NC-ALGORITHMS FOR GRAPHS WITH SMALL TREEWIDTH

Hans L. Bodlaender

RUU-CS-88-4
Februari 1988

Rijksuniversiteit Utrecht

Vakgroep Informatica

Budapestlaan 6 3584 CD Utrecht
Corr. adres: Postbus 80.012 3508 TA Utrecht
Telefoon 030-53 1454
The Netherlands

NC-ALGORITHMS FOR GRAPHS WITH SMALL TREEWIDTH

Hans L. Bodlaender

Technical Report RUU-CS-88-4
February 1988

Department of Computer Science
University of Utrecht
P.O. Box 80.012, 3508 TA Utrecht
The Netherlands

Definition.

Let $G = (V, E)$ be a graph. A tree-decomposition of G is a pair $(\{X_i | i \in I\}, T = (I, F))$, with $\{X_i | i \in I\}$ a family of subsets of V and T a tree, with the following properties:

- $\bigcup_{i \in I} X_i = V$
- For every edge $e = (v, w) \in E$, there is a subset $X_i, i \in I$ with $v \in X_i$ and $w \in X_i$
- For all $i, j, k \in I$, if j lies on the path in T from i to k, then $X_i \cap X_k \subseteq X_j$.

The treewidth of a tree-decomposition $(\{X_i | i \in I\}, T)$ is $\max_{i \in I} |X_i| - 1$. The treewidth of G, denoted treewidth(G) is the minimum treewidth of a tree-decomposition of G, taken over all possible tree-decompositions of G.

For a set S, clique(S) denotes the graph $(S, \{(v, w) | v, w \in S, v \neq w\})$. For graphs $G = (V, E), H = (W, F), G \cup H$ denotes the (possibly non-disjoint) union $(V \cup W, E \cup F)$. For $W \subseteq V, G[W]$ denotes the subgraph of $G = (V, E)$ induced by $W : G[W] = (W, \{(v, w) | v, w \in W \text{ and } (v, w) \in E\})$.

Next we give some graph-theoretic results, which will be used in later sections.

Lemma 2.1

Let $(\{X_i, i \in I\}, T = (I, F))$ be a tree-decomposition of $G = (V, E)$. Suppose $W \subseteq V$ forms a clique in G. Then $\exists i \in I : W \subseteq X_i$.

Proof.

Use induction to the clique size $|W|$. For $|W| \leq 2$, the result follows directly from the definition of tree-decomposition. Suppose the lemma holds up to clique size $l - 1, l \geq 3$. Consider a clique $W \subseteq V$, with $|W| = l$, and suppose the lemma does not hold for W. Choose a vertex $w \in W$, and let $W' = W - \{w\}$. Let $I' \subseteq I$ be the set $\{i \in I | W' \subseteq X_i\}$. By induction $I' \neq \emptyset$. Note that $w \in X_i \Rightarrow i \notin I'$. Now choose a node $i' \in I'$, and a node $i \in I$ with $w \in X_i$. Consider the path in T from i to i'. Let i'' be the last node on this path with $i'' \in I'$, and let i''' be the next node on this path. Now, for every $w' \in W'$, there must be a node $j_{w'}$, with $\{w, w'\} \subseteq X_{j_{w'}}$. Consider the path from i'' to $j_{w'}$. There are two cases.

Case 1: This path does not use i'''. In this case, the path in T from i to $j_{w'}$ uses i''. Now $w \in X_i, w \in X_{j_{w'}}$, hence $w \in X_{i''}$, contradiction.

Case 2: This path uses i'''. Now $w' \in X_{i''}$ and $w' \in X_{j_{w'}}$, hence $w' \in X_{i'''}$.

It follows that for all $w' \in W' : w' \in X_{i'''}$, hence $i''' \in I'$, which contradicts the assumption that i'' was the last node on the path from i to i', that was in I'. □

Definition.

A tree-decomposition $(\{X_i, i \in I\}, T = (I, F))$ of a graph $G = (V, E)$ is called <u>full</u>, iff

(i) $\forall i, j \in I : |X_i| = |X_j|$, and

(ii) $\forall (i, j) \in F : X_i \not\subseteq X_j$ and $X_j \not\subseteq X_i$.

Lemma 2.2
Let $G = (V, E)$ be a graph with treewidth$(G) \leq k$ and $|V| \geq k+1$. Then G has a full tree-decomposition with treewidth k.

