A FAST PARALLEL ALGORITHM FOR COMPUTING ALL MAXIMAL CLIQUES IN A GRAPH AND THE RELATED PROBLEMS (EXTENDED ABSTRACT)

ELIAS DAHLHAUS
AND
MAREK KARPINSKI *

DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF BONN, 5300 BONN 1

Abstract.

We design a fast parallel algorithm for determining **all** *maximal cliques (maximal independent sets)* in an arbitrary graph, working in $O(\log^3(nM))$ parallel time and $O(M^6 n^2)$ processors on a CREW-PRAM, where n is the number of vertices and M the number of maximal cliques. It entails the existence of deterministic NC-algorithms for several important graph classes with a polynomially bounded number of *maximal cliques (maximal independent sets)* in the number of vertices. Our result surprisingly generalizes the recent fast NC-algorithms of [NNS] and [DK 1] for computing all maximal cliques on chordal graphs to the arbitrary classes with polynomially many *maximal cliques*. Examples of these important classes of graphs besides chordal and strongly chordal graphs [NNS], [DK] are circle and circular graphs [Go], [GHS], $K_4\backslash e$ graphs, circular arc graphs, expander graphs, and edge graphs [Ga]. They arise in a number of applications [Ga], [TIAS], [MC], [GMS].

All computational solutions for the set of all maximal cliques or maximal independent sets up to now were inherently sequential and strongly restraining efficient parallelization [TIAS], [CN]. Our result implies that the problem of finding the maximum clique or the lexicographically first maximal clique is efficiently parallelizable for every class of graphs with polynomially many cliques. It stands in contrast to the status of these problems for an unbounded case (NP-completeness and P-completeness [Co]). It also provides another class of problems ([GK]) with superpolynomial (exponential) monotone lower bound complexity [AB], [Ra], and within the uniform Boolean circuits of $O(\log^3 n)$ depth and polynomial size. The following general enumeration problem has also been proved to be in NC: Given

*Supported in part by Leibniz Center for Research in Computer Science and the DFG Grant KA 673/2-1

an arbitrary graph G, and a natural number K in unary, determine K cliques of G or determine there are less than K cliques in G. We apply the new universal algebra method of the Galois connection for the lattice structure of bipartite complete graphs and the recent completeness results on such lattices.

1. Introduction.

Several important graph classes have a number of cliques which is polynomially bounded by the number of vertices. The best known examples are chordal graphs [Go], circular arc graphs, and edge graphs [Ga]. For these classes polynomial time algorithms are known which compute the set of all cliques. A first general algorithm which enumerates all cliques of a graph and executable in polynomial time with respect to the number of vertices and the number of cliques was the algorithm of Bierstone [TIAS]. For further developments in this direction see also [CN]. Quite recently fast parallel algorithms were discovered which compute the set of all cliques in the chordal graphs [NNS], [DK 1]. Surprisingly, generalizing these algorithms, we present here a parallel algorithm which enumerates the cliques of an arbitrary given graph. In Section 2, we give basic definitions and known fundamental results. Section 3 presents a global description of the divide-and-conquer algorithm. Section 4 outlines the fine structure of the algorithm (based on the Galois connection). Section 5 discusses some possible applications and related research topics.

2. Basic Definitions and Results

A graph $G = (V, E)$ consists of a set V of *vertices* and a set E of *edges*. A (maximal) *clique* of G is a maximal (w.r.t. to set theoretic inclusion) complete subgraph of G. In what follows, a clique will be identified with the set of its vertices.

The class of computation problems computable by a log space uniform sequence of Boolean circuits of $O(\log^k n)$ depth and of polynomial size is denoted by NC^k. $NC = \bigcup_k NC$ is identical with the class of problems solvable by parallel random access machines (PRAM) in polylog time and in polynomially bounded number of processors. In this paper we shall employ the model of the concurrent read/exclusive write parallel random access machine (CREW-PRAM). We denote the *number of vertices* by n, the *number of edges* by m, and the *number of cliques* by M.

The basic result on the sequential complexity of computing all cliques is the following

Theorem 1([CN], [TIAS]): There is an algorithm which computes the set of all cliques of any graph and which needs $O(n + m)$ space and $O((n \cdot m)M)$ time.

