MATCHING PROBLEM FOR STRONGLY CHORDAL GRAPHS IS IN NC

by

Elias Dahlhaus

and

Marek Karpinski

University of Bonn

Computer Science Department

Introduction:

Chordal graphs have become intersting as a generalization of interval graphs (see for example [LB]) and cover a large field of applications [Go, BMFY]. That are graphs in which every cycle of of a length greater than three has a chord. We know that chordal graphs have only as many cliques as vertices. Therefore the hypergraph of cliques has a not much larger description than the given chordal graph. Farber [Fa 1, Fa 2] introduced the notion of strongly chordal graphs. That are chordal graphs with the additional property that their clique hypergraphs are acyclic (see [BDS]). Such structures have become interesting in connection to data base schemes [BMFY]. Chordal graphs are also related to elemination schemes (see for example [Go]). Elemination orderings for strongly chordal graphs are presented by M. Farber [Fa 1]. Linear time algorithms to test chordality and to compute an elemination ordering for chordal graphs are known by R. Tarjan and M. Yannakakis [TY].

On the other hand matching is generally in randomized NC (RNC) (see [MVV]). But we do not know a general deterministic NC-algorithm for matching.

Our aim is to present some parallel algorithms corresponding to chordal and strongly chordal graphs. Section 1 gives some foundations in the field of chordal graphs. In section 2 we briefly discuss the parallel complexity of testing chordality and strong chordality. In section 3 we will present an NC^2-algorithm computing a (strong) elemination order for (strongly) chordal graphs. In section 4 we will show that matching for strongly chordal graphs is in NC^2. But it is easily seen that matching restricted to chordal graphs is as difficult as the general bipartite case. A generalization of the parallel algorithm for strongly chordal graphs can have many applications as for example 2-processor scheduling (see [HM]).

Section 1: Fundamental Definitions and Results

1.1. A *chordal graph* is a graph with no induced cycle greater than three.

1.2. Being chordal is equivalent to following statement [Go, Fa 1]:
there is a (perfect) *elemination order* $<$ on the verices:
(I) If $(x,y)\in E$(=set of edges) and $(x,z)\in E$ then $(y,z)\in E$.

1.3. Farber [Fa 1] defined *strongly chordal graphs* as graphs which have a *strong* (perfect) elemination order $<$:
That means that $<$ satisfies (I) and additionally
(II): If $x<u$ and $y<v$ and additionally (x,y), (x,v), $(y,u) \in E$
then $(u,v) \in E$.

1.4. Chordal graphs were defined by forbidden subgraphs. Strongly chordal graphs can also be characterized by forbidden subgraphs: $G=(V,E)$ is strongly chordal if and only if it has no induced *trampoline* [Fa 1] or *sun* [BDS]. A trampoline or sun conists of a cycle of length at least 6 alternating between an independent set and a complete set.

1.5. Chordal graphs can also be characterized in notion of cliques:
Proposition[Di]: A graph $G=(V,E)$ is chordal iff for each induced subgraph of G one can find a *simplicial* vertex x, that means its neighborhood $N(x)=\{y:y=x \text{ or } (y,x)\in E\}$ is complete.
Farber [Fa 1] proved a corresponding result for strongly chordal graphs:
Proposition: G is strongly chordal iff each induced subgraph has a *simple* vertex x, that means $\{N(y):y\in N(x)\}$ can totally be ordered by inclusion.

1.6. The number of cliques (nonextendible complete subgraphs) of a chordal graph is at most as large as the number of vertices. The hypergraph of cliques of a strongly chordal graph is of interst. We can give the following statement:
Proposition[BDS]: The hypergraph of cliques of a graph G is acyclic if G is strongly chordal.

Section 2: Testing Chordality and Strong Chordality is in CoNL

The following statement is easily checked:

Proposition 1: G is not chordal iff there is a cycle C, s.t. for directely consequting x,y,z $(x,z) \notin E$.

That means

Corollary: Chordality is in CoNL.

Now [DD] gave a characterization of strongly chordal graphs which is also a reduction of strong chordality test to chordality test:

Proposition 2 [DD]: Let E^2 be the set of unordered pairs of vertices which have a distance of at most two and which are unequal. Then (V,E) is strongly chordal iff (V,E^2) is chordal and (V,E) does not contain the *Hajos graph* as an induced subgraph: The Hajos graph is the trampoline of a clique and an independent set both of size three.

Corollary: Strong chordality can be tested in CoNL.

We can say that testing (strong) chordality is in NC^2.
In the next section we solve the corresponding construction problems, that means to construct a (strong) elemination order.

Section 3: Computing (Strong) Elemination Orders

At first a remark:
An ordering on the vertices is a (strong) elemination ordering iff each vertex v is simplicial (simple) in the subgraph induced by all vertices w greater or equal v.

