



The University of Sydney

**The Parallel Complexity of
Elimination Ordering Procedures**

Technical Report Number 456

February 1993

Elias Dahlhaus

ISBN 0 86758 664 8

**Basser Department of Computer Science
University of Sydney NSW 2006**

The Parallel Complexity of Elimination Ordering Procedures

Elias Dahlhaus
Basser Dept. of Computer Science
University of Sydney, NSW 2006, Australia

Abstract

We prove that lexicographic breadth-first search is P-complete and that a variant of the parallel perfect elimination procedure of P. Klein [24] is powerful enough to compute a semi-perfect elimination ordering in sense of [23] if certain induced subgraphs are forbidden. We present an efficient parallel breadth first search algorithm for all graphs which have no cycle of length greater four and no house as an induced subgraph. A side result is that a maximal clique can be computed in polylogarithmic time using a linear number of processors.

1 Introduction

Various authors considered elimination orderings to characterize certain graph classes [15, 37, 16, 6]. Such elimination orderings have several applications as sparse Gauss elimination [31], efficient graph coloring [6, 37, 7], etc.. An interesting graph class are the chordal graphs. That are the graphs with the property that any cycle of length greater than three has two nonconsecutive vertices which are adjacent. That are those graphs which have a *perfect elimination ordering*, i.e. the greater neighbors of any vertex form a complete set. Sequential linear algorithms to recognize chordal graphs and to compute a perfect elimination ordering are the lexical breadth-first search method of Rose, Tarjan, and Lueker [32] and the maximum cardinality search method of Tarjan and Yannakakis [36]. An efficient parallel algorithm is due to Klein [24]. It is well known that a minimum coloring of a chordal graph can be obtained efficiently by coloring the vertices in reversal order with the color of the least number which is not the color of a greater neighbor. There is a more generalized result in behind. Chvatal [6] found out that one can proceed in the same way if the ordering is a "perfect ordering" (not perfect elimination ordering), i.e. an ordering with the property that for no path (x_1, x_2, x_3, x_4) , such that nonconsecutive vertices are not adjacent, $x_2 < x_1$ and $x_3 < x_4$. Note that Chvatal presented perfect orderings in the reversed order [6]. Middendorf and Pfeiffer proved that the existence of a perfect ordering is an NP-complete problem [28]. Jamison and Olariu [23] introduced the notion of *semiperfect elimination orderings*, which lie between perfect elimination orderings and perfect orderings. We can state that an ordering is a perfect elimination ordering if any vertex is not the midpoint of an induced path of length two in the given graph restricted to those vertices which are greater than or equal the given vertex. Jamison and Olariu relaxed this condition that any vertex is not the midpoint of an induced path of length three (P_4) in the given graph restricted to those vertices greater or equal the given vertex. Jamison and Olariu [23] stated conditions when a semi-perfect elimination ordering can be computed efficiently by the known perfect elimination algorithms in [32] and [36]. Jamison and Olariu stated some properties that an ordering ever has if one applies the algorithm in [32] or the algorithm in [36] and proved that under certain conditions, an ordering satisfying these properties is a semi-perfect elimination ordering.

Here we deal with the problem to develop parallel algorithms which do essentially the same as lexical breadth-first search or maximum cardinality search. While a variant of P. Klein's parallel elimination algorithm does essentially the same as the maximum cardinality search, lexical breadth-first search is inherently sequential. An immediate consequence is that the computation of a doubly lexicalic ordering in sense of [27] is inherently sequential. Klein [24] developed also an efficient parallel breadth-first search tree algorithm for chordal graphs based

on the existence of a perfect elimination ordering. We develop an efficient parallel breadth-first search algorithm for chordal graphs which is not based on the knowledge of a perfect elimination ordering. We shall see that this algorithm can be extended to a larger graph class. It remains open to extend Klein's depth-first search algorithm for chordal graphs to larger graph classes.

In the second section, we introduce the notation necessary in the whole paper. In the third section, we describe different elimination processes and prove that lexical breadth-first search is P-complete. In the fourth section we discuss the power of a variant of P. Klein's parallel elimination procedure. Here we have to solve the problem to compute an inclusion maximal clique in parallel. As a side result, we found out that an inclusion maximal clique can be computed in polylogarithmic time with a linear processor bound in nearly the same manner as an inclusion maximal independent set [22]. In the fifth section, we discuss an efficient parallel breadth-first search algorithm which is not only applicable for chordal graphs but for graphs not containing the house or a cycle of length greater than four as an induced subgraph.

2 Notation

A graph $G = (V, E)$ consists of a *vertex set* V and an *edge set* E . Multiple edges and loops are not allowed. The edge joining x and y is denoted by xy .

We say that x is a *neighbor* of y iff $xy \in E$. The *neighborhood* of x is the set $\{y : xy \in E\}$ consisting of all neighbors of x and is denoted by $N(x)$. The neighborhood of a set of vertices V' is the set $N(V') = \{y | \exists x \in V', xy \in E\}$ of all neighbors of some vertex in V' .

