Efficient Parallel Algorithm for Clique Separator Decomposition (Extended Abstract)

ELIAS DAHLHAUS

AND

MAREK KARPINSKI*

DEPT. OF COMPUTER SCIENCE
UNIVERSITY OF BONN

Abstract.

We design a first efficient parallel algorithm for decomposing an *arbitrary* graph by clique separators. The algorithm works in $O(\log^3 n)$ parallel time and $O(n^3)$ processors on a CREW PRAM. The theory of clique separators is directly related to the theory of elimination orderings of arbitrary graphs, and is used in efficient algorithms for Gaussian elimination of sparse symmetric matrices [Ro 70], [Ro 73], [RTL 76], [Ta 85]. It is the first sublinear parallel time (and therefore sequential parallel space) algorithm for the clique decomposition, and at the same time an optimal algorithm up to the polylogarithmic factor with respect to the best sequential $O(n^3)$ time algorithm of Tarjan [Ta 85].

1. Introduction.

One of the most important methods supporting the divide-and-conquer techniques in many graph algorithms is the method of computing graph separators either of small size ([Ta 72], [LNS 82], [HT 73]), or a special structure (e.g. *clique separators*, [Di 87], [Ta 85], [GJ 85], [RTL 76], [Wa 37]).

Recently, considerable progress has been achieved in designing efficient parallel algorithms for the first class of separators (cf. [Ra 87], [GM 87]). The second class of separators restrained attempts to design efficient parallel algorithms. The main reason for this was the fact that all the efficient sequential solutions [Ta 85] were based on a 'highly sequential' subroutine for computing a minimal ordering of a given graph [RTL 76].

In this paper we remedy this situation by developing a new method of graph extensions including all minimal chordal extensions of a given graph. This allows us to apply next the

*Supported in part by Leibniz Center for Research in Computer Science, by the DFG Grant KA 673/2-1, and by the SERC Grant GR-E 68297

generalization of Tarjan's [Ta 85] technique of finding clique separators as the intersections of pairs of neighbourhoods in the extension graphs.

The extension algorithm (Parallel-Fill-In; Section 3) works in $O(\log^3 n)$ time and $O(n^2)$ processors on a CREW PRAM for a graph of n vertices. (Only the completeness check for clique separators in the extension of a graph requires $O(n^3)$ processors and $O(\log n)$ time).

The Clique Separator Decomposition algorithm works in $O(\log^3 n)$ parallel time and $O(n^3)$ processors on a CREW PRAM.

2. Preliminaries.

By a graph we always mean an undirected graph $G = (V, E)$ with a *vertex set* V and an *edge set* E with no cycles and no multiple edges. An edge between x and y is denoted by $[x, y]$. The number of vertices $\#V$ is denoted by n, and the number of edges is denoted by m. For any set A, the size of A is denoted by $\#A$.

By a *cut* of vertices x and y of a graph $G = (V, E)$ we mean an inclusion minimal set of vertices which separates x and y. By a *complete* set V' we mean a subset of the vertex set, such that for all $x, y \in V'$ $[x, y] \in E$.

The *Clique Separator Decomposition* is the problem of computing all the clique separators (or a binary decomposition tree, cf. [Ta 85]) for a given input graph G.

A *clique separator* is a complete cut. A graph G with no induced cycle greater than 3 is called a *chordal* graph.

We shall need the following properties of clique separators and minimal chordal extension:

Theorem 1 (see [Ta 85] or [Ha 81]). In chordal graphs, each cut is a clique separator.

Theorem 2 (see [Ta 85]). Any clique separator of $G = (V, E)$ is also a clique separator in any (inclusion) minimal chordal extension (V, E') of G.

Our computational model will be a CREW PRAM (concurrent-read exclusive write) ([FW 78], [Co 85], [KR 88]).

3. The Algorithm.

3.1. An Outline.

Here we present a Clique Separator Decomposition algorithm which combines methods introduced recently in [DK 88b] with certain extensions of P. Klein's FOCS '88 ([Kl 88]) method for chordal graphs. The algorithm computes at first a chordal extension G' which preserves clique separators, together with a perfect elimination scheme on G'. The last step is to check for each clique separator of $G' = (V, E')$, whether it is a clique separator of G (that means a complete subgraph).

