SIAM J. COMPUT.
Vol. 25, No. 6, pp. 1305–1317, December 1996

010

A LINEAR-TIME ALGORITHM FOR FINDING TREE-DECOMPOSITIONS OF SMALL TREEWIDTH*

HANS L. BODLAENDER†

Abstract. In this paper, we give for constant k a linear-time algorithm that, given a graph $G = (V, E)$, determines whether the treewidth of G is at most k and, if so, finds a tree-decomposition of G with treewidth at most k. A consequence is that every minor-closed class of graphs that does not contain all planar graphs has a linear-time recognition algorithm. Another consequence is that a similar result holds when we look instead for *path*-decompositions with pathwidth at most some constant k.

Key words. graph algorithms, treewidth, pathwidth, partial k-trees, graph minors

AMS subject classifications. 68R10, 05C85, 05C05

1. Introduction.

1.1. Background. The notions of "tree-decomposition" and "treewidth" have received much attention recently, not in the least due to the important role they play in the deep results on graph minors by Robertson and Seymour (see, e.g., [27, 28, 29, 30, 31] and many other papers in this series). (See also [21].) Also, many graph problems, including a very large number of well-known NP-hard problems, have been shown to be linear-time solvable on graphs that are given together with a tree-decomposition of treewidth at most k for constant k. (See, among other sources, [2, 5, 6, 7, 8, 12, 14, 15, 16, 33, 35].)

The first step of algorithms that exploit the small treewidth of input graphs is to find a tree-decomposition with treewidth bounded by a constant—although possibly not optimal. Thus far, this step has dominated the running time of most algorithms since the second step (some kind of "dynamic-programming" algorithm using the tree-decomposition) usually costs only linear time. The best algorithm known so far for this "first step" was an algorithm by Reed [26], which costs $O(n \log n)$. In this paper, we improve on this result and give a linear-time algorithm.

The problem "Given a graph $G = (V, E)$ and an integer k, is the treewidth of G at most k?" is NP-complete [3]. Much work has been done on this problem for constant k. For $k = 1, 2, 3$, linear-time algorithms exist [25]. Recently, Sanders [32] established a complex linear-time algorithm for the case where $k = 4$. Arnborg et al. [3] showed that the problem is solvable in $O(n^{k+2})$ time for constant k. Then Robertson and Seymour gave a nonconstructive proof of the existence of $O(n^2)$ decision algorithms [31]. Actually, this algorithm is of a "two-step" form, as described above. The first step is to apply an $O(n^2)$ algorithm that either outputs that the treewidth of G is larger than k or outputs a tree-decomposition with width at most $4k$. (Actually, the result is stated in [31] in terms of "branchwidth," but this is an unimportant technical difference.) The second step uses the notion of graph minors. A graph G is a minor of a graph H if G can be obtained from H by a series of vertex deletions, edge deletions, and edge contractions. Robertson and Seymour have shown that every class of graphs $\mathcal{G}$ that is closed under the taking of minors has a finite set of graphs, called

* Received by the editors June 28, 1993; accepted for publication (in revised form) March 15, 1995. This research was partially supported by the EC ESPRIT Basic Research Actions contract 7141 (project ALCOM II).

† Department of Computer Science, Utrecht University, P. O. Box 80.089, 3508 TB Utrecht, the Netherlands (hansb@cs.ruu.nl).

the obstruction set, with the property that a graph belongs to $\mathcal{G}$ if and only if it has no graph from the obstruction set as a minor. Since the class of graphs with treewidth at most k is closed under minors for every fixed value k, a finite characterization in terms of forbidden minors exists for this class. Hence the second step of the algorithm checks whether this characterization holds for the input graph. This step can be done in linear time using dynamic-programming techniques as used, e.g., in [5, 6, 14]. In [9] (using results from [20]), it was shown that the nonconstructive elements can be avoided using self-reduction without increasing the running time by more than a (huge) constant factor.

Both Lagergren [23] and Reed [26] improved on the "first step." Lagergren gave a sequential algorithm that uses $O(n \log^2 n)$ time and a parallel algorithm that uses $O(n)$ processors and $O(\log^3 n)$ time. Reed gave a sequential $O(n \log n)$ algorithm that has a parallel implementation with $O(n/\log n)$ processors and $O(\log^2 n)$ time. A related probabilistic result (with running time $O(n \log^2 n + n|\log p|)$, where p is the probability of error) was found by Matoušek and Thomas [25]. Each of these algorithms either determines that the input graph G has treewidth greater than k or finds a tree-decomposition of G with treewidth bounded by some constant (linear in k). They all are based upon finding "balanced separators" in some clever way. Our algorithm uses a different approach: we reduce the problem in linear time to a problem on a smaller graph by edge contraction or by removing "simplicial vertices."

