ON THE PARALLEL COMPLEXITY OF MATCHING FOR CHORDAL AND PATH GRAPHS

ELIAS DAHLHAUS

AND

MAREK KARPINSKI

DEPT. OF COMPUTER SCIENCE
UNIVERSITY OF BONN

Introduction.

Chordal graphs have become interesting as a generalization of interval graphs (see for example [LB]) and cover a large field of applications [Go, BMFY]. These are graphs in which every cycle with a length greater than three has a chord. We know that chordal graphs have only as many cliques as vertices. Therefore the description of a hypergraph of cliques is not much larger than that for the given chordal graph. Farber [Fa 1, Fa 2] introduced the notion of strongly chordal graphs with the additional property that their clique hypergraphs are β-acyclic (see [BDS]). Such structures have become interesting in connection with data base schemes [BMFY]. Chordal graphs are also related to elimination schemes (see for example [Go]). Elimination orderings for strongly chordal graphs are presented by M. Farber [Fa 1]. Linear time algorithms to test chordality and to compute an elimination 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 [KUV] and [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. We develop a new fast parallel algorithm for the construction of strong perfect elimination ordering in strongly chordal graphs. The same idea was used in [DK 2] to get a parallel algorithm a bit more efficient than those in [NNS] or [DK 1] for the computation of perfect elimination ordering for chordal graphs (the algorithm works in $O(\log^2 n)$ parallel time and $O(n^3)$ processors). The exact analysis of a number of processors for a strong perfect elimination and matching algorithm will be given in a final version of this paper.

Section 1 gives some foundations in the field of chordal graphs. In section 2 we discuss briefly the parallel complexity of testing chordality and strong chordality. In section 3 we will discuss the parallel complexity of computing a (strong) elimination 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, for multi-processor scheduling problems (for 2-processor scheduling see, e.g., [HM]).

The important feature of our results is its meaning towards the general matching problem. The results are settling a precise new borderline between what is known to be efficiently parallelisable and NC-hard for the general matching problem: Perfect matching for chordal and undirected path graphs are 'NC-hard' for general bipartite perfect matching whereas our algorithms are putting matching problems for strongly chordal and directed path graphs in NC^2 (within $O(\log^2 n)$ parallel time and $O(n^{6.5})$ processors).

Section 1: Basic 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 the following statement [Go, Fa 1]:
there is a (perfect) *elimination order* $<$ on the vertices:
(I) If $(x,y) \in E$ (= set of edges) and $(x,y) \in E$ and $x < y$ and $x < z$, then $(y,z) \in E$.

1.3. Farber [Fa 1] defined *strongly chordal graphs* to be graphs which have a *strong* (perfect) elimination 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* consists of an even cycle of length at least 6 alternating between an independent set and a complete set.