We begin with the algorithm which computes an ascending sequence $(C_1, \cdots, C_n)$, $C_i \subseteq V$, of "convex" sets of G' (see [FJ 86]), such that

i) $C_{i+1} \backslash C_i$ has only one element;

ii) $[x,y] \in E'$ iff $x,y \in C_i$ and there is a connected component $\tilde{C}$ of $V \backslash C_i$, such that $x,y \in N(\tilde{C})$;

iii) each clique separator of G is also a clique separator of G'.

Clearly this sequence $(C_i)_{i=1}^n$ defines a perfect elimination ordering for $G' = (V, E \cup E')$. Therefore G' is chordal. All these C_i are convex for G' in the sense of [FJ 86] (closed by chordless paths).

By a procedure *NONE*, we compute "convex" sets (see [FJ 86]) C_1, C_2, such that $\#C_1, \#(C_2 \backslash C_1), \#(V \backslash C_2) \leq \frac{2}{3}\#V$. Moreover, $(V, E' \cup E)$ shall preserve clique separators. The procedure *REFINE* computes for each "convex" set C suitable "convex" sets C_1, C_2, such that $C \subseteq C_1 \subseteq C_2$ and $\#(C_1 \backslash C), \#(C_1 \backslash C_2), \#(V \backslash C_2) \leq \frac{2}{3}\#(V \backslash C)$. Let $[x,y] \in E'_{\hat{C}}$ iff x and y are adjacent to the same connected component of $V \backslash \hat{C}$, where $\hat{C} = C_1$ or $\hat{C} = C_2$, and let $E_{\hat{C}} := E \cup E'_{\hat{C}}$.

Procedures *NONE* and *REFINE* are based on P. Klein's ([Kl 88]) new technique for chordal graphs.

Observation.

Whenever u and $v \in C_i \backslash C$ are in the same connected component w.r.t. E_{C_i}, then they are in the same connected component of $V \backslash C$ w.r.t. E, and vice versa.

3.2. The Fine Structure of the Algorithm.

Procedure NONE (G (V,E)).

Let $D := \{v \in V : \text{ degree } (v) \geq \frac{2}{3}\#V\}$.
If D is not complete, pick some $x,y \in D, [x,y] \notin E$ and let $C' := \{v | [v,x],[v,y] \in E\}$ be the common neighbourhood of x and y. Make C' a clique. (Comment: each $u,v \in C$ are on the cycle (x,u,y,v,x). Therefore each new edge of C is on a chordless cycle of G. That means that this extension $E \cup E'$ preserves clique separators.)

Let $C_1 := C_2$ be some subset of C', such that $\frac{1}{3}\#V \leq \#C_i \leq \frac{2}{3}\#V$. (Clearly $\#C' < \frac{1}{3}\#V$.)

If D is complete and $\#D \geq \frac{1}{3}\#V$, then C is some subset of D, s.t. $\frac{1}{3}\#V \leq \#C \leq \frac{2}{3}\#V$.

If $\#D \leq \frac{1}{3}\#V$, *D is complete,* and all connected components of $V \backslash D$ have a size $\leq \frac{2}{3}$:
Let $(\tilde{C}_1, \cdots, \tilde{C}_k)$ be an enumeration of these connected components; let for $j = 0, \cdots, k$: $C'_j := D \cup \bigcup_{i \leq j} \tilde{C}_i, \hat{E}'_{C_i} := \emptyset$. ($\#(C'_{j+1} \backslash C'_j) \leq \frac{2}{3}\#V$; note that $C'_k = V$).
Let C_1 be some C'_j, such that $\#C'_j \leq \frac{2}{3}\#V, \#(V \backslash C'_j) \leq \frac{2}{3}\#V$. Set $C_2 := C'_{j+1}$.

If D is complete or empty and there is a connected component $\hat{C}$ of $V \backslash D$, such that $\#(V \backslash D) \geq \frac{2}{3}\#V$:

Compute a spanning tree on $\hat{C}$ and, using this spanning tree, compute an enumeration $(x_i)_{i=1}^m$ of $\hat{C}$, such that each initial segment $\hat{C}_j := \{x_i | i \leq j\}$ is connected; let C'_j be the neighbourhood of $\hat{C}_j$ and $\hat{C}_{m+1} := V$.