Independently, Lagergren and Arnborg [24] and Bodlaender and Kloks [12, 22] showed that the "second step" can be done without the use of graph minors and gave explicit algorithms to test in linear time whether G has treewidth at most k once a tree-decomposition of G with bounded treewidth is available. Moreover, from these results, it follows that a technique of Fellows and Langston [19] can be used to compute the obstruction set of the class of graphs with treewidth $\leq k$. Bodlaender and Kloks also showed how, if it exists, a tree-decomposition with width at most k can be computed in the same time bounds. Results of a similar flavor were obtained independently by Abrahamson and Fellows [1].

Recognition algorithms for graphs with treewidth $\leq k$ (k constant) have been designed by Arnborg et al. [4]. These algorithms use linear time but polynomial—not linear—memory. (It is allowed that the algorithm consults the contents of memory that is never written to.) A disadvantage of this approach is that it is not known how to *construct* tree-decompositions with small treewidth by the method.

1.2. Main idea of algorithm. The main result in this paper is the following.

THEOREM 1.1. *For all $k \in N$, there exists a linear-time algorithm that tests whether a given graph $G = (V, E)$ has treewidth at most k and, if so, outputs a tree-decomposition of G with treewidth at most k.*

We now give an outline of how this result is obtained.

We begin by introducing some notation. For a value d to be fixed later, we define *low-degree* vertices as vertices of degree at most d and *high-degree* vertices as vertices of degree greater than d. A vertex is *friendly* if it is a low-degree vertex and adjacent to another low-degree vertex. A vertex is *simplicial* if its neighbors form a clique. The *improved* graph of a graph G is obtained by adding edges between all vertices that have at least $k+1$ common neighbors of degree at most k. A vertex is *I-simplicial* in a graph G if it is simplicial in the improved graph of G and has degree at most k in G.

The algorithm distinguishes between two cases:

1. *There are "many" friendly vertices.* As shown in §3, any maximal matching

in G contains in this case "sufficiently many" ($\Omega(n)$) edges. We compute the graph G' obtained by contracting all edges in a maximal matching. Recursively, we compute a tree-decomposition of treewidth at most k of G' or conclude that the treewidth of G' and hence the treewidth of G is larger than k. From this tree-decomposition, we can easily build a tree-decomposition of G with treewidth at most $2k+1$. This latter tree-decomposition is used to solve the problem using the algorithm of Bodlaender and Kloks [12, 22]: using the tree-decomposition of G with treewidth at most $2k+1$, it decides whether the treewidth of G is at most k and, if so, finds a tree-decomposition of G with treewidth at most k.

2. *G has "only few" friendly vertices.* In this case, the algorithm starts by computing the improved graph of G. In §4, it is shown that this improved graph has treewidth at most k if and only if G has treewidth at most k. Also, in §4, it is shown that in this case, the improved graph of G has "sufficiently many" ($\Omega(n)$) vertices that are I-simplicial (unless the treewidth of G is more than k). Recursively, a tree-decomposition with treewidth at most k is computed of the graph G' obtained by removing all I-simplicial vertices from the improved graph of G, or we conclude that the treewidth of G' and hence of G is larger than k. Given such a tree-decomposition of G', a tree-decomposition of G with treewidth at most k is computed as follows: since the neighbors of an I-simplicial vertex v form a clique in G', a well-known lemma tells us that there is one node i in the tree-decomposition of G' with X_i containing all neighbors of v. Then we add a new node to the tree-decomposition, adjacent to i, containing v and its neighbors. In this way, we obtain a tree-decomposition of G with treewidth at most k.

In each case, the amount of work of the nonrecursive steps is linear, and each G' has size at most a constant fraction of the size of G. It follows that the algorithm uses linear time.

The basic algorithm will be given in §5. Some implementation details will be discussed in §6. Finally, some consequences of the result will be discussed in §7.

2. Definitions and preliminary results. The notion of treewidth was introduced by Robertson and Seymour [27].

DEFINITION. *A* tree-decomposition of a graph $G = (V, E)$ *is a pair* (X, T), *where* $T = (I, F)$ *is a tree and* $X = \{X_i \mid i \in I\}$ *is a family of subsets of* V, *one for each node of* T, *such that*

(i) $\bigcup_{i \in I} X_i = V$,

(ii) *for all edges* $(v, w) \in E$, *there exists an* $i \in I$ *with* $v \in X_i$ *and* $w \in X_i$, *and*

(iii) *for all* $i, j, k \in I$, *if* j *is on the path from* i *to* k *in* T, *then* $X_i \cap X_k \subseteq X_j$.

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

There are several equivalent notations, e.g., a graph is a partial k-tree if and only if its treewidth is at most k [34].

LEMMA 2.1. (See, e.g., [13].) *Suppose* $(\{X_i \mid i \in I\}, T = (I, F))$ *is a tree-decomposition of* $G = (V, E)$.