Proof.
Start with any tree-decomposition of G with treewidth $\leq k$, and repeat the following operations, until a full tree-decomposition is obtained:

1. If there are $(i_0, i_1) \in F$ with $X_{i_0} \subseteq X_{i_1}$ or $X_{i_1} \subseteq X_{i_0}$, then we make a new tree-decomposition by merging i_0 and i_1. Take $(\{X_i | i \in I - \{i_1\}\}, T' = (I - \{i_1\}, \{(i,j) | i,j \in I - \{i_1\}$ and $(i,j) \in F)\}$ or $(i = i_0$ and $(i_1, j) \in F)$ or $(j = i_1$ and $(i_1, i) \in F))$. This is again a tree-decomposition of G with treewidth $\leq k$, but with a smaller index set I.

2. If there is an $i_0 \in I$ with $|X_{i_0}| \leq k+1$, then either operation 1 can be applied, or there is an adjacent node $i_1 \in I$ with $\exists v \in X_{i_1} : v \in X_{i_0}$. Make a new tree-decomposition by adding v to $X_{i_0} : (\{X_i' | i \in I\}, T = (I, F))$, with $X_{i_0}' = X_{i_0} \cup \{v\}$, and $X_i' = X_i$ for $i \neq i_0$. This is again a tree-decomposition, with the same size of the index set I, and $\sum_{i \in I} |X_i|$ is increased by one.

Clearly, operations 1 and 2 can be applied only a finite number of times. If neither operation 1 or 2 applies, we have a full tree-decomposition of G with treewidth k. □

3 An NC-algorithm for recognizing graphs with small treewidth.

In this section we show that recognizing graphs with treewidth $\leq k$, and finding the corresponding tree-decomposition, are in NC, for constant k. The algorithm is quite inefficient in the use of processors, as it uses $\mathcal{O}(n^{3k+4})$ processors. The algorithm uses $\mathcal{O}(\log n)$ time on a CRCW PRAM.

Suppose $G = (V, E)$ is the input-graph.

First all $(k+1)$-element vertex sets, which are a separator of G, are computed, and numbered $S_1, S_2, \ldots, S_i, \ldots$. For each such S_i, the connected components of $G[V - S_i]$ are numbered $S_i^1, S_i^2, \ldots, S_i^j, \ldots$. (Note the difference with the algorithm of Arnborg, Corneil and Proskurowski [2], where k-element vertex sets were considered, instead of $(k+1)$-element sets.)

For each pair of $(k+1)$-element separators $S_i, S_j, i \neq j$, let $R_{i,j}$ denote the set of vertices v, such that v has a path to a vertex in $S_i - S_j$, which avoids S_j, and a path to a vertex in $S_j - S_i$, which avoids S_i.

Definition.

(i) A pair (S_i, S_i^j) is called <u>good</u>, iff $G[S_i \cup S_i^j] \cup$ clique(S_i) has treewidth $\leq k$.

(ii) A triple $(S_i, S_j, R_{i,j})$ is called good, iff $G[S_i \cup S_j \cup R_{i,j}] \cup$ clique$(S_i) \cup$ clique(S_j) has treewidth $\leq k$.

The next three lemma's give the essential steps of the algorithm.

Lemma 3.1
Let $|V| \geq k+3$. Then: treewidth$(G) \leq k$, if and only if there exists a $(k+1)$-vertex separator S_i, with all (S_i, T_i^j) are good.

Proof.
$(\Rightarrow)$ Consider a full tree-decomposition $(\{X_i | i \in I\}, T = (I, F))$ of G. Take $S_i = X_r$ for an arbitrary node $r \in I$.
$(\Leftarrow)$ For each $G[S_i \cup T_i^j] \cup$ clique(S_i) there exists a tree-decomposition with treewidth $\leq k$. By lemma 2.1, each of these tree-decompositions contains an X_{i_0}, with $S_i \subseteq X_{i_0}$, so $S_i = X_{i_0}$. We now can compose a tree-decomposition of G of the tree-decompositions of the $G[S_i \cup T_i^j] \cup$ clique(S_i) graphs, by identifying the nodes i_0 with $X_{i_0} = S_i$. □

Lemma 3.2
Consider $S_i, S_k, R_{i,k}$ with $R_{i,k} \neq \emptyset$.
$(S_i, S_k, R_{i,k})$ is good, if and only if there exists a $k+1$-vertex cutset $S_j \subseteq S_i \cup S_k \cup R_{i,k}$, such that

(i) $S_j \supseteq (S_i \cup R_{i,j}) \cap (S_k \cup R_{j,k})$

(ii) $(S_i, S_j, R_{i,j})$ and $(S_j, S_k, R_{j,k})$ are good

(iii) $|R_{i,j}| \leq \frac{1}{2}|R_{i,k}|; |R_{j,k}| \leq \frac{1}{2}|R_{i,k}|$

(iv) For all m with $S_j^m \cap (S_i \cup S_k \cup R_{i,j} \cup R_{j,k}) = \emptyset$ and $S_j^m \subseteq R_{i,k} : (S_j, S_j^m)$ is good.