A *subgraph* of (V, E) is a graph (V', E') such that $V' \subset V$, $E' \subset E$. An *induced subgraph* is an edge-preserving subgraph, i.e. (V', E') is an induced subgraph of (V, E) iff $V' \subset V$ and $E' = \{xy \in E : x, y \in V'\}$. The *clique number* of a graph G is denoted by $\omega(G)$. The *chromatic number* of a graph G is denoted by $\chi(G)$. A graph is *perfect* if, for each induced subgraph, the chromatic number and the clique number coincide. A trivial heuristics to color a graph is to color the vertices in reversed order with respect to an enumeration $(v_i)_{i=1}^n$ of the vertex set V with the available color of least index (*greedy coloring*). This heuristics gives a coloring in the bounds of the chromatic number for each induced subgraph if and only if the ordering behind the enumeration is the reversal of a *perfect ordering* [6]:

If (v_i, v_j, v_k, v_l) forms an induced path then $i < j$ or $l < k$.

It is NP-complete to check, for any graph, whether it has a perfect ordering or not [28]. But in special graph classes necessarily a perfect ordering exists. Moreover in these special graph classes, a perfect ordering can be determined efficiently.

Special perfect ordering are the *perfect elimination orderings*:

If $x < y, x < z, xy, xz \in E$, then $yz \in E$.

Lemma 1 [16] *A graph has a perfect elimination ordering iff it is chordal, i.e. it has no induced cycle of length greater than 3.*

A generalization of perfect elimination orderings are semi-perfect elimination orderings [23]. An ordering $<$ on the vertex set V of a graph $G = (V, E)$ is called *semi-perfect* if each vertex $v \in V$ is not the midpoint of an induced path of length three (P_4) in G restricted to $\{w | v \leq w\}$.

M. Farber [15] introduced a subclass of chordal graphs, the *strongly chordal graphs*. This class is interesting because graph problems as Minimum Steiner Tree and Minimum Dominating Set can be solved efficiently:

A graph $G = (V, E)$ is strongly chordal iff there is a perfect elimination ordering $<$ with the following additional property:

- for $x_1y_2, x_2y_1, x_1x_2 \in E$, such that $x_1 < y_1$ and $x_2 < y_2$, we have $y_1y_2 \in E$.

We call such an ordering a *strongly perfect elimination ordering*.

The parallel computation model is the parallel random access machine (PRAM). In most cases we use the Exclusive Read Exclusive Write PRAM (EREW-PRAM), i.e. only one processor is allowed to read from and to write into the same memory cell at the same time. Sometimes we also use the Concurrent Read Concurrent Write PRAM model (CRCW-PRAM), i.e. at the same time arbitrary many processors are allowed to read from the same memory cell and arbitrary many processors are allowed to apply to write into the same memory cell. Usually the processor with the smallest number applying to write into a certain memory cell writes into the memory cell.

3 Elimination Procedures

In this section we recall known elimination procedures and their power.

Jamison and Olariu considered elimination orderings with the following properties [23]:

P1 If $a < b, b < c, ac \in E, bc \notin E$ then there is a vertex b' such that $bb' \in E, ab' \notin E, c < b'$.

P2 If $a < b, b < c, ac \in E, bc \notin E$ then there is a vertex b' such that $bb' \in E, ab' \notin E, b < b'$.

Note that property P1 implies property P2.

Tarjan and Yannakakis [36] proved the following.

Lemma 2 *G is chordal iff any ordering satisfying property P2 is a perfect elimination ordering.*

Note that that the maximum cardinality search procedure of Tarjan and Yannakakis [36] ever computes an ordering which satisfied property P2 and the lexicographical breadth-first search procedure of Rose, Tarjan, and Lueker ever computes an ordering satisfying property P1.

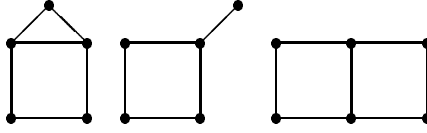
Jamison and Olariu [23] stated necessary and sufficient conditions that an ordering satisfying property P1 or P2 is semi-perfect.

We consider the following graphs:

We call the first graph the *house*, the second graph the *guitar* and the third graph the *domino*.

Lemma 3 [23]

1. For each induced subgraph H of G , any ordering on the vertices of H satisfying P1 is a semi-perfect elimination ordering iff G does not contain a if it does not contain a cycle of length > 4 or the house or the domino as an induced subgraph.



2. For each induced subgraph H of G , any ordering on the vertices of H satisfying P2 is a semi-perfect elimination ordering iff G does not contain a cycle of length > 4 or the house or the guitar as an induced subgraph.

Lubiv [27] extended even the conditions of P1:

P0 If $a < b, ac \in E, bc \notin E$ then there is a vertex b' such that $bb' \in E, ab' \notin E, c < b'$.

In the case of bipartite graphs an ordering satisfying P0 is also called a *doubly lexicalic ordering*. It can be proved that, in any strongly chordal graph, an ordering satisfying P0 is a strongly perfect elimination ordering. Efficient sequential algorithms to compute a doubly lexicalic ordering are due to Lubiv [27], Paige and Tarjan [30], and Spinrad [34].

We shall see that even the computation of an ordering satisfying the property P1 is inherently sequential.

Theorem 1 *The computation of an ordering satisfying P1 is P-complete.*

Proof: We prove that an NC-algorithm for the computation of an ordering satisfying the property P1 induces an NC-algorithm for the following problem which is P-complete [10]:

First Maximal Clique

Input: A graph $G = (V, E)$ and an ordering $<$ on the set V of the vertices of G .