In the next section we will present a most global description of our parallel algorithm.

3. Global Description of a Parallel Algorithm

We assume that $G = (V, E)$ and $V = \{v_1, \cdots, v_n\}$. We start with the top-most level description of the algorithm.

Algorithm:
Input: $(V, E),\ V = \{v_1, \cdots, v_n\}$.

Procedure $CLIQUE(V, E)$ (=set of cliques of $G = (V, E)$).
If $|V| = 1$ **then** $CLIQUE(V, E) := \{V\}$ **else**

begin:
Construct G_1 to be the subgraph of G induced by $\{v_1, \cdots, v_{\lceil n/2 \rceil}\}$
Construct G_2 is the subgraph of G induced by $\{v_{\lceil n/2 \rceil +1}, \cdots, v_n\}$

Do in parallel:
$U := CLIQUE(G_1)$ (=set of cliques of G_1)
$W := CLIQUE(G_2)$ (=set of cliques of G_2)

For each $u \in U$,$v \in W$ do

begin:
Procedure $COMP\_MAX(D_{u,v})$
$(D_{u,v} := \{c \subseteq u \cup v : c$ is complete and maximal in G restricted to $u \cup v\})$
$E_{u,v} := \{c \in D_{u,v} : c$ is a clique in $G\}$
end

$CLIQUE(C) := \bigcup_{\substack{u \in U \\ v \in V}} E_{u,v}$
end
end Procedure $CLIQUE$
Output $CLIQUE(V, E)$

The Correctness of the Algorithm

Let G_1 and G_2 be defined as in the algorithm and V_1 and V_2 be their vertex sets, respectively. Let c be a clique of $G = (V, E)$. Then $c \cap V_1$ and $c \cap V_2$ are subsets of cliques of V_1 and V_2, respectively. Call these cliques u and v, respectively. Then $c \in D_{u,v}$ and therefore $c \in E_{u,v}$.

A First Remark on the Complexity

Cliques can be checked by a CREW-PRAM in time $O(\log n)$ using $O(n^2)$ processors. The recursion depth of the procedure is $\lceil \log^2 n \rceil$.

We have to check the parallel complexity of the computation of $D_{u,v}$, Procedure $COMP\_MAX(D_{u,v})$.

4. The Fine Structure of the Algorithm: The Computation of the Set $D_{u,v}$ of Maximal Complete Subgraphs of G Restricted to $u \cup v$

The complete sets u and v are disjoint. Each maximal complete subgraph of G restricted to $u \cup v$ corresponds to a maximal complete bipartite subgraph of the bipartite graph $(u \cup v, E')$, where E' is the set of edges of E which join any vertex of u to a vertex of v. In [Wi] we find the following result:

Theorem 2 [Wi]: The maximal complete bipartite subgraphs of a bipartite graph form a lattice structure in the sense of universal algebra.

The lattice structure is related to a Galois connection [Bi 1], [Bi 2] and is defined as follows. First, we define an auxiliary closure operator:

Let A be any subset of U. Then
$A_2 := P_2(A) = \{x \in v : \forall y \in A \, \{y, x\} \in E'\}$
and
$A_1 := P_1(A_2) := A_2' := \{y \in u : \forall x \in A_2 \, \{y, x\} \in E'\}$

Observation: $A_1 \cup A_2$ forms a maximal complete bipartite subgraph, and all maximal complete bipartite subgraphs are of this form.

Now we define the lattice operations $\vee$, $\wedge$:

$A_1 \dot\cup A_2 \vee B_1 \dot\cup B_2 := (A_1 \cup B_1)_1 \dot\cup (A_2 \cap B_2) = P_1(A_2 \cap B_2) \dot\cup (A_2 \cap B_2)$
$A_1 \dot\cup A_2 \wedge B_1 \dot\cup B_2 := (A_1 \cap B_1)_1 \dot\cup (A_1 \cap B_1)_2 = (A_2 \cap B_2) \dot\cup P_2(A_2 \cap B_2)$

We observe the following:

Lemma 1: For any maximal complete bipartite subgraph $A_1 A_2$ of $(u \dot\cup v, E')$, we have
$A_1 \dot\cup A_2 = \bigvee_{a \in A_1} (\{a\}_1 \dot\cup \{a\}_2)$.