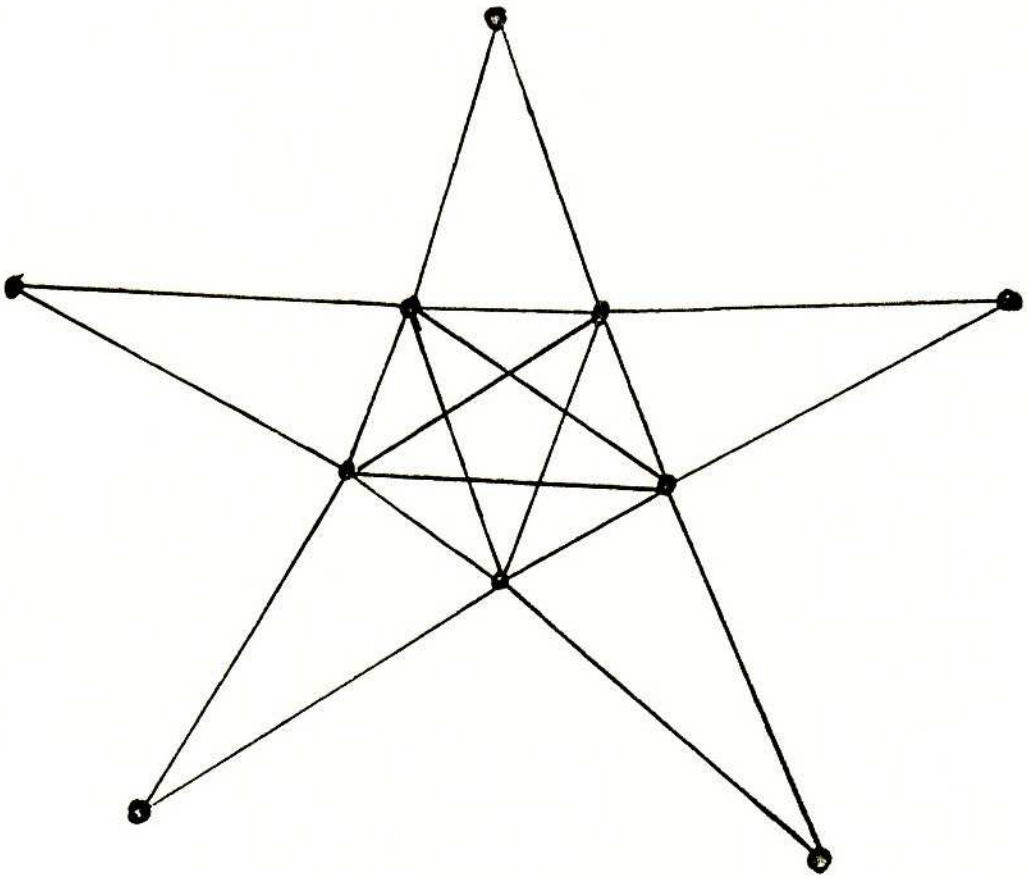


Figure 1: A trampoline with $2k = 10$ vertices

1.5. Chordal graphs can also be characterized in notion of cliques:

Proposition 1 [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 neighbourhood $N(x) = \{y : y = x \text{ or } (y, x) \in E\}$ is complete. Farber [Fa 1] proved a corresponding result for strongly chordal graphs:

Proposition 2: G is strongly chordal iff each induced subgraph has a *simple* vertex x; that means $\{N(y) : y \in N(x)\}$ can be totally 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 interest. We can make the following statement:

Proposition 3 [BDS]: The hypergraph of cliques of a graph G is β- acyclic if G is strongly chordal.

We say, a hypergraph H is β-acyclic if there is no "cycle" x_1h_1, x_2h_2,...,h_kx_{k+1}, s.t. $x_i \in h_{i-1} \cap h_i \bmod k$, and $x \notin h_j, j \neq i-1, i \bmod k$ (compare [BMFY]).

Section 2: Testing Chordality and Strong Chordality is in $CoNL$

The following statement follows easily from a characterization of chordal graphs in [TY 2] (Compare also the chordality test of [NNS]):

Proposition 4: Chordality is in $CoNL$.

To test the strong chordality we state the following

Lemma 5: For a strongly chordal graph we can find for each clique s an edge appearing only in s.

PROOF. Assume there is a clique C which is not generated by an edge. We want to construct a trampoline. Consider any vertex v in C. Then there is a clique $C_0' \neq C$, s.t. $v \in C_0'$. Otherwise each $[v, w]$, s.t. $v, w \in C$ would generate C. We may assume that there is no clique C'', s.t. $C_0' \cap C \subseteq C'' \cap C$ and $C \neq C''$ (maximality condition). Consider any vertex $w \in C \setminus C_0'$ and a clique C_1', s.t. $v, w \in C_1'$. By the maximality condition we can also find the vertex $u \in C_0' \cap C \setminus C_1'$. Find a clique $C_2' \neq C$,
s.t. $u, w \in C_2'$; it follows from the acyclic property of the clique hypergraph that $C_i' \cap C_j' \subseteq C$, $i, j = 0, 1, 2$. Now consider arbitrary vertices $c_i \in C_i' \setminus C$ for $i = 0, 1, 2$. Then c_i, c_j are not joined by an edge. Otherwise there would be clique C_{ij}, s.t. $c_i, c_j \in C_{ij}$ and C_{ij}, C_i', C_j' would form a cycle in the cycle hypergraph.