Output: The clique C which is computed as follows:

1. The smallest element v_0 is in C ,
2. put stepwise the smallest element $v \notin C$ into C which is adjacent to all vertices in C until no vertex not in C is adjacent to all vertices in C .

Suppose $G = (V, E)$ with $V = \{v_1, \dots, v_n\}$ and the ordering $<$ on V with $v_i < v_j$ iff $i < j$ are given.

First we construct a (bipartite) graph $G' = (V', E')$ with a distinguished vertex M such that any ordering with M as maximum which satisfies P1 induces the first maximum clique in G .

Set $V' = \{M\} \cup \bigcup_{i=0}^n V_i$ with $V_0 = \{0\} \times V$, and for $i > 0$, $V_i = \{i\} \times \{v_j | j \geq i\}$.

The edge set E' is defined as follows. $E' = \{Mv | v \in V_0\} \cup \{(0, v_i)(1, v_j) | i \geq j\} \cup \{(i, v_j)(i+1, v_j) | j \geq i+1\} \cup \{(i, v_i)(i+1, v_j) | v_i v_j \in E \text{ and } i \leq j\}$.

It is easily seen that G' can be computed from G by an EREW-PRAM with $O(n^3)$ processors in logarithmic time.

Suppose the first maximal clique C of G is $\{v_{i_1}, \dots, v_{i_k}\}$ and $1 = i_1 < \dots < i_k$. Let $<'$ be a lexicographical breadth first search ordering on G' with M as its maximum. Then by induction on i , one can show:

Lemma 4 *Suppose $i_j < i \leq i_{j+1}$. Then the following statements are satisfied.*

1. $(i, v_{i_{j+1}})$ is the largest vertex in V_i with respect to $<'$,
2. Let $v_l \geq v_i$ be a vertex which is adjacent to all vertices v_{i_1}, \dots, v_{i_j} and $v_{l'} \geq v_i$ be a vertex which is not adjacent to all vertices v_{i_1}, \dots, v_{i_j} . Then $(i, v_{l'}) <' (i, v_l)$.
3. If $v_l \geq v_i$ and $v_{l'} \geq v_i$ are both adjacent to all vertices v_{i_1}, \dots, v_{i_j} then $(i, v_l) <' (i, v_{l'})$ iff $l' < l$.

Proof of Lemma

First note that $i < j$ and $(1, v_i) >' (1, v_j)$ are equivalent statements, because $i < j$ implies that the neighborhood of $(1, v_j)$ in V_0 is a proper subset of the neighborhood of $(1, v_i)$ in V_0 . Therefore $(1, v_1) = (1, v_{j_1})$ is the $<'$ -largest vertex in V_1 . The second statement is trivially true for $i = 1$ because no i_j is smaller than 1. Therefore the lemma has been proved for $i = 1$.

Suppose the lemma is true for $i = 1, \dots, i' - 1$. To prove the lemma for $i = i'$, we have to consider two cases.

Case 1: $i' - 1 = i_j$, for some j .

By induction hypothesis, $(i' - 1, v_{i_j}) = (i' - 1, v_{i' - 1})$ is the largest vertex in $V_{i' - 1}$. Therefore the vertices (i', v_j) with $v_{i' - 1}v_j \in E$ are $<'$ -larger than those vertices (i', v_j) such that v_j is not in the neighborhood of $v_{i' - 1}$, because the neighbors of $(i' - 1, v_{i' - 1})$ in $V_{i'}$ are those (i', v_j) with $v_{i' - 1}v_j \in E$. Note that the only remaining neighbor of (i', v_j) in $V_{i' - 1}$ is $(i' - 1, v_j)$. Therefore for all (i', v_j) and (i', v_k) , such that v_j and v_k are neighbors of $v_{i' - 1}$, $(i', v_j) <' (i', v_k)$ iff $(i' - 1, v_j) <' (i', v_k)$. Therefore, since the second and the third statement is true for $i = i' - 1$, they are even true for $i = i'$. Since $v_{i_{j+1}}$ is the smallest vertex adjacent to all vertices v_{i_1}, \dots, v_{i_j} , also the first statement is true for $i = i'$.

Case 2: $i_j < i' - 1 < i_{j+1}$.

Note that for all v_j with $j \geq i'$, $(i' - 1, v_j)$ is a neighbor of (i', v_j) and $(i' - 1, v_{i' - 1})$ is the only other possible neighbor of (i', v_j) . Moreover $v_{i' - 1}$ is not adjacent to all vertices v_{i_1}, \dots, v_{i_j} . Since the second statement is true for $i = i' - 1$, $(i' - 1, v_{i' - 1}) <' (i' - 1, v_k)$ such that v_k is adjacent to all v_{i_1}, \dots, v_{i_j} . Since the neighbors of (i', v_k) in $V_{i' - 1}$ are $(i' - 1, v_k)$ and possibly $(i' - 1, v_{i' - 1})$, for all v_k, v_l with $(i' - 1, v_{i' - 1}) <' (i' - 1, v_k)$, $(i' - 1, v_l)$, $(i', v_k) <' (i', v_l)$ iff $(i' - 1, v_k) <' (i' - 1, v_l)$. Therefore the second and the third statement are true for $i = i'$. The first statement follows from the third statement by the definition of $v_{i_{j+1}}$.

□ (Lemma)

Therefore C consists of those v_i with $i \leq i_k$ such that (i, v_i) is maximal in V_i with respect to $<'$. Only we do not know i_k .