Pick up a $\#(C'_j \backslash C'_{j-1}) \geq \frac{1}{3}\#V$ (if it exists, here $C'_0 := \emptyset$): Set $C_1 := C'_{j-1}$ and $C_2 := C'_j$. Otherwise, if such C'_j, C'_{j-1} do not exist, there is a C'_j, s.t. $\frac{1}{3}\#V \leq \#C'_j \leq \frac{2}{3}\#V$; pick up such a C'_j, set $C_1 := C_2 := C'_j$.

Set $NONE := (C_1, C_2)$.

End of Procedure *NONE*.

Now we proceed with the procedure *REFINE*.

Procedure $REFINE(G, C)$.

If for each connected component $K_1, \cdots, K_n$ *of* $V \backslash C$ $\#K_i \leq \frac{2}{3}\#(V \backslash C)$.: Let $C_i := C \cup \bigcup_{j<i} K_i$ and $\hat{E}' := \emptyset$; *Otherwise:* Let K_1 be the largest component; apply $REFINE'(C \cup K_1, C)$.

Assume $V \backslash C$ be connected:

Procedure $REFINE'(G, C)$.

Let $D := \{x \in G : d_{V \backslash C}(x) := \#\{y \in V \backslash C | [y, x] \in E\} \leq \frac{2}{3}\#(V \backslash C)\}$.

Let D' be the union of all connected components of D touching C.

(Low degree extension): Compute a spanning forest on D' and, using this, an enumeration $(x_i)_{i=1}^m$ of D', such that for each initial segment $u_j := \{x_i\}_{i=1}^j$, $u_j \cup C$ is connected. Let $\hat{C}_j := N(u_j) \cup C$. Here $N(u_j)$ is the *neighborhood* of u_j.

If $\#(\hat{C}_m \backslash C) \geq \frac{1}{3}\#(V \backslash C)$, then let $\hat{C}_{m+1} := V$ and pick up a $\#(\hat{C}_{j+1} \backslash \hat{C}_j) \geq \frac{1}{3}\#(V \backslash C)$ (if it exists). Let $C_1 := \hat{C}_j$ and $C_2 := \hat{C}_{j+1}$. If such $\hat{C}_j, \hat{C}_{j+1}$ do not exist, let C_1 be any $\hat{C}_j$ s.t. $\frac{1}{3}\#(V \backslash C) \leq \#(\hat{C}_j \backslash C) \leq \frac{2}{3}\#\hat{C}_j$ and $C_2 := \hat{C}_{j+1}$.

If $\#\hat{C}_m < \frac{1}{3}\#(V \backslash C)$: Let $C_1 := \hat{C}_m$ (possibly $C_1 = C$).

If $\exists x \in C_1$, s.t. $d_{V \backslash C_1}(x) \leq \frac{2}{3}\#(V \backslash C)$: Let $C_2 := C_1 \cup N(x)$.

Otherwise (high degree extension):

Let $(x_i)_{i=1}^k$ be an enumeration of all $x \in C_1$, such that $d_{V \backslash C_1}(x) \neq 0$ and let $F_j := \{x | [x, x_i] \in E$ for all $i = 1, \cdots, j\}$. If for some j, $\#F_j \leq \frac{2}{3}\#(V \backslash C)$, set $C_2 := C_1 \cup F_j$. Otherwise apply $NONE$ to $(G \upharpoonright F_k)$ with outputs C'_1 and C'_2. If $\#(C'_2 \backslash C'_1) \geq \frac{1}{3}\#(V \backslash C)$, let $C_1 := C_1 \cup C'_1$ and $C_2 := C_1 \cup C'_2$. If $\#(F_k \backslash C'_2) \geq \frac{1}{3}\#(V \backslash C)$, let $C_2 := C_1 \cup C'_2$. Otherwise $C_2 := C_1 \cup C'_1$.

End of procedure *REFINE*.

The whole algorithm works as follows:

Algorithm Parallel-Fill-In.

Input $G = (V, E)$.

Let $(C_1, C_2) := NONE(G)$. Apply $REC(C_1, C_2)$.

Procedure $REC(C_1, \cdots, C_k)$.