But c_0, v, c_1, w, c_2, u form a trampoline, in contradiction to the assumption that the given graph is strongly chordal.

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

Corollary 7: Strong chordality can be tested in $CoNL$.

We can say that testing (strong) chordality is in NC^2.

In the next section we give a hint to the solution of the construction problem; that means we construct a (strong) elimination order.

Section 3: Computing (Strong) Elimination Orders

We first note:

An ordering on the vertices is a (strong) elimination ordering iff each vertex v is simplicial (simple) in the subgraph induced by all vertices w greater or equal to v.

Theorem 8: A (strong) perfect elimination ordering of a (strongly) chordal graph can be computed in NC^2. For the case of chordal graphs we refer to [NNS]. By a similar argument we can test β-acyclity in $CoNL$.

For a strongly chordal graph we call a ***clique*** a ***simple clique*** 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.

For the case of strongly chordal graphs we shall give an informal desciption of an NC^2-algorithm. A detailled version is to be found in [DK].

We shall construct a strongly perfect elimination ordering on the hypergraph of the cliques. The ordering $<$ we want to construct satisfies the following additional constraint:

If $x, y < z$ and $0 \neq x \cap z \subsetneq y \cap z$ then $x < y$ (comparison axiom).x,y,z stand for hyperedges.

It suffices to construct a partial ordering $<$, s.t. each intersecting pair of hypergraphs is comparable. We choose a maximum m and say $x <'_m y$ iff there is a *chain* from x to m via y, that means there is a sequence ($x_0 := x$, $x_1 ... x_i := y$, $x_{i+1} ... x_k = m$) s.t. $x_{i-1} \cap x_i$ and $x_i \cap x_{i+1}$ are incomparable by inclusion.

$<'_m$ defines a partial ordering which satisfies the axioms of a strongly perfect elimination ordering. Set $x\tilde{<}_m y$ iff there is a z s.t. $x, y <_m z$ and $0 \neq x \cap z \subsetneq y \cap z$.

Let $<_m$ be the transitive closure of $\tilde{<}_m \cup <'_m$.

Lemma 9 $<_m$ is a partial ordering which satisfies the axioms of a strongly perfect elimination ordering and the comparison axiom. It can easily be seen that $<_m$ can be computed in NC^2.

Let $x \sim_m y$ iff $x \cap y \neq 0$ and x, y are incomparable by $<_m$. A connected component w.r.t. $\sim_m$ is called a *heap*. For a heap H the set $D = \{x | h < x$ and $h \cap x \neq 0$ for all $h \in H\}$ is called the domination of H. Now $\cap D_H = x_1 \cap x_2$ for some $x_1, x_2 \in D_H$, $H = \{x : \cap D_H \subset x\}$. We have two possibilities for the structure of heaps:

1) $h_1 \cap \bigcap D_H = h_2 \cap \bigcap D_H$ for each $h_1, h_2 \in H$. In that case set $H' := \{h : \bigcap D_H \cap h = \bigcap D_H \cap h_1\}$ for any $h_1 \in H'$

2) It is not the case: then H contains the maximal elements of H w.r.t. the intersection with any element of D_H of a connected component of H w.r.t. nonempty intersection after the deletion of all elements of D_H.

Let H' be the set of all $\bigcap D_H$ nonempty intersecting x which are in the connected component of H after the deletion of all hyperedges of D'_H. Call H' an extended heap. We observe for $x, y \in H'$:

$x <_m y$ iff there is a $z \in D'_H$ s.t. $x \cap z \subsetneq y \cap z$.

Choose any maximal m_H of H and let $\tilde{<}_{m_H}$ be the transitive closure of $<_{m_H}$ restricted to H and $<_m$.

Now we have heaps again, but they have the same properties as above. We can determine a set of possible extended heaps and their order structure in advance and fill them out in parallel. That means it is possible to construct a partial strong elimination ordering 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 is derived from G by making V_2 a clique. Then G has a perfect matching iff G' has a perfect matching. That means:
Matching on chordal graphs is as hard as perfect matching on bipartite graphs.
But for strongly chordal graphs we get the following result:

Theorem 10: Perfect matching on strongly chordal graphs is in NC^2.