To compute C , we compute the set $C' = \{v_i : (i, v_i) \text{ is maximal with respect to } <' \text{ in } V_i\}$ and determine the first index i' in C' such that there is an index $j < i'$ in C' such that $v_jv_i \notin E$. Apparently $C = \{v_j \in C' | j < i'\}$.

For the general case, we consider two copies of G' and identify in both copies the vertex M . The maximum must be in one of the copies. Any ordering has its maximum in one of the copies of G' . Then M must be the maximum of the other copy and there the behavior of an ordering satisfying P1 must be the same as if we assume that M is the maximum.

□

4 P. Klein's Perfect Elimination Procedure and its Application to Compute Semi-Perfect Elimination Orderings

Here present a variant of Klein's parallel perfect elimination procedure which computes, for any graph, an ordering satisfying P2 [24].

The basic idea of P. Klein's algorithm is that we begin with an initialization procedure NONE which computes levels L_1, \dots, L_k of size at most a fixed ratio of the number of vertices. Successively the levels L_i are refined independently into sublevels that have again a size of at most a fixed ratio of the number of vertices in L_i . We stop this procedure if all levels have the cardinality one.

To compute the initial level partition, P. Klein [24] made use of the fact that the intersection of the neighborhoods of two nonadjacent vertices of a chordal graph induces a complete graph. This is not true in general. Therefore we have to develop an elimination procedure which works for any graph.

We proceed as follows:

1. We compute a maximal clique C on those vertices which have a larger degree than $2/3|V|$. Suppose $C = \{x_1, \dots, x_k\}$
2. For $i = 1, \dots, k$, we set $L_i := \{v \in V \setminus C \mid \text{for all } j < i, x_j v \in E \text{ and } x_i v \notin E\}$ for each $i = 1, \dots, k$. Clearly each L_i has a cardinality of at most $1/3|V|$ and $M = \{v \in V \setminus C \mid \forall x \in C, vx \in E\}$.
3. We compute the connected components M_1, \dots, M_p of G restricted to M and let M_p be that connected component with the largest cardinality. Let $L_{k+i} = M_i$, for $i = 1, \dots, p-1$
4. To refine M_p , we compute a spanning tree T for M_p . We select a vertex r of M_p as root and compute the enumeration (u_1, \dots, u_q) of a postordering with respect to T (see [35]). $v \in M_p \setminus \{r\}$ is set into $L_{k+p+i-1}$ if i is the maximum index such that $vu_i \in E$ and r is set into L_{k+p+q} .
5. We set $L_{k+p+q+i} = \{x_{k-i+1}\}$.

It is easily checked that the levels L_1, \dots, L_k have a cardinality of at most $1/3|V|$, the levels $L_{k+1}, \dots, L_{k+p-1}$ have a cardinality of at most $1/2|V|$, and the levels $L_{k+p}, \dots, L_{k+p+q-1}$ have a cardinality of at most $2/3|V|$ (note that all common neighbors of C have a degree of at most $2/3|V|$).

The following fact can be interpreted in such a way that the ordered partition (L_1, \dots, L_{2k+p+q}) is an approximation of an ordering satisfying P2.

Fact 1 *Suppose $a \in L_i, b \in L_j, c \in L_l, i < j < l$, and $ac \in E, b_c \notin E$. Then there is a vertex $b' \in L_{l'}$ with $ab' \notin E, bb' \in E$, and $l' > j$.*

To refine the level with maximal index, we only have to apply above procedure.

To refine any other level, say L_i into sublevels $L_{i,1}, \dots, L_{i,r}$, such that the requirements of 1 are satisfied, we can proceed as in [24].

Let $G_i = (V_i, E_i)$ with $V_i = L_i \cup \{x \mid \exists y \in L_i, xy \in E \text{ and } x \in L_j \text{ for some } j > i\}$ and $E_i = \{xy \in E \mid x, y \in V_i, x \in L_i \text{ or } y \in L_i\}$. Let S_i be the set of those vertices in V_i with at

most $2/3|L_i|$ neighbors in L_i (sparse vertices) and D_i be the set of those vertices in V_i which have more than $2/3|L_i|$ neighbors in L_i (dense vertices).

Let B_i be the union of all connected components of S_i which contain at least some vertex in some L_j with $j > i$. Let L_i^2 be the set of all vertices in L_i which are in B_i or adjacent to some vertex in B_i and L_i^1 be the set of vertices in L_i which are not adjacent to some vertex in B_i .

If L_i^2 has a size greater than $2/3|L_i|$ then we refine the level L_i^2 as follows:

1. We compute a spanning forest F_i for B_i consisting of the trees T_i^1, \dots, T_i^r .
2. For each such tree T_i^j , we choose root R_i^j which is in some $L_{i'}$ with $i' > i$ (note that such a vertex R_i ever exists).
3. We compute the concatenation (u_1, \dots, u_s) of postorders of the trees T_i^j .
4. For any $v \in L_i^2$, we put v into $L_{i,j}$ if u_j is the neighbor of v in B_i with the largest index j .
5. The new levels are $L_i^1, L_{i,j}, j = 1, \dots, s$. L_i^1 is that level with the smallest index.