Proof.
$(\Rightarrow)$ Consider a full tree-decomposition $(\{X_i, i \in I\}, T = (I, F))$ of $G' = G[S_j \cup S_k \cup R_{i,k}] \cup$ clique$(S_i) \cup$ clique(S_j), with treewidth k. Note that there must be $i_0 \in I$ with $X_{i_0} = S_i$ and $i_1 \in I$ with $X_{i_1} = S_k$, by lemma 2.1. One may suppose that i_0 and i_1 are leaves in the tree T, else one can remove some nodes from T and still have a full tree-decomposition of G' with treewidth k. For each node $i' \in I$ on the path from i_0 to $i_1, i' \neq i_0, i' \neq i_1$, we have that $X_{i'}$ is a $k+1$-vertex cutset of G' (and hence of G). As $R_{i,k} \neq \emptyset, |I| \geq 3$, and so there is at least one such node i'. For each such $i' \in I$, let $X_{i'} = S_{\alpha(i')}$, and $s(i') = \max(|R_{i\alpha(i')}|, |R_{k\alpha(i')}|)$. Suppose i_2 has minimal $s(i_2)$ over all i' on the path from i_0 to $i_1, i' \notin \{i_0, i_1\}$. We claim that $s(i_2) \leq \frac{1}{2}|R_{i,k}|$. Note that $X_{i'} \cap R_{i\alpha(i_2)} \neq \emptyset \Leftrightarrow i'$ belongs to the tree in $T - \{i_2\}$ which also contains S_i, and similarly, $X_{i'} \cap R_{k(i_2)} \neq \emptyset \Leftrightarrow i'$ belongs to the tree in $T - \{i_2\}$ which also contains S_k. W.l.o.g. suppose $s(i_2) = |R_{k\alpha(i_2)}|$. Let i_3 be the next node on the path from i_2 to i_1. Now $R_{k\alpha(i_3)} \subset R_{k\alpha(i_2)}$ and $R_{k\alpha(i_3)} \neq R_{k\alpha(i_2)}$; and $v \in R_{k\alpha(i_2)} \Rightarrow v \notin R_{i\alpha(i_3)}$. This follows from the definition of tree-decomposition. It follows that $|R_{i\alpha(i_3)}| \geq s(i_2)$ and $|R_{i\alpha(i_3)}| \leq |R_{ik}| - |R_{k\alpha(i_2)}| = |R_{ik}| - s(i_2)$, hence $s(i_2) \leq \frac{1}{2}|R_{ik}|$.

Now let $S_j = X_{i_2}$. Property (i) follows from the observations, made before and the definition of tree-decomposition. Properties (ii) and (iv) follow because the corresponding graphs are subgraphs of $G^1 \cup$ clique(S_j), which clearly has treewidth $\leq k$.

($\Leftarrow$) Note that if $S_j^m \cap (S_i \cup S_k \cup R_{i,j} \cup R_{j,k}) \neq \emptyset$, then $S_j^m \cap R_{i,k} \subseteq R_{i,j} \cup R_{j,k}$. Further, note that if a vertex belongs to two or more of the sets $S_i \cup R_{i,j} \cup S_j$, $S_j \cup R_{j,k} \cup S_k$, $S_j \cup S_j^m$, for any m with $S_j^m \cap (S_i \cup S_k \cup R_{i,j} \cup R_{j,k}) = \emptyset$ and $S_j^m \subseteq R_{i,k}$, then it belongs to S_j.

Now make tree-decompositions with treewidth $\leq k$ of $G[S_i \cup S_j \cup R_{i,j}] \cup$ clique(S_i)$\cup$ clique(S_j), $G[S_j \cup S_k \cup R_{j,k}] \cup$ clique(S_j)$\cup$ clique(S_k), and $G[S_j \cup S_j^m] \cup$ clique(S_j), for all m as before. Each of these tree-decompositions contains an i' with $X_{i'} = S_j$. By identifying all these i' and so "glueing" the tree-decompositions together we obtain a new tree-decomposition with treewidth $\leq k$. By the two observations made above, it follows that this is indeed a correct tree-decomposition of $G[S_i \cup S_k \cup R_{i,k}] \cup$ clique(S_i)$\cup$ clique(S_k). □

Lemma 3.3
Consider (S_i, S_i^j) with $|S_i^j| \geq k+1$. (S_i, S_i^j) is good, if and only if there exists a $(k+1)$-vertex cutset S_j, such that

(i) $(S_i, S_j, R_{i,j})$ is good

(ii) for all m, with $S_j^m \cap (R_{i,j} \cup S_i) = \emptyset$ and $S_j^m \subseteq S_i^j : (S_j, S_j^m)$ is good and $|S_j^m| \leq \frac{1}{2}|S_i^j|$.