Remark: Observe that our Theorem 10 entails the General (Maximum) Matching Problem on strongly chordal graphs to be also in NC^2. The reason is that the Maximum Matching Problem is NC-reducible to Perfect Matching, and the reduction preserves strong chordality.

SKETCH OF THE PROOF: We use the following

Lemma 11 [RV], [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 elimination 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. By the fact that $<$ is a strong elimination ordering, $(u_2 v_2) \in E$. We can thus *remove* the defect by taking (u_1, v_1) and (u_2, v_2) into the matching. The key is the following

Lemma 12: Removing a defect diminishes the sum of labels of a matching.

PROOF. Consider any defect. Then we have to consider only the sum of squares or distances of the edges (u_1, u_2), (v_1, v_2) and the sum of squares of distances of the edges $(u_1, v_1), (u_2, v_2)$.

We have to consider a couple of subcases:

1st case: $u_1 < v_2 < v_1 < u_2$. Then

$(u_2 - v_2)^2 + (v_1 - u_1)^2 = (u_2 - v_2)^2 + ((v_1 - v_2) + (v_2 - u_1))^2 = (u_2 - v_2)^2 + (v_1 - v_2)^2 + 2(v_1 - v_2)(v_2 - u_1) + (v_2 - u_1)^2 < (v_1 - v_2)^2 + (u_2 - v_2)^2 + 2(u_2 - u_1)(v_2 - u_1) + (v_2 - u_1)^2 = (v_1 - v_2) + (u_2 - u_1)^2$

2nd case: $u_1 < v_1 < v_2 < u_2$. Then

$(v_1 - u_1)^2 + (u_2 - v_2)^2 < (u_2 - u_1)^2 < (u_2 - u_1) + (v_2 - v_1)^2$.

3rd case: $u_1 < v_1 < u_2 < v_2$. Then

$(v_1 - u_1)^2 + (v_2 - u_2)^2 < (u_2 - u_1)^2 + (v_2 - v_1)^2$

All other possible cases are permutations of these three cases. This completes the proof of this lemma.

Figure 2: Removing a defect

Lemma 13: There is a unique defect free matching.

PROOF: Assume we have more than one defect free perfect matching, say M and M'.

Consider any $[x_1, y_1] \in M$, $[x_1, y_0] \in M'$, $[x_0, y_0] \in M$, $[x_2, y_1] \in M'$, $[x_2, y_2] \in M$. The defect freeness of M forces $x_1 < x_2$ and $y_0 < y_1$ or both vice versa. W.l.o.g., $x_1 < x_2$. Then the defect-freeness of M' forces $y_1 < y_2$. Consider now generally x_i, y_i, s.t.$[y_{i-1}, x_i] \in M'$ and $[x_i, y_i] \in M$. By induction we can conclude $x_i < x_{i+1}$ and $y_i < y_{i+1}$. But the finiteness of the strongly chordal graph makes such a situation impossible. From these two lemmas the theorem can be easily proved.

Corollary 14: Perfect matching on chordal bipartite graphs is in NC^2.

Section 5: Connections to other graph classes

We have seen that matching restricted to chordal graphs (and also for split graphs) is as hard as bipartite matching. On the other hand perfect matching restricted to strongly chordal graphs is in NC^2. Therefore perfect matching restricted to directed path graphs and therefore to interval graphs is in NC^2. The only gap which we have to fill out is the matching problem restricted to (nondirected) path graphs.

Theorem 15: Perfect matching restricted to path graphs is as hard as perfect bipartite matching.