(i) *If* $W \subseteq V$ *forms a clique in* G, *then there exists an* $i \in I$ *with* $W \subseteq X_i$.

(ii) *If each vertex in* $W_1 \subseteq V$ *is adjacent to each vertex in* $W_2 \subseteq V$, *then there exists an* $i \in I$ *with* $W_1 \subseteq X_i$ *or* $W_2 \subseteq X_i$.

The *contraction* operation removes two adjacent vertices v and w and replaces them with one new vertex that is made adjacent to all vertices that were adjacent to v and w.

We say that a tree-decomposition (X, T) of treewidth k is *smooth* if for all $i \in I, |X_i| = k+1$ and for all $(i,j) \in F, |X_i \cap X_j| = k$. Any tree-decomposition of a graph G can be transformed to a smooth tree-decomposition of G with the same treewidth. Apply the following operations until none is possible:

(i) If for $(i,j) \in F$, $X_i \subseteq X_j$, then contract the edge (i,j) in T and take as the new node $X_{j'} = X_j$.

(ii) If for $(i,j) \in F$, $X_i \not\subseteq X_j$ and $|X_j| < k+1$, then choose a vertex $v \in X_i - X_j$ and add v to X_j.

(iii) If for $(i,j) \in F$, $|X_i| = |X_j| = k+1$ and $|X_i - X_j| > 1$, then subdivide the edge (i,j) in T; let i' be the new node; choose a vertex $v \in X_i - X_j$ and a vertex $w \in X_j - X_i$, and let $X_{i'} = X_i - \{v\} \cup \{w\}$.

LEMMA 2.2. *If (X,T) is a smooth tree-decomposition of $G = (V,E)$ with treewidth k, then $|I| = |V| - k$.*

Proof. The proof is by induction on $|I|$. If $|I| = 1$, then clearly $|V| = k+1$. Suppose that the lemma holds for $|I| = r-1$. Consider a smooth tree-decomposition (X,T) of a graph $G = (V,E)$ with treewidth k and $|I| = r$. Let i be a leaf of T. There is a unique vertex v that belongs to X_i but not to any set X_j, $j \in I - \{i\}$. If we remove node i from T, we get a smooth tree-decomposition of $G[V - \{v\}]$ with treewidth k and with $|I| - 1$ nodes. The result now follows by induction. □

The following well-known lemma can be easily proved by induction on the number of vertices, removing vertices as in Lemma 2.2.

LEMMA 2.3. *If the treewidth of $G = (V,E)$ is at most k, then $|E| \leq k|V| - \frac{1}{2}k(k+1)$.*

The set of neighbors of a vertex v in $G = (V,E)$ is denoted by $N_G(v) = \{w \in V \mid (v,w) \in E\}$.

THEOREM 2.4. (See Bodlaender and Kloks [12, 22].) *For all k and l, there exists a linear-time algorithm that, when given a graph $G = (V,E)$ together with a tree-decomposition (X,T) of G with treewidth at most l, determines whether the treewidth of G is at most k and, if so, finds a tree-decomposition of G with treewidth at most k.*

Analysis of this algorithm shows that its constant factor is at most $l^{l-2} \cdot ((2l+3)^{2l+3} \cdot (\frac{8}{3} \cdot 2^{2k+2})^{2l+3})^{2l-1}$, i.e., when $l = O(k)$, exponential in k^3. The analysis leading to this constant is rather crude, however, and a precise analysis should give a much better and smaller estimate.

3. Friendly, high-degree, and low-degree vertices. In this section, we introduce the concepts of the "friendly," "high-degree," and "low-degree" vertex. We show that a graph with treewidth at most k has "few" high-degree vertices, and when it has "many" friendly vertices, then it has a "large" maximal matching.

In the remainder, we assume that k is a given fixed constant. Let

$$c_1 = \frac{1}{k^2 \cdot (k+1) \cdot (4k^2 + 12k + 16)}$$

and

$$c_2 = \frac{1}{8k^2 + 24k + 32}.$$

Note that

$$c_2 = \frac{1}{4k^2 + 12k + 16} - \frac{c_1 \cdot k^2 \cdot (k+1)}{2}.$$

Let $d = 2k^3 \cdot (k+1) \cdot (4k^2 + 12k + 16)$. Note that $d = 2k/c_1$. We say that a vertex with degree at most d is a *low-degree vertex* and a vertex with degree larger than d is a *high-degree vertex*. A vertex is said to be *friendly* if it is a low-degree vertex and is adjacent to at least one other low-degree vertex.

We show below that c_1 is an upper bound on the fraction of high-degree vertices. In this section and the next, we show that c_2 is a lower bound on the fraction of vertices that can be removed in one of the two cases, as mentioned in §1.2.

