [5, 6] avoiding the PQ-tree data-structure [2].

Although this applications are very different, they all rely on partition refinement techniques. To gather them in the same framework, we have developed a generic procedure that can be easily instantiated. Then similar correctness proofs (invariants) and complexity analysis can be done.

We propose in this lecture, as applications of this procedure two new decomposition algorithms. Although linear time modular decomposition algorithms already exist [4, 9], simple (easy to program, to maintain ...) algorithms are still missing. The algorithms we proposed constitute a major step in this direction. The first one recognizes if a graph is a cograph, the second one produces the modular decomposition tree. They both have a $O(n + m \log n)$ time bound.

These algorithms rely on the existence of a decomposition tree and compute a *factorizing permutation* [3]. This new notion leads to a very interesting new approach on the modular decomposition problem.

In a graph $G = (V, E)$, a set of vertices $M \subseteq V$ such that for any $x \in M$ and $y \in M$, $N(x) \setminus M = N(y) \setminus M$ is called a *module*. A module M is a *strong module* if and only if M does not strictly overlap any other module.

Definition 1 *[3] A* factorizing permutation *is a permutation σ of the vertices such that any strong module appear consecutively in σ.*

Given a factorizing permutation σ of a graph $G = (V, E)$, its modular tree decomposition can be computed in $O(n + m)$ [3]. So computing the tree

[1]LIRMM, UMR CNRS-Université Montpellier II, 161 rue Ada, 34392 Montpellier cedex 5, France; e-mail:{habib,paul}@lirmm.fr
[2]LIAFA, Université Paris VII

decomposition or a factorizing permutation are equivalent problem for the complexity issue.

In this lecture, we discuss some experimental results of the generic procedure and some further applications: as previously mentioned, the PQ-tree data-structure can be avoided for the interval graph recognition, is it also possible for the planarity test ? Some directions to improve the complexity bound of the decomposition algorithms will also be presented.

References

[1] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *Design and analysis of computer algorithms*. Addison-Wesley, 1974.

[2] K.S. Booth and G.S. Lueker. Testing for the consecutive ones property, interval graphs and graph planarity using pq-tree algorithms. *J. Comput. and Syst. Sciences*, 13:335–379, 1976.

[3] C. Capelle. *Decomposition de graphes et permutations factorisantes*. PhD thesis, Univ. de Montpellier II, 1996.

[4] 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.

[5] 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. *Theoritical Computer Science*, 1997. to appear.

[6] M. Habib, C. Paul, and L. Viennot. A synthesis on partition refinement: a useful routine for strings, graphs, boolean matrices and automata. In *STACS 98*, number 1373 in LNCS, pages 25–38. 15th Symposium on Theoretical Aspect of Computer Science, 1998.

[7] J.E. Hopcropft. A nlogn algorithm for minizing states in a finite automaton. *Theory of Machine and Computations*, pages 189–196, 1971.

[8] A. Lubiw. Doubly lexical orderings of matrices. *SIAM J. of Algebraic Disc. Meth.*, 17:854–879, 1987.

[9] R.M. McConnell and J.P. Spinrad. Linear-time modular decomposition and efficient transitive orientation of comparability graphs. In *Proc. of SODA*, pages 536–545, 1994.

[10] R.M. McConnell and J.P. Spinrad. Linear-time modular decomposition and efficient transitive orientation of undirected graphs. In *Proc. of SODA*, 1997.

[11] R. Paige and R.E. Tarjan. Three partition refinement algorithms. *SIAM Journ. of Comput.*, 16(6):973–989, 1987.

[12] J.P. Spinrad. Graph partitioning, 1986. unpublished.

[13] J.P. Spinrad. Doubly lexical ordering of dense 0-1 matrices. *Information Processing Letters*, 45:229–235, 1993.