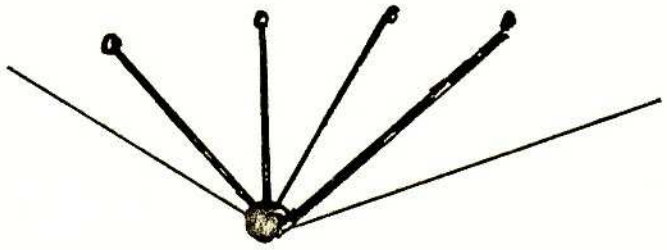
SKETCH OF THE PROOF: We reduce the perfect matching problem restricted to split graphs to the perfect matching problem restricted to path graphs.

For any split graph $G = (V, E)$ we can construct a corresponding system of subtrees of a tree in NC^1:

The global tree consists of a "central" vertex w and vertices u_v for each vertex v of the independent set I of split graph G. w is connected with each u_v. The subtree coresponding to a vertex $V \in I$ is $\{u_v\}$ and the subtree T_x corresponding to a vertex x of the clique of the split graph G is $\{w\} \cup \{u_v | [v, x] \in E\}$.

Now we construct a path graph G' in the following way:

If $|\{v \in I | [v, x] \in E\}| \leq 2$ we have nothing to do. Otherwise if $C_x := \{v \in I | [v, x] \in E\}$ has a cardinality greater than 2, we add to T a path $(w, i_{x,1}, ..., i_{x,|c_x|-1})$ and replace T_x by a collection of paths $P_v^x := (u_v, w, i_x, ..., i_x, |c_x| - 1)$, s.t. $v \in c_x$ and one element paths $\{i_{x,j}\}$. Exactly one of the p_v^x is not married with an $i_{x,j}$ and can therefore be married with u (or any other element not being a p_v^x). This path graph G' has a perfect matching iff G has a perfect matching.



is transformed to

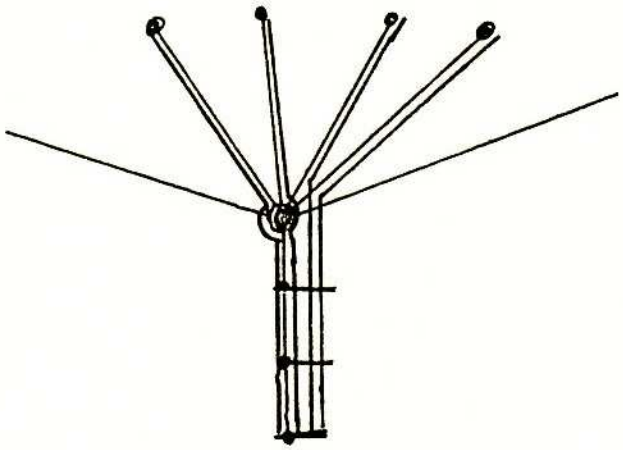


Figure 3: Transformation of a split graph to a path graph

Finally we have to remark that interval graphs are exactly the chordal complements of comparability graphs [GH]. It is known that matching restricted to complements of comparability graphs is equivalent to 2-processor-scheduling and that this is again in NC^2 [HM]. There remains the open problem of finding a meaningful upper class of strongly chordal graphs and of complements of comparability graphs, s.t.its matching problem is in NC^2.