LEMMA 3.1. *There are fewer than $c_1 \cdot |V|$ high-degree vertices in a graph with treewidth k.*

Proof. If there are n_l high-degree vertices, then G must contain at least $n_l \cdot d/2$ edges. By Lemma 2.3, $n_l \cdot d/2 < k|V|$. □

A *maximal matching* of a graph $G = (V, E)$ is a set of edges $M \subseteq E$ such that no two edges in M share an endpoint and every $e \in E - M$ shares an endpoint with an edge in M. We can easily find a maximal matching in $O(|V|+|E|)$ time with a greedy algorithm. Note that by Lemma 2.3, $O(|V| + |E|) = O(|V|)$ for graphs $G = (V, E)$ with their treewidth bounded by a constant.

LEMMA 3.2. *If there are n_f friendly vertices in $G = (V, E)$, then any maximal matching of G contains at least $n_f/(2d)$ edges.*

Proof. Consider a maximal matching M. Any friendly vertex must be endpoint of an edge in M or adjacent to a friendly vertex that is an endpoint of an edge in M. With each edge e of M, we associate the at most $2d$ friendly vertices that are endpoints of e or adjacent to friendly (and hence low-degree) endpoints of e. If a friendly vertex has not been associated with at least one edge in M, then M is not maximal. Hence $|M| \geq n_f/(2d)$. □

Let M be a maximal matching in $G = (V, E)$, and let $G' = (V', E')$ be the graph obtained by contracting all edges in M. Define $f_M : V \to V'$ by $f_M(v) = v$ if v is not an endpoint of an edge in M, and let $f_M(v) = f_M(w)$ be the vertex that the contraction of the edge $(v, w) \in M$ results in.

LEMMA 3.3. *Let M, G, G', and f_M be as above. If (X, T) is a tree-decomposition of G' with treewidth k, then (Y, T) defined by $Y_i = \{v \in V \mid f_M(v) \in X_i\}$ is a tree-decomposition of G with treewidth at most $2k + 1$.*

Proof. This easily follows from the definitions. □

LEMMA 3.4. (See, e.g., [27].) *If G' is a minor of G, then the treewidth of G' is at most the treewidth of G.*

4. Simplicial vertices. In this section, we introduce the improved graph of a graph G. We show that a graph G has treewidth at most k if and only if its improved graph has treewidth at most k. The main result of this section is Theorem 4.3, which states that every graph of treewidth at most k contains "many" friendly vertices or "many" I-simplicial vertices.

For a graph $G = (V, E)$, let the *improved* graph $G' = (V, E')$ of G be the graph obtained by adding an edge (v, w) to E for all pairs $v, w \in V$ such that v and w have at least $k + 1$ common neighbors of degree at most k in G.

LEMMA 4.1. *If the treewidth of G is at most k, then the treewidth of the improved graph of G is at most k. Moreover, any tree-decomposition of G with treewidth at most k is also a tree-decomposition of the improved graph with treewidth at most k, and vice versa.*

Proof. Suppose that (X, T) is a tree-decomposition of $G = (V, E)$ with treewidth at most k. Consider vertices v and w with at least $k + 1$ common neighbors. By Lemma 2.1(ii), there exists either an $i \in I$ with $v, w \in X_i$ or an $i \in I$ with X_i

containing the set W of all common neighbors of v and w. In the latter case, (X, T) is also a tree-decomposition of the graph G'' obtained from G by adding edges between all vertices in W. However, G'' contains a clique with at least $k+2$ vertices (namely, $W \cup \{v\}$) and has treewidth at most k. This contradicts Lemma 2.1(i).

Therefore, for all v and w that have $k+1$ common neighbors, there exists an $i \in I$ with $v, w \in X_i$. Hence (X, T) is also a tree-decomposition of the improved graph of G. The lemma now follows directly. □

We say that a vertex v is *simplicial in* G if its neighbors form a clique in G. We say that v is I-simplicial if it is simplicial in the improved graph of G and is of degree at most k in G.

We now derive via a series of lemmas the following result, which states that if we have "few" friendly vertices and the treewidth of G is at most k, then we have "many" I-simplicial vertices.

THEOREM 4.2. *For every graph $G = (V, E)$ with treewidth at most k, at least one of the following properties holds:*

(i) *G contains at least $|V|/(4k^2 + 12k + 16)$ friendly vertices.*

(ii) *The improved graph of G contains at least $c_2|V|$ I-simplicial vertices.*

Proof. The proof of Theorem 4.2 will be given with help of several lemmas.

A vertex $v \in V$ is said to be *T-simplicial* with respect to some tree-decomposition (X, T) if it is not friendly and there exists a node $i \in I$ such that all neighbors of v belong to X_i. A T-simplicial vertex has degree at most k since all of its neighbors belong to a set X_i, $|X_i| \leq k+1$.