(iii) $|R_{i,j}| \leq \frac{1}{2}|S_i^j|$.

Proof.
($\Rightarrow$) Consider a full tree-decomposition $(\{X_i | i \in I\}, T = \{I, F\})$ of $G' = G[S_i \cup S_i^j] \cup$ clique(S_i). For each $i' \in I$ and each component $T' = (I', F')$ of $T - \{i'\}$, let $s(T', i') = |\{v \in X_{i''} \cap S_i^j | i'' \in I' \text{ and } v \notin X_{i'}\}|$. For all $i' \in I$, define $s(i')$ to be the maximum of $s(T', i')$ over all connected components T' of $T - \{i'\}$. Let $i_0 \in I$ be the node, such that $s(i_0)$ is minimal over all $i' \in I$, and $|X_{i_0} \cap S_i^j|$ is minimal over all i' with minimal $s(i')$. From $|S_i^j| \geq k+1$, it easily follows that i_0 is an internal node of G', and hence X_{i_0} is a $(k+1)$-vertex cutset of G', and hence also of G. Take $S_j = X_{i_0}$.

We claim that $s(i_0) \leq \frac{1}{2}|S_i^j|$. Let i_1 be the node that is adjacent to i_0 and in the component T' of $T - \{i_0\}$ with $s(T', i_0) = s(i_0)$. Consider the component T'' of $T - \{i_1\}$, that contains i_0. From the definition of tree-decomposition it follows that for all $v \in S_i^j : v \in X_{i''}$ for some i'' in T' and $v \notin X_{i_0} \Rightarrow v \notin X_{i''}$ for all i'' in T'' or $v \in X_{i_1}$. So $s(T', i_0) + s(T'', i_1) \leq |S_i^j|$. If there is an $v \in S_i^j$, with $v \in X_{i_0}$ and $v \notin X_{i_1}$, then the result follows. All components T''' of $T - \{i_1\}$, except T'', are contained in T', and for each of these components we have $s(T''', i_1) < s(T', i_0)$. So $s(T'', i_1) \geq s(T', i_0)$, and hence $s(i_0) = s(T', i_0) \geq \frac{1}{2}|S_i^j|$.

Also, if $s(i_1) > s(i_0)$, one easily derives that $s(i_0) \leq \frac{1}{2}|S_i^j|$. So suppose $s(i_1) = s(i_0)$ and $v \in S_i^j \cap X_{i_0} \Rightarrow v \in X_{i_1}$, i.e. $|S_i^j \cap X_{i_0}| \geq |S_i^j \cap X_{i_1}|$. By definition of $i_0 : |S_i^j \cap X_{i_0}| = |S_i^j \cap X_{i_1}|$. It follows that $S_i^j \cap X_{i_0} = S_i^j \cap X_{i_1}$. One can derive that $S_i \cap X_{i_0} = S_i \cap X_{i_1}$, and hence $X_{i_0} = X_{i_1}$. So the tree-decomposition was not full. Contradiction. So the claim $s(i_0) \leq \frac{1}{2}|S_i^j|$ follows.

One can now check without difficulty that conditions (i) - (iii) are fulfilled, when taking $S_j = X_{i_0}$.
($\Leftarrow$) Similar as in lemma 3.2. □

With help of these 3 lemma's, an NC-algorithm, using $\mathcal{O}(n^{3k+4})$ processors and $\mathcal{O}(\log n)$ time on a CRCW PRAM can be derived.

First, determine the set of $(k+1)$-vertex cutsets S_i.

Secondly, determine which (S_i, S_i^j) are good for $|S_i^j| \leq k$, and which $(S_i, S_j, R_{i,j})$ are good, for $R_{i,j} = \emptyset$. This can be done with $\mathcal{O}(n^{2k})$ processors in $\mathcal{O}(1)$ time.

Then, in $\log n$ phases, one can determine for all (S_i, S_i^j) and $(S_i, S_k, R_{i,k})$ whether they are good, with lemma 3.2 and 3.3; in phase l one considers S_i^j and $R_{i,k}$ with $|S_i^j|, |R_{i,k}| \in \{2^{l-1}+1, \ldots, 2^l\}$.