Now we suppose that L_i^1 is of size greater than $2/3|L_i|$. Note that all vertices in L_i^2 or in some L_j with $j > i$ which have neighbors in L_i^1 have more than $2/3|L_i|$ neighbors in L_i and are therefore in D_i . To get a refinement of L_i^1 , we first fix some enumeration (v_1, \dots, v_t) of all vertices in D_i which are in L_i^2 or not in L_i . We set $L_{i,j} = \{v \in L_i^1 | vv_j \notin E \text{ and for all } j' < j, vv_{j'} \in E\}$. All these levels $L_{i,j}$ are of cardinality at most $1/3|L_i|$.

To refine the set M_i of all vertices in L_i^1 which are adjacent to all vertices v_j , we apply the initial procedure to create a first refinement to G restricted to M_i .

Note that fact 1 remains true and all levels have a cardinality of at most $2/3|L_i|$.

Before we can make a complexity statement, we have to consider the complexity to compute an inclusion maximal clique.

Theorem 2 *A maximal clique can be computed by an EREW-PRAM in $O(\log^3 n)$ time using $O(n + m)$ processors.*

Proof: We proceed in a similar way as in the maximal independent set algorithm of Goldberg and Spencer [22].

To make the paper more self contained, we repeat the algorithm of Goldberg and Spencer.

The main subprocedure is *FINDSET* which computes an independent set I such that $|I \cup N(I)| \geq (1/2 - o(1))(n + m)$.

We apply *FINDSET* to the original graph, remove all vertices of $I \cup N(I)$ and apply again *FINDSET* to the rest. This is done $O(\log n)$ times.

The procedure *FINDSET* works as follows:

At each step, the remaining vertex set is partitioned into color classes, which all are independent sets, and uncolored vertices.

Initially, each vertex forms a color class.

Denote the degree of a vertex x by $deg(x)$. The *weight* of a vertex set X is defined as $\sigma(X) = \sum_{x \in X} (1 + deg(x))$.

If there is a color class C^* such that $\sigma(N(C^*)) \geq (n + m)/\log n$ then we delete all vertices in $C^* \cup N(C^*)$ and add C^* to the independent set I (Phase 1).

If we do not find such a C^* then we enter phase 2 defined as follows:

We suppose the color classes are C_0, \dots, C_{k-1} . We partition the pairs $(i, j), i = 0, \dots, k-1$ as follows:

If k is even then

$$\text{index}(i, j) = (i + j) \bmod (k - 1), \text{ for } i, j \neq k - 1$$

and

$$\text{index}(k - 1, j) = \text{index}(j, k - 1) = 2j.$$

If k is odd then

$$\text{index}(i, j) = (i + j) \bmod k.$$

Fact [22]: If $\text{index}(i, j) = \text{index}(i', j')$ then the unordered pairs $\{i, j\}$ and $\{i', j'\}$ are equal or disjoint.

We find a *suitable* index l and mix all pairs C_i, C_j with $\text{index}(i, j) = l$ as follows:

We compare the weights of $C_i \cap N(C_j)$ and $C_j \cap N(C_i)$. Suppose $C_i \cap N(C_j)$ is of smaller weight. Then we uncolor $C_i \cap N(C_j)$ and make $(C_i \cap C_j) \setminus (C_i \cap N(C_j))$ a new color class.

An index l is considered as suitable if the weight of the set of uncolored vertices is minimal. The whole procedure *FINDSET* stops if there is only one color class.

To turn the algorithm of Goldberg and Spencer into an algorithm to compute a maximal clique, we proceed as follows. Color classes are pairwise disjoint complete sets. To check some weight constraints, we have to compute the degree $\text{deg}'(x)$ of any vertex in the complement of G . Clearly $\text{deg}'(x) = n - \text{deg}(x) - 1$. We define the weight of a set C as $\sigma'(C) = \sum_{x \in C} (1 + \text{deg}'(x))$.

Instead of computing the neighborhood of a color class C_i , we have to compute the set J_i of vertices that are adjacent to all vertices in C_i . This can be done by computing, for each v and each C_i with an edge $vy \in E$ with $y \in C_i$, the number of neighbors of v in C_i . If this number coincides with $|C_i|$ then v belongs to J_i .

For all i simultaneously, J_i can be computed in $O(\log n)$ time using $O(n + m)$ processors.

Let $N'(C)$ be the neighborhood of C in the complement of G .

To check whether phase 1 can be applied to C_i , we have to compute the weight $\sigma'(N'(C_i))$ of $N'(C_i)$. Note that $N'(C_i) = V \setminus (C_i \cup J_i)$. Therefore $\sigma'(N'(C_i)) = \sigma'(V) - \sigma'(C_i) - \sigma'(J_i)$. Note that the number of pairs (i, x) with $x \in J_i$ is bounded by m . Therefore for all i simultaneously, $\sigma'(J_i)$ can be computed in $O(\log n)$ time using $O(n + m/\log n)$ processors. The values $\sigma'(C_i)$ can be computed, simultaneously for all i , in $O(\log n)$ time with $O(n)$ processors.

The step to delete C_i and all vertices in $N'(C_i)$ is the deletion of all vertices which are not in J_i . This can be done in $O(\log n)$ time using $O(n)$ processors by checking whether x is in J_i .

Instead of finding a partition of the color classes that minimizes the weight of the uncolored vertices, we find a partition which maximizes the weight of the set of vertices which remain colored.