LEMMA 4.3. *For all smooth tree-decompositions (X, T) of $G = (V, E)$ with treewidth k, the following conditions hold:*

(i) *We can associate with every leaf i of T a low-degree vertex $v \in X_i$ that is friendly or T-simplicial with respect to (X, T), and there does not exist a $j \in I$, $j \neq i$, with $v \in X_j$.*

(ii) *We can associate with every path $i_0, i_1, \ldots, i_{k^2+3k+3}$ in T with $i_1, \ldots, i_{k^2+3k+2}$ nodes of degree 2 in T at least one vertex $v \in X_{i_1} \cup \cdots \cup X_{i_{k^2+3k+2}}$ that is friendly or T-simplicial with respect to (X, T) such that v does not belong to a set X_j with $j \in I$ a node not on this path.*

Proof. (i) Let j be the neighbor of leaf i in T. Let v be the unique vertex in $X_i - X_j$. v is adjacent to only vertices in X_i. Either all neighbors of v are of high degree, in which case v is T-simplicial with respect to (X, T), or a neighbor of v is of low degree, in which case v is friendly.

(ii) Note that $|X_{i_0} \cup \cdots \cup X_{i_{k^2+3k+3}}| = k^2 + 4k + 4 \leq d$. Hence all vertices in $X_{i_1} \cup \cdots \cup X_{i_{k^2+3k+2}} - (X_{i_0} \cup X_{i_{k^2+3k+3}})$ are of low degree. Suppose that neither of them is friendly, i.e., they are adjacent to only high-degree vertices in $X_{i_0} \cup X_{i_{k^2+3k+3}}$. Suppose that X_{i_0} contains r high degree vertices, say $w_1, \ldots, w_r$. Clearly, $r \leq |X_{i_0}| = k+1$. For each s, where $1 \leq s \leq r$, assume that w_s belongs to successive sets $X_{i_0}, X_{i_1}, \ldots, X_{i_{w_s}}$. Suppose w.l.o.g. $i_{w_1} \leq i_{w_2} \leq \cdots \leq i_{w_r}$. If some low-degree vertex v belongs to exactly one set X_{i_j}, $1 \leq j \leq k^2 + 3k + 2$, then it must be T-simplicial with respect to (X, T). If some low-degree vertex v belongs only to (a subset of) sets $X_{i_{w_j+1}}, \ldots, X_{i_{w_{j+1}}}$, then all neighbors of v belong to $X_{i_{w_{j+1}}}$; hence v is T-simplicial with respect to (X, T). All vertices in $X_{i_1} \cup \cdots \cup X_{i_{k^2+3k+2}}$ that are not of one of these two types must belong to at least one of the sets $X_{i_0}, X_{i_{w_1}}, \ldots, X_{i_{w_r}}, X_{i_{k^2+3k+3}}$. These are, in total, at most $(k+1)(k+3) = k^2 + 4k + 3$ vertices. Therefore, at least one vertex in $X_{i_1} \cup \cdots \cup X_{i_{k^2+3k+2}} - X_{i_0} - X_{i_{k^2+3k+3}}$ must be T-simplicial with respect to (X, T). □

A leaf-path collection of a tree T is a collection of leaves in T plus a collection of paths of length k^2+3k+4 in T where all nodes on a path that are not endpoints of a path have degree 2 in T and do not belong to any other path in the collection. The size of the collection is the total number of leaves plus the total number of paths in the collection.

LEMMA 4.4. *Each tree with r nodes contains a leaf-path collection of size at least $r/(2k^2+6k+8)$.*

proof Let r_b be the number of nodes of degree at least 3, r_l be the number of leaves, and r_2 be the number of nodes of degree 2. Clearly, $r_b < r_l$. All nodes of degree 2 belong to $< r_l+r_b$ connected components of the forest, obtained by removing all leaves and all nodes with degree 3 or larger from the tree. Each such component contains at most k^2+3k+3 nodes that are not part of a leaf-path collection of maximum size. Therefore, there are fewer than $(r_b+r_l)(k^2+3k+3)$ nodes of degree 2 that are not on a path in the collection. Hence there are at least

$$\frac{r_2-(r_b+r_l)(k^2+3k+3)}{k^2+3k+4}$$

paths in a leaf-path collection of maximum size. It follows that the maximum size of a leaf-path collection is at least

$$\max\left(r_l, \frac{r_2-(r_b+r_l)(k^2+3k+3)}{k^2+3k+4}+r_l\right) \geq \frac{1}{2}\cdot\frac{r}{k^2+3k+4}. \qquad \square$$

COROLLARY 4.5. *If (X,T) is a smooth tree-decomposition of $G=(V,T)$ with treewidth k, then G contains at least $|V|/(2k^2+6k+8)-1$ vertices that are friendly or T-simplicial with respect to (X,T).*

Proof. T contains $|V|-k$ nodes (Lemma 2.2). Now apply Lemmas 4.3 and 4.4. $\square$

A set $Y \subseteq V$ of high-degree vertices is said to be *semiimportant* with respect to the tree-decomposition (X,T) of $G=(V,E)$ if there exists an $i \in I$ with $Y \subseteq X_i$. A set Y is said to be *important* if it is semiimportant with respect to (X,T) and not contained in any larger semiimportant set with respect to (X,T).