Finally, verifying whether treewidth$(G) \leq k$ can be done in $\mathcal{O}(1)$ time, with lemma 3.1, with $\mathcal{O}(n^{k+2})$ processors.

We also note that finding a corresponding tree-decomposition can be done in $\mathcal{O}(\log n)$ time, with about the same number of processors, using the construction method for the tree-decompositions, indicated in the proofs of lemma 3.1 - 3.3.

Theorem 3.4
For each constant k, there exists an algorithm that uses $\mathcal{O}(\log n)$ time and $\mathcal{O}(n^{3k+4})$ processors on a CRCW PRAM that determines whether a given input graph has treewidth $\leq k$, and if so, finds a corresponding tree-decomposition.

The algorithm is quite inefficient in the use of processors. By parallizing the algorithm of Arnborg, Corneil and Proskurowski [2] one obtains without much difficulty the following result.

Theorem 3.5
For each constant k, there exists an algorithm that uses $\mathcal{O}(n)$ time and $\mathcal{O}(n^{k+1})$ processors on a CRCW PRAM, that determines whether a given input graph has treewidth $\leq k$, and if so, finds a corresponding tree-decomposition.

4 NC-algorithms for NP-complete problems, restricted to graphs with small treewidth.

In this section we give a method to design NC-algorithms, for a large number of graph problems, that are NP-complete for arbitrary graphs, when restricted to the class of graphs with treewidth $\leq k$. The algorithms are based upon the sequential algorithms in [7] (but several other algorithms on graphs with treewidth $\leq k$ can be dealt with in a similar way).

The sequential algorithms are of the following form: we suppose a tree-decomposition of G is given. For each node $i \in I$ we compute a table TABLE(i). For computing TABLE(i) one needs TABLE(j) for all sons j of i in T. The time to compute such a table is in several cases $\mathcal{O}(\#$ sons of j), in other cases it is polynomial in n. A close observation of the algorithms in [7] learns, that if i has $\mathcal{O}(1)$ sons, then either TABLE(i) can be computed in $\mathcal{O}(1)$ time, or TABLE(i) can be computed with an NC-algorithm. We will not give the details here, but refer the reader to [7]. A large number of problems can be dealt with in this manner, including vertex cover, dominating set, domatic number, chromatic number, monochromatic triangle, feedback vertex set, feedback arc set, minimum maximal matching, partition into triangles, partition into Hamiltonian subgraphs, partition into forests, partition into cliques, clique, independent set, induced path, balanced complete bipartite subgraph, cubic subgraph, Hamiltonian completion, Hamiltonian circuit, Graph contractability to a fixed graph H, Minor(H), for a fixed graph H, Graph Homomorphism to a fixed graph H, Kernel, k-closure, degree-k spanning tree, Maximum leaf spanning tree, Bounded Diameter Spanning tree, Steiner Tree in Graphs, Max Cut, Minimum Cut into Bounded Sets, Longest Circuit, Longest Path, Chromatix Index [8].

Each of these problems can be solved in NC, when restricted to graph with treewidth $\leq k$. The following two theorems give the main idea.

Theorem 4.1
Every binary tree $T = (V, E)$ has a tree-decomposition $\{\{X_i | i \in I\}; T' = (I, F)\}$ with treewidth ≤ 3, and the depth of T' is at most $2\lceil \log_{\frac{5}{4}}(|V|)\rceil$, and T' is a binary tree.

Proof.
Our result is based upon the method of parallel tree-contraction of Miller and Reif [11]. We will obtain a series of (rooted) trees $T = T_0 = (V_0, E_0), T_1 = (V_1, E_1), T_2 = (V_2, E_2), \ldots, T_r = (V_r, E_r)$, with $|V_r| = 1$. To each $v \in V_i$ we assign a set $\varphi(v, i) \subseteq V$ representing the set of "vertices that are contracted to v". Define $\varphi(v, 0) = \{v\}$.

Each T_{i+1} is obtained from T_i by applying the following two operations in parallel:

1. RAKE. For each node $v \in V_i$, with at least 1 child of v is a leaf in T_i : remove the children from v that are a leaf, and take $\varphi(v, i+1) = \bigcup\{\varphi(w, i) | w = v$ or w is a child of v, and w is a leaf $\}$.
2. COMPRESS. A sequence of nodes $v_1, \ldots, v_k$ is a chain if v_{j+1} is the only child of v_j, and v_k has exactly one child and that child is not a leaf. Now, in each maximal chain, identify v_i and v_{j+1} for j odd and $1 \leq j < k$. Let w_i be the new node. We take $\varphi(w, i+1) = \varphi(v_j, i) \cup \varphi(v_{j+1}, i)$.