We can now state the following algorithm to compute all maximal complete bipartite subgraphs of $(u \dot\cup, E')$.

Procedure $COMP\_MAX(D_{u,v})$

1) $i := 0, U_0 := \{\emptyset_1 \dot\cup \emptyset_2\} \cup \{\{a\}_1 \dot\cup \{a\}_2 : a \in u\}$
2) **Repeat:** $i := i + 1$
   $U_i := UNION(U_{i-1}) := \{r \vee s : r, s \in U_{i-1}\}$
   Until $U_i = U_{i-1}$
3) Output $D_{u,v} := U_i$

Analysis of the Algorithm

By induction it is easily seen that U_i contains at least all A_1, A_2, s.t. the size of A_1 is at most 2^i. Therefore the repeat loop is executed at most $O(\log n)$ times.

The computation of A_1 and A_2 from A needs $O(\log n)$ time and $O(n^2)$ processors. This is also true for the computation of $\vee$. Clearly the size of U_i is bounded by M.

The computation of U_i from U_{i-1} has to be partitioned into the following subprocedures:

1) For each $s, t \in U_{i-1}$ compute $s \vee t$;
2) erase duplicates in U_i;
3) press U_i into an array of length of at most M by sorting.

1) • can be executed in $O(\log n)$ time by $O(M^2 n^2)$ processors;
2) • can be executed in $O(\log n)$ time by $O(M^4 n)$ processors;
3) • can be done in $O(\log M)$ time by $O(M^2)$ processors (see for example [Hi], [Cl]).

Consequence: U_i can be computed from U_{i-1} in time max $(O(\log n), O(\log M))$ by $\max(O(M^4 n),\, O(M^2 n^2))$ processors.

We can conclude with the following:

Theorem 3: The set of all cliques of any graph can be computed by a CREW-PRAM in time $\max(O(\log n)^3,\, O(\log M)^3)$ in $\max(O(M^6 n),\, O(M^4 n^2))$ processors.

An extended analysis of the algorithm allows us to solve the following problem in NC:

Input: A graph G and a natural number K in unary description
Output: K cliques of G, if they exist; otherwise the information "there are less than K cliques".

Sketch of an Algorithm:

We consider the algorithm which computes the set of all cliques of a graph. We start the algorithm and stop as soon as we have a section of the divide-and-conquer method which has K or more cliques. We extend these cliques of the section to the cliques of the whole graph by one of the known MIS-algorithms (see [Lu], [GS]).

5. Possible Applications and Related Research Topics

The immediate consequence of the results of this paper is that the problems of computing all *cliques* (and *maximal independent sets*) are efficiently parallelizable for several important classes of graphs. The results also entail the existence of uniform Boolean circuits of $O(\log^3 n)$ depth and poly-size for computing all cliques for arbitrary classes of graphs, provided the number of cliques is bounded by a polynomial. This seems to be related to the recent results of [GK] on parallel enumeration of all perfect matchings in bipartite graphs with polynomially bounded permanents.

A related problem is a fast parallel clique decomposition of a graph. [Ta] has designed an algorithm depending on a highly sequential subroutine for computing minimal orderings. Since the number of clique separators in an arbitrary graph is polynomial in the number of vertices, one can ask for the fast parallel enumerator of all the clique separators. We have been able to put the problem of clique separators in NC, and therefore also the problem of the *clique decomposition* of an arbitrary graph. This is connected to the general problem of the elimination orderings, and the problem of Gaussian elimination on sparse matrices. We shall deal with these topics in detail in the final version of this paper.

In the context of our results for the various subclasses of perfect graphs, the general question of parallel computation of the maximum clique or *maximum independent set* for perfect graphs ([GLS]) becomes even more exciting.

Acknowledgements

We are grateful to Avi Wigderson, Eli Upfal, Noga Alon, Seffi Naor, and Alex Schäffer for many stimulating discussions which were starting points for the present paper.