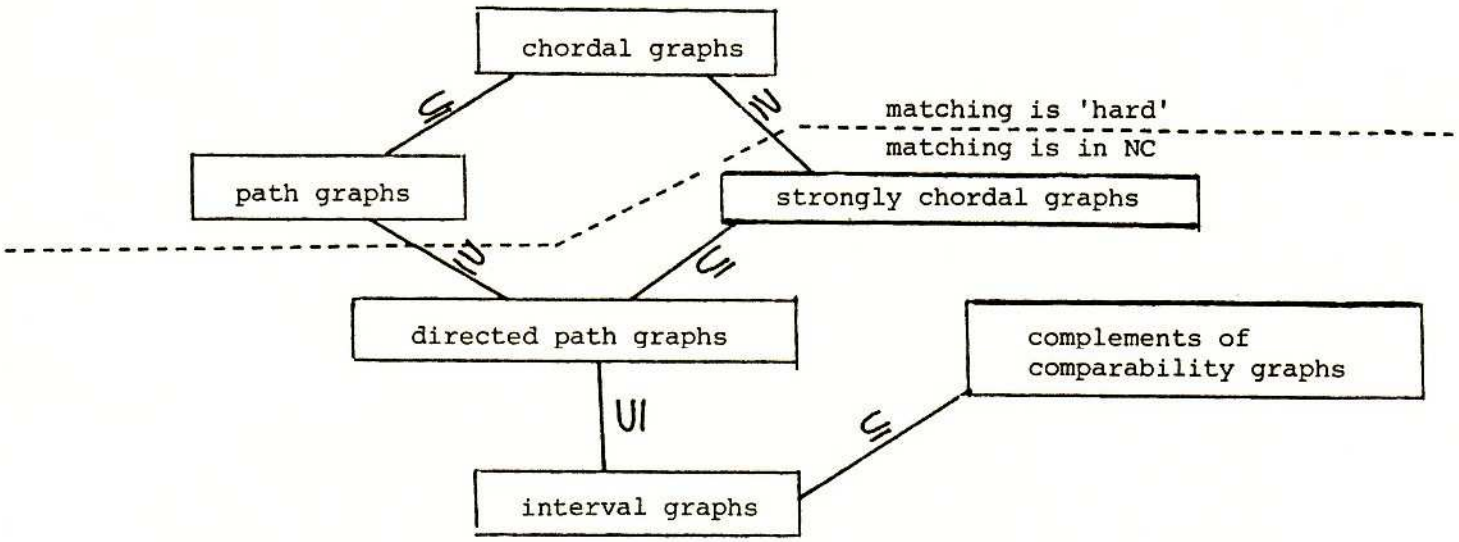


Figure 4: The inclusion structure of graph classes related to chordal graphs and the complexity of their matching problem (compare also [Jo])

Acknowledgements:

We are thankful to Avi Wigderson, Joseph Naor, Ernst Mayr and Alejandro Schaeffer for a number of interesting conversations. We thank Barbara Chapman for her comments on the final version of the paper.

References

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

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

[Bu] Buneman, A.,
A Characterization of Rigid Circuit Graphs
Discrete Math. 9 (1974), pp. 205-212

[BDS] Brouwer, A., Duchet, P., and Schrijver, A.,
Graphs Whose Neighbourhoods have no Special Cycle
Discrete Math. 47 (1983), pp. 177-182

[Co] Cook, S.A.,
A Taxonomy of Problems with Fast Parallel Algorithms
Information and Control **64** (1985), pp. 2-22

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

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

[DK 2] Dahlhaus, E., and Karpinski, M.,
Fast Parallel Computation of Perfect and Strongly Perfect Elimination Orderings
Research Report No. 8513 - CS, University of Bonn (1987)

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

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

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

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

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

[HM] Hembold, D., and Mayr, E.,
Two Processor Scheduling is in NC, in: *VLSI Algorithms and Architectures (ed. Makedon et. al.*
LNCS 227, pp. 12-15

[Jo] Johnson, D.S.,
NP-Completeness Column
Journal of Algorithms 6 (1985), pp. 434-451

[KUW] Karp, R., Upfal, E., and Wigderson, A.,
Finding a Maximum Matching in NC
17^{th} STOC (1985), pp. 22-32

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

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

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

[NNS] Naor, J., Naor, M., Schaeffer, A.,
Fast Parallel Algorithms for Chordal Graphs
to appear in: 19^{th} STOC (1987)