Miller and Reif [11] showed that after $\lceil \log_{\frac{5}{4}} n\rceil$ simultaneous applications of RAKE and COMPRESS, T is reduced to a single vertex. So it follows that $r \leq \lceil \log_{\frac{5}{4}} n\rceil$.
Each $\varphi(v, i)$ represents the set of vertices that are contracted to v in i contractions. Note that each $\varphi(v, i)$ induces a connected subtree of T and that for each i, all $\varphi(v, i)$ are disjunct and partition V. Furthermore, if $(v, w) \in E$, then either $\exists x \in V_i$ with $v, w \in \varphi(x, i)$ or $\exists x, y \in V_i$ with $(x, y) \in E_i$ and $v \in \varphi(x, i)$ and $w \in \varphi(y, i)$.

Now define $\beta(v, i) = \{w \in \varphi(v, i) | \exists w' \in V$ with $(w, w') \in E$ and $w' \notin \varphi(v, i)\}$. $\beta(v, i)$ represents the vertices that are at the "border" of $\varphi(v, i)$. The following properties hold:

1. If $v \in V_i$, and the degree of v in T_i is 3, then $|\varphi(v, i)| = |\beta(v, i)| = 1$.
2. If $v \in V_i$, then $|\beta(v, i)|$ is at most the degree of v in T_i.

To make $V_o, \ldots, V_r$ disjoint we label all $v \in V_i$ with i. We now give a "first version" of a tree-decomposition $\{\{X_i, i \in I\}, T' = (I, F)\}$ of T, which "almost" satisfies the constraints. We take $I = \bigcup_{i=o}^{r} V_i$. If vertices v^i, w^i, or $v^i, w^i, x^i \in V_i$ are contracted to y^{i+1}, then y^{i+1} is the father of v^i, w^i (and x^i) in T'. If v is unchanged by going from T_i to T_{i+1}, the v^{i+1} is the father of v^i in T'. Further we take $X_w = \bigcup\{R_x | x$ is a son of W in $T'\}$.

We claim that this is a correct tree-decomposition of T with treewidth ≤ 4, and no node in T' has degree ≥ 4.

First, for each edge $(v, w) \in E$, there must be an i, such that $v \in \varphi(x^i, i), w \in \varphi(y^i, i)$ and $v, w \in \varphi(z^{i+1}, i+1)$, for some x^i, y^i, z^{i+1}. Now $v \in \beta(x^i, i), w \in \beta(y^i, i)$, so $v, w \in X_{z^{i+1}}$.

Secondly, for each $v \in V$: on each level i of the tree T', there is exactly one $w^i \in V_i$ with $v \in \varphi(w^i, i)$, and hence at most one $w^i \in V_i$ with $v \in \beta(w^i, i)$, so also at most one $w^i \in V_i$ with $v \in X_{w^i}$. Furthermore, if $v \in X_{w^i}$, and x^{i+1} is the father of w^i in T', then either $v \in X_{x^{i+1}}$, or v does not belong to any X_y for y on a level, higher than $i+1$. It follows that we have a correct tree-decomposition.

For all $w \in I, |X_w| \leq 4$: either two vertices with degree ≤ 2 are contracted, or a vertex with degree 3 is contracted with one or two leaves.
We show now that the tree-decomposition can be slightly modified, such that T' is binary, and the treewidth ≤ 3.

For a node with 3 children, use the transformation in figure 4.1.

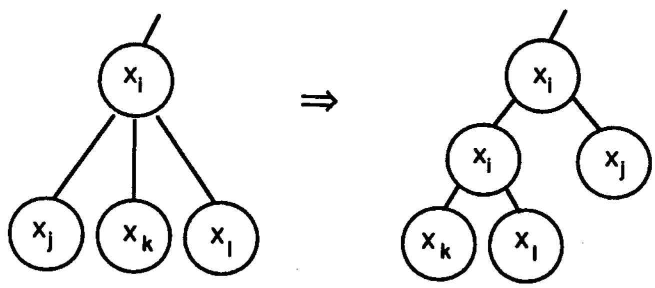

Figure 4.1

If $|X_{w^{i+1}}| = 4$, then w^{i+1} is obtained by contracting two nodes with degree 2, say x^i and y^i. Suppose $\beta(x^i, i) = \{v_0, v_1\}; \beta(y^i, i) = \{v_2, v_3\}$ and $(v_1, v_2) \in E$. Then transform as in figure 4.2.