If $\#C_1 = 1$ and $\#(C_{i+1} \backslash C_1) = 1$ for each i, then stop.

Compute new "convex" sets:

Let C_1^0, C_2^0 be the two C_1, C_2 arising from the application of $NONE(G \restriction C_1)$; let C_1^i, C_2^i be the two C_1, C_2 arising from the application of $REFINE(G \restriction C_{i+1}, C_i)$

Add new edges:

For C_i and C_{i+1} and each connected component K of $C_{i+1} \backslash C_i$, let x_K be a vertex in C_i adjacent to K, such that $x_K \notin C = C_k^j \backslash C_j$ such that C maximal (x_K is in a minimal number of old and new known "convex" sets).

Join each vertex x of C_i adjacent to K with x_K by an edge $[x, x_K] \in E_{\text{new}}$ (hereby we have guaranteed that each $x, y \in C_i \backslash C_{i-1}^2, C_i \backslash C_{i-1}^1, C_i \backslash C$ etc. resp. those which are in the same connected component of $V \backslash C_{i-1}^2, \ldots$, resp. are also in the same connected component of $C_i \backslash C_{i-1}^2, C_i \backslash C_{i-1}^1, C_i \backslash C_{i-1}, \ldots$ etc. resp.):

Let K be a connected component of $C_{i+1} \backslash C_i^2$ and x_K be again an $x \in C_i^2$ contained in a minimal member of known "convex" sets C_i, C_i^j, and for each x adjacent to K an edge $[x, x_K] \in E_{\text{new}}$.

Do the same procedure also with the level $C_i^2 \backslash C_i^1$. (Hereby it is guaranteed that connectedness in $\tilde{C}_2 \backslash \tilde{C}_1$ and $V \backslash \tilde{C}_1$ for each $\tilde{C}_1 \subseteq \tilde{C}_2$, s.t. $\tilde{C}_1, \tilde{C}_2 \in \{C_i, C_i^j | i = 1 \ldots k, j = 1,2\}$ are equivalent (compare the observation)); $E := E \cup E_{\text{new}}$;

apply $REC(C_0^1, C_0^2, C_1, C_1^1, C_1^2 \ldots)$.

End of Procedure *REC*.

Output (V, E).

End of the Algorithm Parallel-Fill-In.

3.3. Analysis of the Algorithm:

1) The recursion depth of *REC* is $O(\log n)$, since the maximum cardinality of levels $C_{i+1} \backslash C_i$ goes down by at least $\frac{1}{3}\#(C_{i+1} \backslash C_i)$ at each step.
2) as mentioned above, the following is valid for the output (V, E) in $C_i \subset C_j$, $x, y \in C_j \backslash C_i$: x, y in the same connected component of $C_j \backslash C_i \iff x, y$ in the same connected component of $V \backslash C_i$.
3) Let $x < y$ if for some $k : y \in C_k$ but $x \notin C_k$. For the corresponding chordal extension E' of (V, E_{old}), $[x, y] \in E' \iff [x, y] \in E$ or x, y are adjacent to the same connected component K of $\{v | v < x\}$. But for each such connected component K, we have $x_K = x$. But $[y, x] = [y, x_K] \in E_{\text{new}}$.

Therefore the output (V, E) *is a chordal extension of* $G = (V, E_{\text{old}})$. *Moreover: Output* $(V, E) = (V, E')$.

4) *Old clique separators are preserved:* We have to prove this statement for each step of the application of *NONE* or *REFINE*.

NONE: Additional edges of common neighbours of $x, y \in D$ are in a cycle of legth four. Therefore clique separators are preserved. Let M be a connected subset of V and $C := M'$ be the set of neighbours of M. Let $x, y \in M'$ be adjacent to the same connected component of $V \backslash M'$. Then $x, y \in M' \backslash M$. Consider a path $x, y_1, \ldots, y_p, y$, s.t.$y_1 \ldots y_p$ is chordless and p is minimal. But then there is also no chord $[x, y_i]$ or $[y_j, y]$. Let $x, z_1, \ldots z, y$ be a chordless path, s.t. $z_1, \cdots z_p \in M$. But then the concatenation of these two paths form a cycle. Hereby the application of the procedure *NONE* preserves clique separators.