Let $G_0=G$ and $G_{i+1}=G_i-\{x: x \text{ simplicial (simple) in } G_i\}$.
Then each ordering, s.t. $x \notin G_i$ and $y \in G_i$ implies x less than y, is a (strong) elemination ordering. It suffices to test "$x \in G_i$" in NC^2.
Then we have also an algorithm computing a (strong) elemination order.

THEOREM 1: "$x \in G_i$" can be tested by an alternating logspace machine with polynomial tree size.

By [Ru] we can conclude that above problem can be tested in NC^2.
Before we prove the theorem some remarks:
For a strongly chordal graph we call a *clique simple* iff the intersections with other cliques can be ordered by inclusion. Clearly
x is a simple vertex iff it is simplicial and its unique clique is simple.
Let S be the set of all cliques of a strongly chordal graph G..We can define analogeously:

$S_0=S$ and $S_{i+1}=S_i-\{s:s \text{ simple in } S_i\}$. Clearly $x\in G_i$ iff $x\in s\in G_i$ for some s.

We can state the following

Lemma:For a strongly chordal graph we can find for each clique s an edge appearing only in s.

Therefore every clique of a strongly chordal graph is generated by an edge. That means: The set of cliques of a strongly chordal graph can be generated by a uniform sequence of circuits having unbounded fan in, constant depth and polynomial size. Therefore

Corollary:The set of cliques of any strongly chordal graph can be computed in NC^1.

For $x \in G_{i+1}-G_i$ resp. $S_{i+1}-S_i$ let C_x be the clique generated by x in G_i resp. the set of cliques $s \in S_i$ which intersect x nonempty.

Now we can define a tree structure T_G resp. T_S on $\{C_x:x\in G \text{ (resp. } S)\}$. The immediate successor of C_x is the unique C_y, s.t. $y\in C_x$ and y simplicial in G_j resp. simple in S_j and $j>i$ minimal. Note that in the chordal case the collection of $T_x:=\{C_y:x\in C_y\}$ defines a collection of subtrees of T_G, and the intersection graph of the T_x is again G (compare [Ga]). We can observe that $x \in G_i$ iff there are two disjoint paths on T_G from C_x to a leaf. The analogeous observation is also true for $x \in S_i$.

The alternating polynomial tree size algorithm is based on such tree structures. We have to explain how to handle with C_x which have only one predecessor.

Let $(C_i)_{i=1}^n$ be a sequence of C_x, s.t. C_i is the unique predecessor of C_{i+1}. In the strongly chordal case we have consequently a sequence $(s_i)_{i=1}^n$, s.t. for i=1...n-1 $s_i\cap s_{i+1}$ and $s_i\cap s_{i-1}$ are incomparable with respect to the set theoretic inclusion.

In the chordal case we get a sequence $(x_i,u_i)_{i=0}^{n-1},x_n$, s.t.

(x_i,x_{i+1}), $(u_i,u_{i+1}) \in E$ and $(x_i,u_i) \notin E$, $(x_{i+1}\ u_i) \in E$.

We call such sequences *chains* of *length* n.from s_0 resp. x_0 to s_n resp. x_n.

If C_x has at least two predecessors C_y and C_z then there are two possibilities:

a) (y,x'), $(z,x') \in E$, $C_x=C_{x'}$

b) (y,x_1), $(z,x_2) \in E$, $C_x=C_{x_1}=C_{x_2}$ but (y,x_2), $(z,x_1) \notin E$

In the strongly chordal case we have the corresponding situation:

$y\cap x_1$, $z\cap x_2$ noncomparable and $x_1=x_2$ or $y\cap x_1, x_1\cap x_2$ noncomparable and $z\cap x_2$ and $x_1\cap x_2$ noncomparable.

Now we can present an alternating logspace algorithm of *polynomial tree size* checking whether $x \in G_i$ resp. S_i:

begin
for each vertex x '(clique x)' set $x \in G_0$ '(S_0)'
repeat
for each i,j,k≤|G|
for each $x \in G_i$, $y \in G_j$
for each x_1, u_1, u_2, y_1, s.t.
$(x_1, u_1), (y_1, u_2) \in E$ and $((u_1,u_2) \in E$ or $u_1=u_2)$ and $(x_1,y_1) \notin E$
'($x_1 \cap u_1$, $y_1 \cap u_2$ incomparable and ($u_1=u_2$ or $u_1 \cap u_2$ incomparable with $x_1 \cap u_1$ and $y_1 \cap u_2$)'
and
if there is a sequence $\bar{s}=(x_1,u_1)_{1=0}^{k-i}$, s.t. $x=x_0$ and a sequence $\bar{t}=(y_1,u_1)_{1=0}^{k-j}$ s.t. $y=y_0$
and $\bar{s}x_1, \bar{t}y_1, \bar{s}u_1, \bar{t}u_2$ are chains or($x=x_1$ and $y=y_1$ and $i=j=k$) then set $u_1, u_2 \in G_{k+1}$
'(if there is a chain of length k-i via x_1 from x to u_1 and a chain of length k-j via y_1 to u_2 from y then set $u_1, u_2 \in S_k$)'

until one does not get new statements of the form "$x \in G_i$".