For each i , we compute the weight $w_{(i,j)}$ of the nonempty sets $J_i \cap C_j$ in $O(\log n)$ time with $O(n + m)$ processors. The set of vertices of $C_i \cup C_j$ which remain colored is $w'_{(i,j)} = \sigma'(C_i) + w_{(i,j)}$. We have the choice to uncolor vertices in C_i or to uncolor vertices in C_j . To

minimize the weight of uncolored vertices or to maximize the weight of colored vertices, we consider the maximum $w''_{i,j} = \max(w'_{(i,j)}, w'_{(j,i)})$. Now we have no direct access to the pairs (i, j) such that $C_j \cap J_i = \emptyset$ and $C_i \cap J_j = \emptyset$ if we want to restrict the number of processors by $O(n + m)$. For such pairs, the C_i or C_j with the maximum weight remains colored. To find the sum of weights of such color classes, we first sort the color classes C_i with respect to their weights (in $O(\log n)$ time with $O(n)$ processors [8]). We consider now the set S_l of all i such that for that j with $index(i, j) = l$, $i > j$. If k is odd then that are those i with $l/2 \leq i < l$ and those i with $(k + l)/2 \leq i < k$, i.e. of the intervals $[l/2, l)$ and $[(k + l)/2, k)$. If k is even then S_l consists of those i with $i = k - 1$ or $l/2 < i < l$ or $(k + l - 1)/2 < i < k - 1$, i.e. of the intervals $(l/2, l)$ and $((k + l - 1)/2, k)$.

For each pair C_i, C_j , C_i is a maximum weight class modulo $index(i, j)$ if the weight of C_i is at least as large as the weight of C_j .

To compute the weight of the set of vertices which remain colored if we choose those pairs with $index(i, j) = l$, we compute the sum of weights of the maximum weight classes modulo l and for each pair i, j with $(C_i \cap J_j) \cup (C_j \cap J_i) \neq \emptyset$ and $index(i, j) = l$, we add the difference $d''(i, l)$ of $w''_{i,j}$ and the maximum weight of the color classes C_i and C_j . Note that the number of pairs i, j with $(C_i \cap J_j) \cup (C_j \cap J_i) \neq \emptyset$ is bounded by m . Therefore it is possible, to compute, for all such pairs i, j simultaneously, the maximum weight $m(i, j)$ of C_i and of C_j in constant time with $O(m)$ processors. We get the same time and processor bound to compute $d(i, l) = d(i, index(i, j)) = w''(i, j) - m(i, j)$. It remains to compute the sum of weights of the maximum weight classes modulo l , for all l simultaneously. For this purpose, we compute the prefix sums $Sum_i = \sum_{j < i} \sigma'(C_j)$, for all $i = 0, \dots, k - 1$ simultaneously, in $O(\log n)$ time with $O(n/\log n)$ processors (see for example [20]). Using the fact that S_l is ever a union of two intervals, we get the sum of the weights of the maximum weight classes modulo l in constant time with $O(n)$ processors, for all l simultaneously.

Therefore we can execute phases 1 and 2 in $O(\log n)$ time with $O(n + m)$ processors.

Now we only have to follow the arguments of [22] to get a time bound of $O(\log^3 n)$ to compute an inclusion maximal clique.

□

Putting above elimination algorithm and last theorem together, we get the following.

Theorem 3 *The enumeration of an ordering satisfying property P1 can be computed by an EREW-PRAM in $O(\log^4 n)$ time using $O(n + m)$ processors.*

An immediate consequence is the following.

Corollary 1 *For any graph which does not contain a cycle of length greater than four, a house, or a guitar as an induced subgraph, a semi-perfect elimination ordering can be computed by an EREW-PRAM in $O(\log^4 n)$ time using $O(n + m)$ processors.*

5 An Efficient Parallel Breadth-First Search Algorithm for Graphs without Houses and without Cycles of Length Greater than Four

P. Klein's parallel breadth first search algorithm for chordal graphs computes a breadth first search tree by setting, for each vertex the largest neighbor with respect to a perfect elimination

ordering as its parent. We shall see that the following algorithm computes a breadth first search tree of a chordal graph without the use of a perfect elimination ordering.

Algorithm BFC:

1. Compute a spanning tree T for the graph $G = (V, E)$.
2. Compute the enumeration (u_1, \dots, u_n) of a postorder on T .
3. $L_i = \{v \mid i \text{ is the maximal index such that } vu_i \in E\}$ and $L_{n+1} = \{u_n\}$.
4. Let $p(v)$ be the maximum i such that v has a neighbor in L_i .
5. The parent $P(v)$ is a vertex $w \in L_{p(v)}$ with a maximum number of neighbors in $\bigcup_{j>p(v)} L_j$.

The following extended result is true.

Lemma 5 *The tree T_P constructed in algorithm BFC is a breadth-first search tree if $G = (V, E)$ has no house and no cycle of length > 4 as an induced subgraph.*

Proof: If $x \in L_i$, $y \in L_j$, and $x < j$ then we write also $x < y$. We say $x \leq y$ iff $x \in L_i$, $y \in L_j$, and $i \leq j$.

Clearly $v < P(v)$. By construction, if $xy \in E$ and $y \neq P(x)$ then y cannot be an ancestor of x .

Let r be the root of T , and therefore also the root the tree T_P with parent function P .