REFINE (Low degree extension): Assume any "convex" set $C := C_i$ is given. Assume that vertices adjacent to the same connected component of $V \backslash C$ form a clique in an extension E' of E preserving clique separators. Assume M is connected and intersects C. Let M' be defined as above and $C' := M' \cup C$. Let $x \in M' \backslash M$ and $y \in M' \backslash M$ be connected by a shortest path p in $V \backslash C'$. Then by the same arguments as before $[x, y]$ is a chord of a chordless cycle, if $[x, y] \notin E$.

Assume now $y \notin M' \backslash M$; that means $y \in C$. But y and $M \cap C$ are adjacent to the same connected component of $V \backslash C$. But then we find a y' and a path $P_1 \subseteq M$, such that (y, y', P_1, x, P) forms a chordless cycle in a clique separator preserving extension E' of E.

REFINE (High degree extension): Assume now $V \backslash C$ is connected, $D \subset C$ and $C' = C \cup \{y | \forall x \in D, [y, x] \in E\}$. Then we may assume that D forms a clique in some clique separator preserving extension E' of E. We have to prove that for $x, y \in C' \backslash C$ which are adjacent to the same connected component of $V \backslash C'$, we can join them by an edge and no clique separator is destroyed. Let $x \in C' \backslash C$, $y \in C' \backslash C$ and $p = (x, x_1, x_2, x_3, \ldots)$ be a shortest (chordless) path $\subseteq V \backslash C'$ connecting x and y. Let $d \in D$ and $[d, x_i], [d, x_j] \in E$ but not $[d, x_k] \in E$ for $i < k < j$. Then $(d, x_i, x_{i+1}, \ldots, x_j, d)$ forms a chordless cycle. Therefore $[x_i, x_j]$ can be added, such that no clique separator is destroyed. Since each x_i is not adjacent to at least one $d \in D$, it is possible to add a chord abbreviating p, such that x_i is not used and no clique separator is destroyed. Therefore an edge $[x, y]$ can be added, such that no clique separator of G is destroyed.

Now let $x \in C' \backslash C$, but $y \in C$. Then the same argument to add an edge $[x, y]$ works. By these observations no edge of $E' \backslash E$ destroys some clique separator.

We can conclude by the following:

Theorem 3. It is possible to compute a clique separator preserving extension G' of G in $O(\log^3 n)$ time and $O(n^2)$ processors on a CREW PRAM.

The clique separators of G' can be computed in $O(\log n)$ time and $O(n^2)$ processors. Only to check whether a clique separator of G' is complete in G requires $O(n^3)$ processors and $O(\log n)$ time.

Therefore our Main Result follows:

Theorem 4. There exists a parallel algorithm for Clique Separator Decomposition of an arbitrary graph with n vertices working in $O(\log^2 n)$ parallel time and $O(n^3)$ processors on a CREW PRAM. □

NONE: Additional edges of common neighbours of $x, y \in D$ are in a cycle of legth four. Therefore clique separators are preserved. Let M be a connected subset of V and $C := M'$ be the set of neighbours of M. Let $x, y \in M'$ be adjacent to the same connected component of $V \backslash M'$. Then $x, y \in M' \backslash M$. Consider a path $x, y_1, \ldots, y_p, y$, s.t.$y_1 \ldots y_p$ is chordless and p is minimal. But then there is also no chord $[x, y_i]$ or $[y_j, y]$. Let $x, z_1, \ldots z, y$ be a chordless path, s.t. $z_1, \cdots z_p \in M$. But then the concatenation of these two paths form a cycle. Hereby the application of the procedure *NONE* preserves clique separators.

REFINE (Low degree extension): Assume any "convex" set $C := C_i$ is given. Assume that vertices adjacent to the same connected component of $V \backslash C$ form a clique in an extension E' of E preserving clique separators. Assume M is connected and intersects C. Let M' be defined as above and $C' := M' \cup C$. Let $x \in M' \backslash M$ and $y \in M' \backslash M$ be connected by a shortest path p in $V \backslash C'$. Then by the same arguments as before $[x, y]$ is a chord of a chordless cycle, if $[x, y] \notin E$.

