NC-ALGORITHMS
FOR GRAPHS WITH SMALL TREEWIDTH

Hans L. Bodlaender
Department of Computer Science, University of Utrecht
P.O.Box 80.012, 3508 TA Utrecht, the Netherlands

Abstract

In this paper we give a parallel algorithm for recognizing graphs with treewidth \( \leq k \), for constant \( k \), and building the corresponding tree-decomposition, that uses \( O(\log n) \) time and \( O(n^{3k+4}) \) processors on a CRCW PRAM. Also, we give a parallel algorithm that transforms a given tree-decomposition of a graph \( G \) with treewidth \( k \) to another tree-decomposition of \( G \) with treewidth \( \leq 3k + 2 \), such that the tree in this tree-decomposition is binary and has logarithmic depth. The algorithm uses a linear number of processors and \( O(\log n) \) time. Many NP-complete graph problems are known to be solvable in polynomial time, when restricted to graphs with treewidth \( \leq k \), \( k \) constant. From the results in this paper, it follows that most of these problems are also in NC, when restricted to graphs with treewidth bounded by a constant.

1 Introduction

The class of graphs with treewidth \( \leq k \) has the property that many graph problems, which are NP-complete for arbitrary graphs, become solvable in polynomial time, when restricted to this class \([4,7,6,9,10,16,17]\). Arnborg, Corneil and Proskurowski gave an \( O(n^{k+2}) \) algorithm to recognize graphs with treewidth \( \leq k \), and find the corresponding tree-decompositions [2]. Deep results from Robertson and Seymour on graph minors show that there exist \( O(n^2) \) algorithms to recognize graphs with treewidth \( \leq k \) [16]. Recently, we were able to use this result to show the existence of \( O(n^2) \) algorithms that find the corresponding tree-decompositions. The non-constructive elements in the result of Robertson and Seymour can often, and also in this case, be avoided with a technique of Fellows and Langston [11].

In this paper we consider the parallel complexity of the problems. To be precise: we show that the problems are in NC, i.e. they can be solved on a CRCW PRAM, using a polynomial number of processors and polylogarithmic time. Chandrasekharan and Sitharama Iyengar [8] considered the related problem of recognizing \( k \)-trees, and showed that this can be done in \( O(\log n) \) time on a CRCW PRAM with \( O(n^4) \) processors. A related result on graphs with bounded treewidth and bounded degree was obtained by Engelfriet, Leih and Welzl [10]. A special case of these problems is considered in [10].

This paper is organized as follows. In section 2 a number of fundamental definitions are given and some basic graph theoretic results are derived. In section 3 we show that recognizing graphs with treewidth \( \leq k \) and finding the corresponding tree-decompositions is in NC, for constant \( k \). In section 4 we give a parallel algorithm that transforms a given tree-decomposition of a graph \( G \) with treewidth \( k \) to another tree-decomposition of \( G \) with treewidth \( \leq 3k + 2 \), such that the tree in this tree-decomposition is binary and has logarithmic depth. From this result, it follows that many sequential polynomial time algorithms for graphs with bounded treewidth can be transformed to NC-algorithms. All problems considered in [3] and [6] can be dealt with in this way.

2 Definitions and graph-theoretic results

First we give the definition of the treewidth of a graph, introduced by Robertson and Seymour [15]. Some alternative definitions of the same class of graphs can be found in [1].

Definition.

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

- Every vertex of \( G \) is part of some set \( X_i \).
- If \( \{u, v\} \in E \), then \( \exists i \in I \) with \( u, v \in X_i \).
- For every vertex \( v \in V \), the set \( \{i \in I \mid v \in X_i\} \) is connected.
- \( T \) is a tree.
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 $\{(X_i \mid i \in I), T\}$ is $\max_{i \in I} |X_i| - 1$. The treewidth of $G$, denoted $\text{treewidth}(G)$ 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 $(S, \{(v, w) \mid v, w \in S, v \neq w\})$. 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 $W : G[W] = (W, \{(v, w) \mid v, w \in W \text{ and } (v, w) \in E\})$.

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

**Lemma 2.1**
Let $\{(X_i \mid i \in I), T = (I, F)\}$ be a tree-decomposition of $G = (V, E)$. Suppose $W \subseteq V$ forms a clique in $G$. Then $\exists i : W \subseteq X_i$.

**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 $\{i \in I \mid W' \subseteq X_i\}$. 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 $i'$ 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 $\{(X_i \mid i \in I), T = (I, F)\}$ of a graph $G = (V, E)$ is called full, iff

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

(ii) $\forall (i, j) \in F : X_i \notin X_j$ and $X_j \notin X_i$.

**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 $\{(X_i \mid i \in I - \{i_1\}), T' = (I - \{i_1\}, \{(i, j) \mid (i, j) \in I - \{i_1\} \text{ and } (i, j) \in F\}) \text{ or } (i = i_0 \text{ and } (i_1, j) \in F) \text{ or } (j = i_1 \text{ and } (i, i) \in F)\}$. 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 $|X_{i_0}| \leq k$, then either operation 1 can be applied, or there is an adjacent node $i_1 \in I$ with $\exists v \in X_{i_0} : v \in X_{i_1}$. Make a new tree-decomposition by adding $v$ to $X_{i_0} : \{(X_i \mid i \in I), T = (I, F)\}$, with $X'_{i_0} = X_{i_0} \cup \{v\}$, and $X'_{i_1} = X_{i_1} \text{ for } i \neq i_0$. This is again a tree-decomposition of $G$ with treewidth $\leq k$. In this case the size of the index set $I$ does not change, but $\sum_{i \in I} |X_i|$ is increased by one.