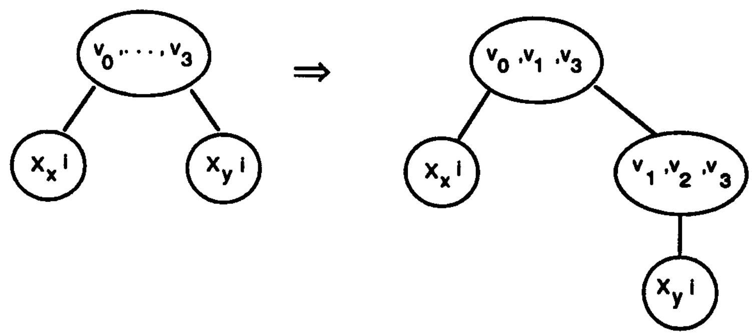

Figure 4.2

Note that $\beta(w^{i+1}, i+1) = \{v_0, v_3\}$. A new correct tree-decomposition with treewidth ≤ 3 results with the depth of T' increased by at most a factor 2, and T' is a binary tree. □

Theorem 4.2
Let $G = (V, E)$, with $|V| = n$, and treewidth(G) $\leq k$. Then G has a tree-decomposition $(\{X_i | i \in I\}, T = (I, F))$ with T a binary tree with depth $\leq 2\lceil \log_{\frac{5}{4}}(2n) \rceil$, and with treewidth of this decomposition $\leq 3k + 2$.

Proof.
Let $(\{X_i | i \in I_1\}, T_1 = (I_1, F_1))$ be a tree-decomposition of G with treewidth $\leq k$, and $|I_1| \leq n$. By transforming nodes in T_1 as in figure 4.3, one obtains a new tree-decomposition $(\{X_i | \in I_2\}, T_2 = (I_2, F_2))$ of G with treewidth $\leq k$ and $|I_2| \leq 2n$, and T is

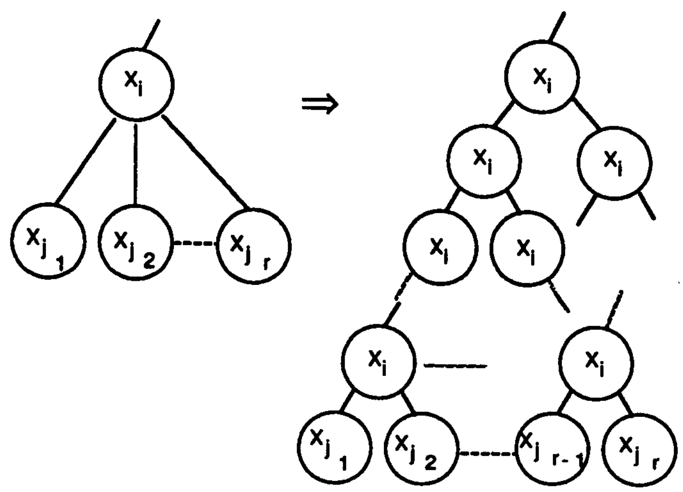

Figure 4.3.

a binary tree. Let $(\{Y_i | i \in I_3\}, T_3 = (I_3, E_3)\}$ be a tree-decomposition of T_2, with T_3 a binary tree with depth $\leq 2\lceil \log_{\frac{5}{4}} |I_2| \rceil \leq 2\lceil \log_{\frac{5}{4}}(2n) \rceil$, and treewidth of this tree-decomposition ≤ 3, (cf. theorem 4.1.). Then $(\{Z_i | i \in I_3\}, T_3 = (I_3, E_3))$ with $Z_i = \bigcup\{X_j | j \in Y_i\}$ is a tree-decomposition of G, with the required properties. □

More-over, the tree-decompositions, indicated in theorem 4.1 and 4.2 can be found in polylogarithmic parallel time. From the results of Miller and Reif [11], it easily follows that the construction indicated in the proof of theorem 4.1 can be carried out in $\mathcal{O}(\log^2 n)$ time on a CRCW PRAM, and with a probabilistic algorithm, in $\mathcal{O}(\log n)$ expected time on a CRCW PRAM. One easily sees that the construction, indicated in the proof of theorem 4.2 can be carried out in the same time.