Let M_i be the set of vertices with distance i from r in T_P . To prove that P is the parent function of a breadth-first search tree, we have to prove that if $xy \in E$, $x \in M_i$, and $y \in M_j$ then $|i - j| \leq 1$.

We prove this indirectly. Assume $xy \in E$, $x \in M_i$, $y \in M_j$, and $j - i \geq 2$. Moreover we assume that j is a maximum index, such that there is an i with $j - i \geq 2$ and there are $x \in M_i$ and $y \in M_j$ with $xy \in E$ (j satisfies the *maximality condition*).

Claim: $P(x)P(y) \in E$.

Proof of Claim: Since $j - i \geq 2$, $y < P(x)$. We consider two cases:

1. $P(y) \leq P(x)$: Note that $x < y < P(y) \leq P(x)$. That means there is an l such that $P(x), P(y) \in \bigcup_{i \geq l} L_i$ and $x, y \notin \bigcup_{i \geq l} L_i$. Therefore x and y are not neighbors of the set $S_l = \{u_l, \dots, u_n\}$ and $P(x)$ and $P(y)$ are neighbors of S_l . Note that S_l induces a subtree of T and therefore a connected subgraph of G . Therefore there is a path p from $P(x)$ to $P(y)$ such that all inner vertices are in S_l . $P(x), x, y, P(y)$ together with p forms a cycle C of length greater than four. We can assume that p has no chord. Therefore the only chord of C is $P(x)P(y)$ and such a chord must exist, because we assume that G has no cycle of length greater than four as an induced subgraph.
2. $P(x) < P(y)$: Note that $x < y < P(x) < P(y), P(P(x))$. Then we find an l such that $x, y \notin \bigcup_{i \geq l} L_i$ and $P(x) \in L_l$. Therefore x and y are not neighbors of u_l and $P(x)$ is a neighbor of u_l . Note that $P(x) < u_l$. Therefore we find a k such that $x, y, P(x) \notin \bigcup_{i \geq k} L_i$ and $P(y), u_l \in \bigcup_{i \geq k} L_i$. Therefore $P(y)$ and u_l are in the neighborhood of $S_k = \{u_k, \dots, u_n\}$ but $x, y, P(x)$ are not in the neighborhood of S_k . Since S_k is connected, there is a (chordless) path p from $P(y)$ to u_l with inner vertices in S_k . $P(y), y, x, P(x), u_l$ together with p forms a cycle C .

Assume $P(y)P(x) \notin E$. Since even a chord yu_l does not exist, we are forced to choose chords $P(y)u_l$ or $yP(x)$. But if we choose only the chord $P(y)u_l$ then we still have an induced cycle $P(y), u_l, P(x), x, y$ of length greater five, and if we have only the chord $yP(x)$ then $P(y), y, P(x), u_l$ together with p forms an induced cycle of length at least five. Therefore both possible chords of C must exist as edges. But then $\{x, y, P(y), u_l, P(x)\}$ induces a house. This is a contradiction.

□ (Claim)

Note that $P(x) \in M_{i+1}$ and $P(y) \in M_{j+1}$. The fact that $P(x)P(y) \in E$ leads to a contradiction to the maximality condition of for j .

□(Lemma)

Note that all steps of the algorithm BFC with the exception of the first step can be executed by an EREW-PRAM in $O(\log n)$ time using $O(n + m)/\log n$ processors. The first step can be done by a CRCW-PRAM in $O(\log n)$ time with $O(n + m)$ processors [33]. Therefore we get the following result.

Theorem 4 *Suppose G is a graph not containing the house or a cycle of length greater than 4 as an induced subgraph. Then a breadth-first search tree can be computed in the same time and processor bound as a spanning tree, i.e. in $O(\log^2 n)$ time with $O(n + m)$ processors on a EREW-PRAM or with the same processor bound in $O(\log n)$ time on a CRCW-PRAM.*

The depth-first search tree algorithm of P. Klein cannot be improved in the same way. It remains an open problem to compute a depth first search tree efficiently in parallel for certain graph classes beyond chordal graphs or planar graphs.

6 Conclusions

We did not give a characterization of graphs having a semiperfect elimination ordering. It is also not clear how to check efficiently (with a linear processor number in polylogarithmic time) that the ordering we just computed is a semiperfect elimination ordering. It might be an interesting problem to check efficiently (in parallel) that a given ordering on the vertices of a graph is a semiperfect elimination ordering.

References

- [1] A. Aho, J. Hopcroft, J. Ullman, *The Design and Analysis of Computer Algorithms*, Addison-Wesley, Reading Mass., 1975.
- [2] R. Anstee, M. Farber, *Characterization of Totally Balanced Matrices*, Journal of Algorithms 5 (1984), S. 215-230.
- [3] C. Beeri, R. Fagin, D. Meyer, M. Yannakakis, *On the Desirability of Acyclic Database Schemes*, Journal of the ACM 30 (1983), S. 479-513.
- [4] A. Brower, P. Duchet, A. Schrijver, *Graphs, whose Neighborhood has no Special Cycles*, Discrete Mathematics 47(1983), S. 177-182.