LEMMA 4.6. *Let (X,T) be a tree-decomposition of $G=(V,E)$ with treewidth k. The number of different important sets with respect to (X,T) is at most the number of high-degree vertices in G.*

Proof. Let L be the set of high-degree vertices in G. $(\{X_i \cap L \mid i \in I\}, T)$ is a tree-decomposition of $G[L]$. Each important set Y is a set $X_i \cap L$ that is not contained in another set $X_{i'} \cap L$. Repeatedly contract edges (i,i') in T with $X_i \cap L \supseteq X_{i'} \cap L$ with the newly formed node containing all vertices in X_i. The resulting tree-decomposition of $G[L]$ contains the same maximal sets X_i and will have at most $|L|$ nodes. $\square$

A function f that maps each T-simplicial (with respect to some tree-decomposition (X,T)) vertex v to an important (with respect to (X,T)) set Y with $N_G(v) \subseteq Y$ is called a *T-simplicial-to-important function* for (X,T). By definition, a T-simplicial-to-important function always exists.

LEMMA 4.7. *Let f be a T-simplicial-to-important function for a smooth tree-decomposition (X,T) of $G=(V,E)$ with treewidth k. Let Y be an important set with respect to (X,T). Then at most $\frac{1}{2}k^2(k+1)$ T-simplicial vertices with respect to (X,T) (and G) in $f^{-1}(Y)$ are not I-simplicial.*

Proof. Assign each non-I-simplicial T-simplicial vertex v to a pair of neighbors of v, that are nonadjacent in the improved graph. To each pair of vertices, there

cannot be assigned more than k vertices since otherwise they would have at least $k+1$ common neighbors of degree at most k and there would be an edge between them in the improved graph.

It follows that the number of non-I-simplicial T-simplicial vertices v with $f(v) = Y$ is at most $\frac{1}{2}|Y| \cdot (|Y| - 1) \leq \frac{1}{2}k^2(k+1)$. □

We can now prove Theorem 4.2. Suppose that G contains fewer than $|V|/(4k^2 + 12k + 16)$ friendly vertices and that the treewidth of G is at most k. By Lemma 3.1, there are at most $c_1 \cdot |V|$ high-degree vertices in G, and hence, by Lemma 4.6, the number of important sets with respect to an arbitrary smooth tree-decomposition (X, T) of G with treewidth $\leq k$ is at most $c_1 \cdot |V|$. Using both the fact that a T-simplicial-to-important function always exists and Lemma 4.7, it follows that at most $\frac{1}{2}k^2(k+1) \cdot (c_1 \cdot |V| - 1)$ T-simplicial vertices with respect to (X, T) are not I-simplicial. Using Corollary 4.5, it follows that $|V|/(2k^2 + 6k + 8) - 1 - |V|/(4k^2 + 12k + 16) - \frac{1}{2}k^2(k+1) \cdot (c_1 \cdot |V| - 1) \geq c_2 \cdot |V|$ vertices are I-simplicial. This completes the proof of Theorem 4.2. □

LEMMA 4.8. *Let (X, T) be a tree-decomposition of treewidth at most k of the graph G' obtained by removing all I-simplicial vertices (and adjacent edges) from the improved graph of graph $G = (V, E)$. Then for all I-simplicial vertices v, there exists an $i \in I$ with $N_G(v) \subseteq X_i$.*

Proof. Note that, by definition, I-simplicial vertices are nonadjacent in G, and their neighborhood forms a clique in the improved graph of G. The result now follows directly from Lemma 2.1(i). □

5. Main algorithm. We now give a recursive description of our main algorithm. Some details will be discussed in §6. Our algorithm, when given a graph $G = (V, E)$, either

(i) outputs that the treewidth of G is larger than k or

(ii) outputs a tree-decomposition of G with treewidth at most k.

For "very small graphs" (i.e., with at most some constant number of vertices), any other finite algorithm is used to solve the problem. Otherwise, the following algorithm is used:

First, check whether $|E| \leq k \cdot |V| - \frac{1}{2}k(k+1)$. If this is not the case, we know by Lemma 2.3 that the treewidth of G is larger than k: **stop**.

Now count the number of friendly vertices. If there are at least $|V|/(4k^2+12k+16)$ friendly vertices, do the following:

(i) Find a maximal matching $M \subseteq E$ in G.

(ii) Compute the graph $G' = (V', E')$ obtained by contracting every edge in M.

(iii) Recursively apply the algorithm to G'.

(iv) If G' has treewidth larger than k, **stop**. The treewidth of G is also larger than k. (See Lemma 3.4.)