Statements in '( )'-delimiters are alternatives for the strongly chordal case.

Corollary: It is possible to construct a structure of subtrees of a tree representing a given chordal graph in NC^2.

Section 4: Matching on Strongly Chordal Graphs

We begin with a
Remark: Let G be a bipartite graph s.t. its two partitions V_1 and V_2 have the same power. Let G' be the chordal graph which arises from G by making V_2 to a clique. Then G has a perfect matching iff G' has a perfect matching. That means:
Matching on chordal graphs is as hard as matching on bipartite graphs.
But for strongly chordal graphs we get the following result:
THEOREM 2: Matching on strongly chordal graphs is in NC^2.
Sketch of the proof: We use the following
Lemma[MVV]: For a graph labeled by unary numbers a minimal perfect matching can be computed in NC^2 provided it is unique.

For the case of strongly chordal graphs we label the edges by the *square of distances related to a strong elemination order.*

We call a pair of edges $(u_1,u_2),(v_1,v_2)$ a *defect* iff $(u_1,v_1) \in E$ and $u_1<v_2$ and $v_1<u_2$. Suppose a matching has a defect as above. But by the fact that < is a strong elemination ordering, $(u_2v_2) \in E$. We can *remove* the defect by taking $(u_1,v_1)$and

(u_2,v_2) into the matching. The key is the following
Lemma 1: Removing a defect deminuishes the sum of labels of a matching.
Lemma 2: There is a unique defect free matching.
From this two lemmas the theorem is easily checked.

Corollary: Perfect matching on chordal bipartite graphs is in NC^2.

Section 5: A Remark on Future Research

The above result includes the matching problem restricted to interval graphs [KVV], which is included again by 2-processor scheduling [HM]. It is known that 2-processor scheduling is in NC^2[HM]. But the complement of a comparability graph need not be chordal. Only in the case of chordality we have an interval graph [GH]. But the techniques of [HM] give us a hint how to generalize our result and Helmbold and Mayr´s result to a common upper class of graphs. It might suffice that we have a partial order on the vertices, s.t. (II) is fulfilled and pairs of vertices not joined by an edge are comparable.

References:

[BFMY] C.Beeri, R. Fagin, D. Maier, M. Yannakakis, *On the Desirability of Acyclic Data Base Schemes*, JACM 30, p. 479-513 (1983)

[BK] A.Brower, A. Kolen, *A Super-Balanced Hypergraph has a Nest Point*, Report ZW 146, Mathematisch Centrum, Amsterdam (1980)

[Bu] A. Buneman, *A Characterization of Rigid Circuit Graphs*, Discrete Math. 9, p. 2o5-212 (1974)

[BDS] A. Brouwer, P. Duchet, A. Schrijver, *Graphs whose Neighborhoods have no Special Cycle*, Discrete Math. 47, p. 177-182 (1983).

[DD] E. Dahlhaus, P. Duchet, unpublished manuscript

[Di] G. Dirac, *On Rigid Circuit Graphs*, Abhandlungen Mathematischer Seminare der Universität Hamurg 25, p. 71-76 (1961)

[Fa 1] M. Farber, *Characterizations of Strongly Chordal Graphs*, Discrete Math. 43, p. 173-189(1983)

[Fa 2] M. Farber, *Applications of L.P-Duality to Problems Involving Independence and Domination*, Ph.D. Thesis, Computer Science Department, Rutgers University, New Brunswick, NJ 1982

[Ga] F. Gavril, *The Intersection Graphs of Subtrees of a Tree are Exactly the Chordal Graphs*, J. Combinatorial Theory Ser. B 16, p. 47-56 (1974)

[GH] P. Gilmore, A. Hoffman, *A Characterization of Comparability Graphs and of Interval Graphs*, Can. J. Math. 16, p. 539-548 (1964)

[HM] D. Helmbold, E. Mayr, *Two Processor Scheduling is in NC*, in *VLSI Algorithms and Architectures* (Makedon et al. ed.), LNCS 227, p. 12-25

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

[KVV] D. Kozen, U. Vazirani, V. Vazirani, *NC-Algorithms for Comparability Graphs, Interval Graphs and Testing Unique Perfect Matching*, to appear

[LB] C. Lekkerkerker, D. Boland, *Representation of Finite Graphs by a Set of Intervals on the Real Line*, Fund. Math. 51, p. 45-64 (1962)

[MVV] K. Mulmuley, U. Vazirani, V. Vazirani, *A Parallel Algorithm for Matching*, to appear

[Ru] W. Ruzzo, *Tree Size Bounded Alternation*, JCSS 21, p. 218-235 (1980)

[TY] R. Tarjan, M Yannakakis, *Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclity of Graphs, Test Acyclity of Graphs, Test Acyclity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs*, SIAM J. Comp. 13, p. 566- 579(1984)