- [5] A. Brower, A. Kolen, *A Super-Balanced Hypergraph Has a Nest-Point*, Report ZW 146/80, Mathematisch Centrum, Amsterdam.
- [6] V. Chvatal, *Perfectly Ordered Graphs*, Topics on Perfect Graphs, C. Berge, V. Chvatal eds., North Holland, Amsterdam(1984), pp. 63-65.
- [7] V. Chvatal, C. Hoang, N. Mahadev, D. de Werra, *Four Classes of Perfectly orderable Graphs*, Journal of Graphs Theory 11 (1987), pp. 481-495.
- [8] R. Cole, *Parallel Merge Sort*, 27. IEEE-FOCS (1986), pp. 511-516.
- [9] P. Bunemann, *A Characterization of Rigid Circuit Graphs*, Discrete Mathematics 9 (1974), S. 205-212.
- [10] S. Cook, *A Taxonomy of Problems with Fast Parallel Algorithms*, Information and Control 64 (1985), S. 2-22.
- [11] E. Dahlhaus, *Chordale Graphen im besonderen Hinblick auf parallele Algorithmen*, Habilitation thesis, 1991, University of Bonn.
- [12] E. Dahlhaus, M. Karpinski, *Fast Parallel Computation of Perfect and Strongly Perfect Elimination Schemes*, Forschungsbericht Nr. 8513-CS, Universit"at Bonn (1987).
- [13] A. D'Atri, M. Moscarini, *On Hypergraph Acyclicity and Graph Chordality*, Information Processing Letters 29(1988), pp. 271-274.
- [14] R. Fagin, *Degrees of Acyclicity and Relational Database Schemes*, Journal of the ACM 30 (1983), S. 514-550.
- [15] M. Farber, *Characterizations of Strongly Chordal Graphs*, Discrete Mathematics 43 (1983), S. 173-189.
- [16] D. Fulkerson, O. Gross, *Incidence Matrices and Interval Graphs*, Pacific Journal of Mathematics 15 (1965), pp.835-855.
- [17] M. Farber, R. Jamison, *Convexity in Graphs and Hypergraphs*, SIAM Journal on Algebraic and Discrete Methods 7 (1986), S. 433-444.
- [18] S. Fortune, J. Wyllie, *Parallelism in Random Access Machines*, 10. ACM-STOC (1978), S. 114-118.
- [19] F. Gavril, *The Intersection Graphs of Subtrees in Trees Are Exactly the Chordal Graphs*, Journal of Combinatorial Theory Series B, vol. 16(1974), S. 47-56.
- [20] A. Gibbons, W. Rytter, *Efficient Parallel Algorithms*, Cambridge University Press, Cambridge, 1989.
- [21] J. Gilbert, H. Hafsteinsson, *Parallel Solution of Sparse Linear Systems*, SWAT 88 (1988), LNCS 318, S. 145-153.
- [22] M. Goldberg, T. Spencer, *Constructing a Maximal Independent Set in Parallel*, SIAM Journal on Discrete Mathematics 2 (1989), pp. 322-328.

- [23] B. Jamison, S. Olariu, *On the Semi-Perfect Elimination*, Advances in Applied Mathematics 9 (1988), pp. 364-376.
- [24] P. Klein, *Efficient Parallel Algorithms for Chordal Graphs*, 29. IEEE-FOCS (1988), S. 150-161.
- [25] P. Klein, J. Reif, *An Efficient Parallel Algorithm fo Planarity*, Journal of Computer and Systems Sciences 37 (1988), S. 190-246.
- [26] J.C.T. Lee, Chin-Wen Ho, *Efficient Parallel Algorithms for Finding Maximal Cliques, Clique Trees, and Minimum Coloring on Chordal Graphs*, Information Processing Letters 28 (1988), S. 301-309.
- [27] A. Lubiw, *Doubly Lexical Orderings of Matrices*, SIAM Journal on Computing 16 (1987), S. 854-879
- [28] M. Middendorf, F. Pfeiffer, *On the Complexity of Recognizing Perfectly Orderable Graphs*, Discrete Mathematics 80 (1990), pp. 327-333.
- [29] J. Naor, M. Naor, A. Schäffer, *Fast Parallel Algorithms for Chordal Graphs*, SIAM Journal of Computing 18 (1989), S. 327-349.
- [30] R. Paige, R. Tarjan, *Three Partition Refinement Algorithms*, SIAM Journal on Computing 16 (1987), S. 973-989.
- [31] D. Rose, *Triangulated Graphs and the Elimination Process*, Journal of Mathematical Analysis and Applications 32 (1970), S. 597-609.
- [32] D. Rose, R. Tarjan, G. Lueker, *Algorithmic Aspects on Vertex Elimination on Graphs*, SIAM Journal on Computing 5 (1976), S. 266-283.
- [33] Y. Shiloach, U. Vishkin, *An $O(\log n)$ Parallel Connectivity Algorithm*, Journal of Algorithms 3 (1982), S. 57-67.
- [34] J. Spinrad, *Doubly Lexical Ordering for Dense 0-1 Matrices*, to appear.
- [35] R. Tarjan, U. Vishkin, *Finding Biconnected Components in Logarithmic Parallel Time*, SIAM-Journal on Computing 14 (1984), S. 862-874.
- [36] R. Tarjan, M. Yannakakis, *Simple Linear Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Acyclic Hypergraphs*, SIAM Journal on Computing 13 (1984), S. 566-579.
Addendum: SIAM Journal on Computing 14 (1985), S. 254-255.
- [37] D. Welsh, M. Powell, *An Upper Bound on the Chromatic Number of a Graph and its Application to Timetabling Problems*, Computer Journal 10 (1967), pp. 85-87.