(v) Suppose that the recursive call yielded a tree-decomposition (X, T) of G' with treewidth k. Construct a tree-decomposition (Y, T) of G with treewidth at most $2k+1$, as in Lemma 3.3.

(vi) Use the algorithm of Theorem 2.4 to compute whether the treewidth of G is at most k and, if so, compute a tree-decomposition of G of treewidth at most k.

If there are fewer than $|V|/(4k^2 + 12k + 16)$ friendly vertices, do the following:

(i) Compute the improved graph of G. (See §6.)

(ii) If there exists an I-simplicial vertex with degree at least $k+1$, then **stop**: the improved graph of G contains a clique with $k+2$ vertices; hence the treewidth of G is more than k.

(iii) Put all I-simplicial vertices in some set SL. Compute the graph G' obtained by removing all I-simplicial vertices and adjacent edges from G.

(iv) If $|SL| < c_2|V|$, then **stop**: the treewidth of G is larger than k. (See Theorem 4.2.)

(v) (Now $|SL| \geq c_2|V|$.) Recursively apply the algorithm on G'.

(vi) If the treewidth of G' is larger than k, then **stop**: since G' is a subgraph of G, we also have that the treewidth of G is larger than k.

(vii) Suppose that the recursive call yielded a tree-decomposition (X, T) of G' with treewidth k. For all $v \in SL$, find an $i_v \in V$ with $N_G(v) \subseteq X_{i_v}$, add a new node j_v to T with $X_{j_v} = \{v\} \cup N_G(v)$, and make j_v adjacent to i_v in T. (Such a node i_v exists by Lemma 4.8.) The result is a tree-decomposition of G with treewidth at most k.

The correctness of the algorithm follows from results given in §§2 and 4.

The running time of the algorithm can be estimated as follows. We recursively apply the algorithm on either a graph with $(1 - 1/(2d(4k^2 + 12k + 16))) \cdot |V|$ vertices (Lemma 3.2) or a graph with $(1 - c_2)|V|$ vertices. Write

$$c_3 = \frac{1}{8k^6 + 32k^4 + 56k^3 + 32k^3} = \max\left(1 - c_2,\ 1 - \frac{1}{2d \cdot (4k^2 + 12k + 16)}\right).$$

Since all nonrecursive steps have a linear-time implementation (see also §6), we have that if the algorithm takes $T(n)$ time on a graph with n vertices in the worst case, then $T(n) \leq T(c_3 \cdot n) + O(n)$; hence $T(n) = O(n)$. It also follows that the algorithm uses linear memory.

6. Some details of the algorithm. In this section, we show that the steps of the algorithm given in §5 can be implemented in linear time and linear memory. Most steps are either rather straightforward and thus left to the reader or follow from earlier results. Note that we may always assume that the number of edges that we are working with is linear in the number of vertices. All graphs that we work with will be represented by their adjacency lists.

When we contract the edges in a matching M, we directly get an implicit representation of G' by a bag of edges, where some edges of G' may appear twice. By bucket sorting this bag of edges twice, we can remove all multiple copies of edges and easily obtain an adjacency list representation of G'.

Computing the improved graph and the I-simplicial vertices. Number the vertices $v_1, v_2, \ldots, v_n$. We use a queue Q that contains triples of the form $((v, w), x)$ with $v, w, x \in V$ or of the form $((v, w),—)$, $v, w \in V$. Also, we use an array S with, for each $v_i \in V$, a list $S[v_i]$ containing pairs of vertices. For all $(v_i, v_j) \in E$ with $i < j$, put $((v_i, v_j),—)$ on Q. For all vertices $v \in V$ with degree at most k, for all pairs of neighbors $v_i, v_j \in N_G(v)$ with $i < j$, put $((v_i, v_j), v)$ on Q. Now "bucket sort" Q twice, once to the first-vertex entries and once to the second-vertex entries. After this double bucket sort, all pairs of the form $((v_i, v_j), \ldots)$ for fixed v_i and v_j will be in consecutive positions in Q. By inspecting Q, we can directly see what pairs of vertices have at least $k + 1$ common neighbors of degree at most k. (If at least $k + 1$ entries $((v_i, v_j), v)$ are adjacent in Q for some pair v_i, v_j, (v_i, v_j) must be present in the improved graph.) For each such pair (v_i, v_j) and if a triple $((v_i, v_j),—)$ is in Q, add the pair (v_i, v_j) to all lists $S[v]$ for vertices v with $((v_i, v_j), v)$ in Q. This all can be done in linear time using the consecutiveness of the pairs of the form $((v_i, v_j), \ldots)$.

Checking whether a vertex v of degree at most k is I-simplicial can be done by inspecting $S[v]$. $S[v]$ will consist precisely of all edges between neighbors of v. Since v has degree at most k, $S[v]$ is of size, bounded by a constant.