Assume now $y \notin M' \backslash M$; that means $y \in C$. But y and $M \cap C$ are adjacent to the same connected component of $V \backslash C$. But then we find a y' and a path $P_1 \subseteq M$, such that (y, y', P_1, x, P) forms a chordless cycle in a clique separator preserving extension E' of E.

REFINE (High degree extension): Assume now $V \backslash C$ is connected, $D \subset C$ and $C' = C \cup \{y | \forall x \in D, [y, x] \in E\}$. Then we may assume that D forms a clique in some clique separator preserving extension E' of E. We have to prove that for $x, y \in C' \backslash C$ which are adjacent to the same connected component of $V \backslash C'$, we can join them by an edge and no clique separator is destroyed. Let $x \in C' \backslash C$, $y \in C' \backslash C$ and $p = (x, x_1, x_2, x_3, \ldots)$ be a shortest (chordless) path $\subseteq V \backslash C'$ connecting x and y. Let $d \in D$ and $[d, x_i], [d, x_j] \in E$ but not $[d, x_k] \in E$ for $i < k < j$. Then $(d, x_i, x_{i+1}, \ldots, x_j, d)$ forms a chordless cycle. Therefore $[x_i, x_j]$ can be added, such that no clique separator is destroyed. Since each x_i is not adjacent to at least one $d \in D$, it is possible to add a chord abbreviating p, such that x_i is not used and no clique separator is destroyed. Therefore an edge $[x, y]$ can be added, such that no clique separator of G is destroyed.

Now let $x \in C' \backslash C$, but $y \in C$. Then the same argument to add an edge $[x, y]$ works. By these observations no edge of $E' \backslash E$ destroys some clique separator.

We can conclude by the following:

Theorem 3. It is possible to compute a clique separator preserving extension G' of G in $O(\log^3 n)$ time and $O(n^2)$ processors on a CREW PRAM.

The clique separators of G' can be computed in $O(\log n)$ time and $O(n^2)$ processors. Only to check whether a clique separator of G' is complete in G requires $O(n^3)$ processors and $O(\log n)$ time.

Therefore our Main Result follows:

Theorem 4. There exists a parallel algorithm for Clique Separator Decomposition of an arbitrary graph with n vertices working in $O(\log^2 n)$ parallel time and $O(n^3)$ processors on a CREW PRAM. □

Acknowledgement.

We thank Richard Karp, Michael Rabin, Bob Tarjan, and Avi Wigderson for interesting discussions on the topics related to this paper.

References.

[Co 85] Cook, S.A., *A Taxonomy of Problems with Fast Parallel Algorithms*, Information and Control 64 (1985), pp. 2-22

[DHK 88] Dahlhaus, E., Hajnal, P., and Karpinski, M., *Optimal Parallel Algorithm for the Hamiltonian Cycle Problem on Dense Graphs*, Proc. 29th IEEE FOCS (1988)

[DK 86a] Dahlhaus, E., and Karpinski, M., *Fast Parallel Computation of Perfect and Strongly Perfect Elimination Schemes*, Research Report No. 8513-CS, University of Bonn 1986; submitted for publication

[DK 86b] Dahlhaus, E., and Karpinski, M., *The Matching Problem for Strongly Chordal Graphs in in NC*, Research Report No. 855-CS, University of Bonn, 1986, pp. 1-7

[DK 88a] Dahlhaus, E., and Karpinski, M., *A Fast Parallel Algorithm for Computing All Maximal Cliques in a Graph and the Related Problems*, Proc. SWAT '88, Springer LNCS 318 (1988), pp. 139-144

[DK 88b] Dahlhaus, E., and Karpinski, M., *Fast Parallel Decomposition by Clique Separators*, Research Report No. 8525-CS, University of Bonn 1988

[Di 87] Distel, R., *Simplicial Decomposition of Graphs - Some Uniqueness Results*, Journal of Combinatorial Theory, Ser. B 42 (1987), pp. 133-145

[FJ 86] Farber, M., and Jamison, R.E., *Convexity in Graphs and Hypergraphs*, SIAM J. of Algebraic and Discrete Methods 7 (1986), pp. 433-444

[FW 78] Fortune, S., and Wyllie, S., *Parallelism in Random Access Machines*, Proc. 10th ACM-STOC (1978), pp. 114-118