References

[AB] Alon, N., and Boppana, R.B., *The Monotone Circuit Complexity of Boolean Functions*, Manuscript, MIT 1986

[AM] Auguston, J.M. and Minker, J., *An Analysis of Some Graph Theoretical Cluster Techniques*, J. ACM 17(1970), pp. 571-588

[Bi 1] Birkhoff, G., *Subdirect Unions in Universal Algebra*, Bull. Amer. Soc. 50(1944), pp. 764-768

[Bi 2] Birkhoff, G., *Lattice Theory*, 3^{rd} ed. Amer. Soc., Providence 1967

[CN] Chiba, N., and Nishivuki, T., *Arboricity and Subgraph Listing Algorithms*, SIAM J. of Comput. 14(1985), pp. 210-223

[Cl] Cole, R., *Parallel Merge Sorting*, Proc. 27^{th} IEEE FOCS (1986), pp. 511-516

[CV] Cole, R., and Vishkin, U., *Approximate and Exact Scheduling with Applications to List, Tree and Graph Problems*, Proc. 27^{th} IEEE FOCS (1986), pp. 478-491

[Co] Cook, S.A., *A Taxonomy of Problems with Fast Parallel Algorithms*, Information and Control 64 (1986), pp. 2-22

[DK 1] Dahlhaus, E., and Karpinski, M., *The Matching Problem for Strongly Chordal Graphs is in NC*, Research Report No. 855-CS, Department of Computer Science, University of Bonn 1986

[DK 2] Dahlhaus, E., and Karpinski, M., *Fast Parallel Computation of Perfect and Strongly Perfect Elimination Schemes*, IBM Research Report # RJ 5901 (59206), IBM Almaden Research Center, San Jose 1987; submitted for publication

[GHS] Gabor, C.P., Hsu, W.L., and Supowit, K,J., *Recognizing Circle Graphs in Polynomial Time*, Proc. 26^{th} IEEE FOCS (1985), pp. 106-116

[GJ] Garey, M.R., and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman:San Francisco 1979

[Ga] Gavril, F., *Algorithms for Minimum Coloring, Maximum Clique, Minimum Coloring by Cliques, and Maximum Independent Sets of a Chordal Graph*, SIAM J. Comput. (1972), pp. 180-187

[GS] Goldberg, M., and Spencer, T., *A New Parallel Algorithm for the Maximal Independent Set Problem*, Proc. 28^{th} IEEE FOCS (1987), pp. 161-165

[Go] Golumbic, M.C., *Algorithmic Graph Theory and Perfect Graphs*, Academic Press, New York 1980

[GK] Grigoriev, D.Yu., and Karpinski, M., *The Matching Problem for Bipartite Graphs with Polynomially Bounded Permanents is in NC*, Proc. 28^{th} IEEE FOCS (1987), pp. 166-172

[GLS] Grötschel, M., Lovász, L., and Schrijver, A., *The Ellipsoid Method and its Consequences in Combinational Optimization*, Combinatorica 1(1987), pp. 169-197

[Hi] Hirschberg, D., *Fast Parallel Sorting Algorithms*, Communications of the ACM 21(1978), No. 8, pp. 657-661

[Lu] Luby, M., *A Simple Parallel Algorithm for the Maximal Inpedendent Set Problem*, Proc. 17^{th} ACM STOC (1985), pp. 1-9

[MC] Mulligan, G.D., and Corneil, D.G., *Corrections to Bierstone's Algorithm for Generating Cliques*, JACM 19(1972), pp. 244-247

[NNS] Naor, J., Naor, M., and Schäffer, A., *Fast Parallel Algorithms for Chordal Graphs*, Proc. 19^{th} ACM STOC (1987), pp. 355-364

[Ra] Razborov, A.A., *Bound on the Monotone Network Complexity of the Logical Permanent*, Matem. Zametk 37 (1985); in Russian

[Ta] Tarjan, R., *Decomposition by Clique Separations*, Discrete Mathematics 55(1985), pp. 221-232

[TIAS] Tsukiyama, S., Ide, M., Ariyoshi, H. and Shirakawa, I., *A New Algorithm for Generating All the Maximal Independent Sets*, SIAM J. Comput 6(1977), pp. 505-517

[Wi] Wille, R., *Subdirect Decomposition of Concept Lattices*, Algebra Universalis 17(1983), pp. 275-287