So, for graphs $G = (V, E)$ with treewidth$(G) \leq k = \mathcal{O}(1)$, we can find a tree-decomposition of G, with treewidth $\mathcal{O}(1)$, and T a binary tree with depth $\mathcal{O}(\log n)$, with an NC-algorithm. Now the TABLE's can be computed level by level: first compute the tables for all $i \in I$ with maximum distance to the root of T, then for all $i \in I$ with distance one smaller, etc. Each step either takes $\mathcal{O}(1)$ time, or can be carried out in NC. After $\mathcal{O}(\log n)$ such steps, we have found the table for the root of T. Finding the answer to the query then costs $\mathcal{O}(1)$ time, or is easily seen to be in NC.

5 Remarks and open problems.

One practical disadvantage of the sequential algorithms on graphs with small treewidth [3,4,8,14] is that of the large constants involved in the algorithms. For instance, Arnborg and Proskurowski gave a linear algorithm for Hamiltonian circuit (among others) [4], but consider their algorithm unfeasible for $k \geq 8$.
The NC-algorithms, given in this paper will only add to this problem of the large constant factor. However, parallelism will help in a very straightforward way to decrease the running time, as the large constant is in a large extent due to a large number of actions which can be carried out in parallel. So, in many cases, using a large, but constant number of processors may decrease the running time by a large, but again constant factor. Thus, the results in this paper seem to be of mainly theoretical interest.

However, the theoretical importance of the results in this paper are stressed by the fact that a large number of classes of graphs have associated a constant c with them, such that each graph in the class has treewidth $\leq c$. Examples of such classes of graphs are: graphs with bandwidth $\leq k$, graphs with cutwidth $\leq k$, the series-parallel graphs, the outerplanar graphs, the k-outerplanar graphs, Halin graphs, chordal graphs with maximum clique size k, graphs with genus $\leq d$ and disk dimension $\leq k$(d, k constants). For an overview of several results of this type, see [6]. The class of partial k-trees equals the class of graphs with treewidth $\leq k$.

An interesting open problem is whether there exists a parallel variant of Robertson and Seymours algorithm [13], that finds a branch-decomposition (and hence also a tree-decomposition) of a graph with constant branchwidth or treewidth, that has again constant, but not necessarily optimal branchwidth or treewidth. Their algorithm uses $\mathcal{O}(n^2)$ time. A parallel variant of this algorithm could be used to determine whether a graph has treewidth $\leq k$, and as first step for the algorithms in section 4, using perhaps a smaller number of processors as the algorithm of section 3.

References

[1] S. Arnborg. Efficient algorithms for combinatorial problems on graphs with bounded decomposability – A survey. *BIT*, 25:2–23, 1985.

[2] S. Arnborg, D. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–284, 1987.

[3] S. Arnborg, J. Lagergren, and D. Seese. Which problems are easy for tree-decomposable graphs. 1987. Ext. abstract to appear in proc. ICALP 88.

[4] S. Arnborg and A. Proskurowski. *Linear time algorithms for NP-hard problems on graphs embedded in k-trees.* TRITA-NA-8404, Dept. of Num. Anal. and Comp. Sci., Royal Institute of Technology, Stockholm, Sweden, 1984.

[5] H. L. Bodlaender. 1988. Unpublished result.

[6] H. L. Bodlaender. *Classes of Graphs with Bounded Treewidth.* Technical Report RUU-CS-86-22, Dept. Of Comp. Science, University of Utrecht, Utrecht, 1986.

[7] H. L. Bodlaender. *Dynamic programming algorithms on graphs with bounded tree-width.* Tech. Rep., Lab. for Comp. Science, M.I.T., 1987. Ext. abstract to appear in proc. ICALP 88.

[8] H. L. Bodlaender. *Polynomial algorithms for Chromatic Index and Graph Isomorphism on partial k-trees.* Tech. Rep. RUU-CS-87-17, Dept. of Comp. Science, University of Utrecht, 1987.

[9] N. Chandrasekharan and S. S. Iyengar. *NC Algorithms for Recognizing Chordal Graphs and k-Trees.* Tech. Rep. 86-020, Dept. of Comp. Science, Louisiana State University, 1986.

[10] J. Engelfriet, G. Leih, and E. Welzl. Characterization and complexity of boundary graph languages. 1987. Manuscript.

[11] G. Miller and J. Reif. Parallel tree contraction and its application. In *Proc. of the 26th Annual IEEE Symp. on the Foundations of Comp. Science*, pages 478–489, 1985.

[12] N. Robertson and P. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. of Algorithms*, 7:309–322, 1986.

[13] N. Robertson and P. Seymour. Graph minors. XIII. The disjoint paths problem. 1986. Manuscript.

[14] P. Scheffler and D. Seese. A combinatorial and logical approach to linear-time computability. 1986. Extended abstract.