[Ga 72] 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

[GM 87] Gazit, H., and Miller, G.L., *A Parallel Algorithm for Finding a Separator in Planar Graphs*, Proc. 28th IEEE FOCS (1987), pp. 238-248

[GS 87] Goldberg, M., and Spencer, T., *A New Parallel Algorithm for the Maximal Independent Set Problem*, Proc. 28th IEEE FOCS (1987), pp. 161-165

[GJ 84] Golumbic, M.C., and Jamison, R.E., *The Edge Intersection Graphs of Paths in a Tree*, J. Combin. Theory Ser. B; to appear

[GJ 85] Golumbic, M.C., and Jamison, R.E., *Edge and Vertex Intersections of Paths in Trees*, Discrete Math. 55 (1985), pp. 151-159

[Ha 81] Halin, R., *Graphentheorie II*, Wissenschaftliche Buchgesellschaft, Darmstadt 1981

[HJ 73] Hopcroft, J.E., and Tarjan, R.E., *Dividing a Graph into Triconnected Components*, SIAM J. Comput. 2 (1973), 135-158

[KR 88] Karp, R., and Ramachandran, V., *A Survey of Parallel Algorithms for Shared-Memory Machines*, Research Report No. UCB/CSD 88/407, University of California, Berkeley (1988); to appear in Handbook of Theoretical Computer Science, North Holland (1988).

[Kl 88] Klein, Ph., *Efficient Parallel Algorithms on Chordal Graphs*, Proc. 29th IEEE FOCS (1988)

[LT 79] Lipton, R.J., and Tarjan, R.E., *A Separator Theorem for Planar Graphs*, SIAM J. Appl. Math. 36 (1979), pp. 177-189

[Lu 85] Luby, M., *A Simple Parallel Algorithm for the Maximal Independent Set Problem*, Proc. 17th ACM STOC (1985), pp. 1-9

[NNS 87] Naor, J., Naor, M., and Schäffer, A., *Fast Parallel Algorithms for Chordal Graphs*, Proc. 19th ACM STOC (1987), pp. 355-364

[Ra 87] Rao, S., *Finding New Optimal Separators in Planar Graphs*, Proc. 28th IEEE FOCS (1987), pp. 225-237

[Ro 70] Rose, D., *Triangulated Graphs and the Elimination Process*, J. Math. Anal. Appl. 32 (1970), pp. 579-609

[Ro 73] Rose, D., *"A Graph-Theoretic Study of the Numerical Solution of Sparse Positive Definite Systems of Linear Equations"*, in: Graph Theory and Computing, R. Read ed., (Academic Press, New York 1973), pp. 183-217

[RTL 76] Rose, D., Tarjan, R.E., and Lueker, G., *Algorithmic Aspects of Vertex Elimination on Graphs*, SIAM J. Comput. 5, pp. 266-283

[TNS 82] Takamizawa, K., Nishizeki, T., and Saito, N., *Linear-time Computability of Combinatorial Problems on Series-Parallel Graphs*, J. ACM, Vol. 29 (1982), pp. 623-641

[Ta 72] Tarjan, R.E., *Depth-first Search and Linear Graph Algorithms*, SIAM J. Comput 1 (1972), pp. 146-160

[Ta 85] Tarjan, R.E., *Decomposition by Clique Separators*, Discrete Mathematics 55 (1985), pp. 221-232

[TY 84] Tarjan, R.E., and Yannakakis, M., *Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Hypergraphs*, SIAM J. Comput. 13 (1984), pp. 566-579

[TY 85] Tarjan, R.E., and Yannakakis, M., *Simple Linear-Time Algorithms to Test Chordality of Graphs, Test Acyclicity of Graphs, Test Acyclicity of Hypergraphs, and Selectively Reduce Hypergraphs; Addendum*, SIAM J. Comput. 14 (1985), pp. 254-255

[Wa 37] Wagner, K., *Über eine Eigenschaft der ebenen Komplexe*, Mathematische Annalen 114 (1937), pp. 570-590

[Wh 81] Whitesides, F.H., *An Algorithm for Finding Clique Cut-sets*, Information Processing Letters 12 (No 1) (1981), pp. 31-32