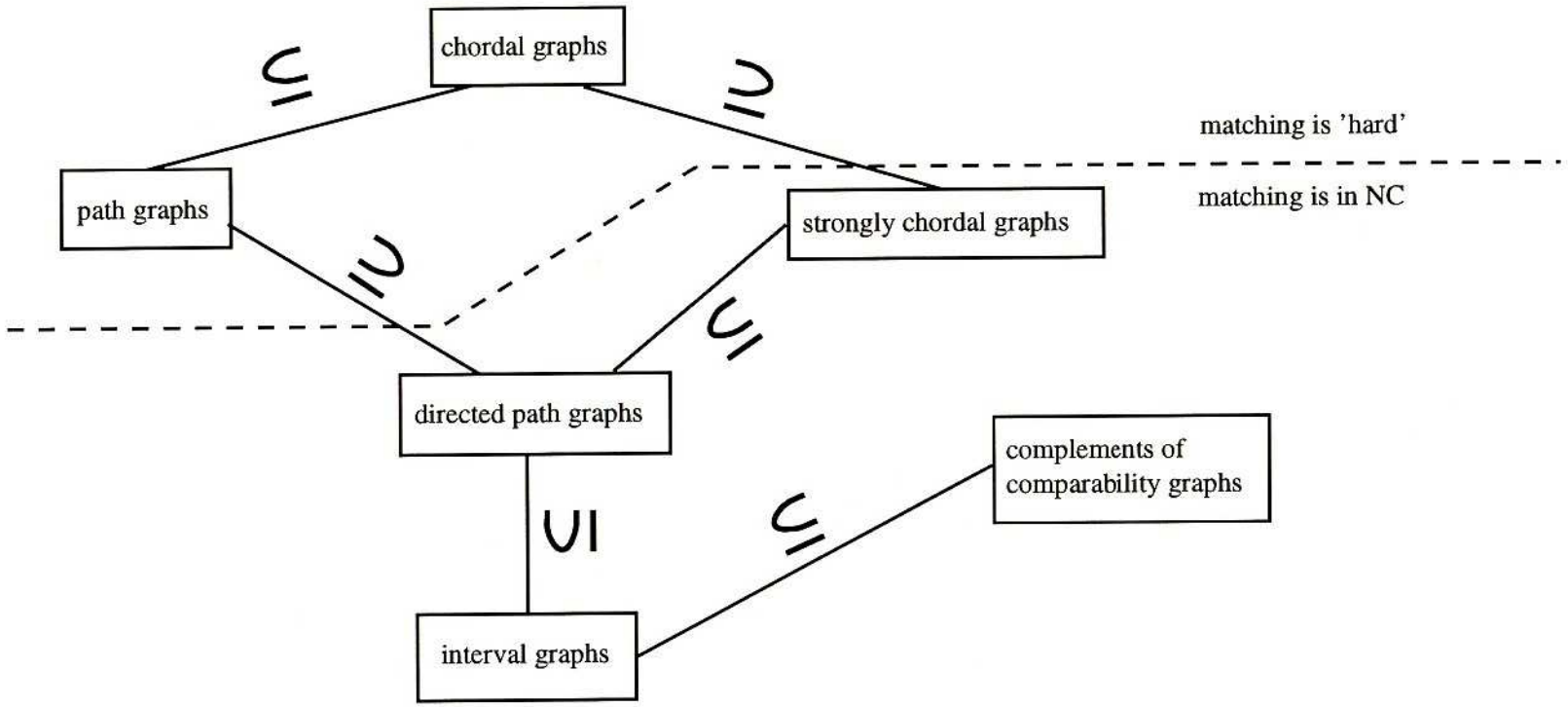
[Ru] Ruzzo, W.,
Tree Size Bounded Alternation
JCSS 21 (1980), pp. 218-135

[RV] Rabin, M.O., and Vazirani, V.,
Maximum Matchings in General Graphs through Randomization
Report No. TR 15−84, Center for Research in Computing Technology, Harvard University, Cambridge (1984)

[SV] Shiloach, Y., and Vishkin, K.,
An $O(\log n)$ Parallel Connectivity Algorithm
J. Algorithms **3** (1982), pp. 57-67

[TY 1] Tarjan, R., and Yannakakis, M.,
Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclity of Graphs, Test Acyclity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs
SIAM J. Comput. 13 (1984), pp. 566-579

[TY 2] Tarjan, R., and Yannakakis, M.,
Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclity of Graphs, Test Acyclity of Hypergraphs, and Selectively Reduce Hypergraphs; Addendum
SIAM J. Comput. 14 (1985), pp. 254-255


chordal graphs
path graphs
strongly chordal graphs
directed path graphs
interval graphs
complements of
comparability graphs
matching is 'hard'
matching is in NC