Adding I-simplicial vertices back in the tree-decomposition. Suppose that we have a tree-decomposition (X, T) of $G[V - SL]$ and we want to add all I-simplicial vertices in SL. For all $l \leq k$, we take a queue Q_l, in which we place all pairs $((v_{i_1}, \ldots, v_{i_l}), i)$ for $v_{i_1}, \ldots, v_{i_l} \in X_i$, $i \in I$, $i_1 < i_2 < \cdots < i_l$, and all pairs $((v_{i_1}, \ldots, v_{i_l}), v)$ with v I-simplicial and $N_G(v) = \{v_{i_1}, \ldots, v_{i_l}, i_1 < i_2 < \cdots < i_l\}$.

For each l, $1 \leq l \leq k$, bucket sort Q_l l times, once for each of the l positions in the l-tuple. All entries of the form $((v_{i_1}, \ldots, v_{i_l}), \ldots)$ will be at successive positions in Q_l after this operation. By a simple scan of Q_l, we can find for each entry $((v_{i_1}, \ldots, v_{i_l}), v)$ an entry of the form $((v_{i_1}, \ldots, v_{i_l}), i)$ for some $i \in I$. This node i is a node that the new node j_v with $X_{j_v} = \{v\} \cup N_G(v)$ can be made adjacent to.

Analysis of the constant factor. We now analyze the constant factor of the algorithm somewhat more precisely. (In the following analysis, k is no longer considered a constant.) There are two nonrecursive steps that take time with a constant factor that is not polynomial in k: the application of the algorithm of Theorem 2.4 and the addition of I-simplicial vertices back in the tree-decomposition. One directly sees that the former constant is largest. Note that Theorem 2.4 is applied with $l = 2k + 1$. Now, since $1 - c_3 = \Theta(k^{-5})$, $T(n) \leq T(c_3 n) + (2k+1)^{2k+1-2} \cdot ((2(2k+1)+3)^{2(2k+1)+3} \cdot (\frac{8}{3} \cdot 2^{2k+2})^{2(2k+1)+3})^{2(2k+1)-1} \cdot n$, we have that $T(n) = O(k^5 \cdot (2k+1)^{(2k+1)-2} \cdot ((2(2k+1)+3)^{2(2k+1)+3} \cdot (\frac{8}{3} \cdot 2^{2k+2})^{2(2k+1)+3})^{2(2k+1)-1} \cdot n)$, i.e., linear with a constant factor that is exponential in k^3.

7. Final remarks. A consequence of the result in this paper is that all results that state that certain problems are solvable in linear time for graphs that are given together with a tree-decomposition of width bounded by a constant are turned into results that state that these problems can be solved in linear time on graphs with treewidth bounded by a constant. One of the most notable of such results is the following.

THEOREM 7.1. *Every class of graphs that does not contain all planar graphs and is closed under the taking of minors has a linear-time recognition algorithm.*

Proof. See, e.g., [31]. Use the algorithm described in this paper to find a tree-decomposition of the input graph with treewidth bounded by a constant, and use this tree-decomposition to test for minor inclusion for all graphs in the obstruction set of the class. □

Note that the result of Theorem 7.1 is nonconstructive: it relies on the nonconstructively proven fact that every minor-closed class of graphs has a finite characterization in terms of an obstruction set (see [31]). Thus we know that an algorithm exists, but the algorithm itself is not known. Even worse, even if an obstruction set and hence the algorithm were known for a certain class of graphs, the algorithm obtained with this method would only produce "yes" and "no" answers and would not construct any additional desired information. (For example, the result states that the class of graphs which are subgraphs of a planar graph with diameter at most d is linear-time recognizable for fixed d. However, such an algorithm would not produce such planar supergraphs of diameter at most d for "yes" instances.) In [17, 18], several classes of graphs to which Theorem 7.1 can be applied can be found. For several of these classes, we expect that constructive linear-time algorithms for recognition and construction of solutions can be found. Recent research [10] shows that linear-time algorithms can be constructed that solve minimum-cut linear arrangement, search

number, and some related problems for constant k and, for "yes" instances, output the required linear arrangement.

Note that the result shown in this paper is equivalent to stating that (for fixed k) partial k-trees can be recognized and embedded in a k-tree (or a chordal graph with maximum clique size $k+1$) in linear time. Also, a direct consequence is the following.

THEOREM 7.2. *For all $k \in N$, there exists a linear-time algorithm that tests whether a given graph $G = (V, E)$ has pathwidth at most k and, if so, outputs a path-decomposition of G with pathwidth at most k.*

Proof. First, use the algorithm described in this paper. When the treewidth of G is larger than k, then clearly we also have that the pathwidth of G is larger than k. Otherwise, use the result from [12, 22] that states that for all constants k and l, there exists a linear-time algorithm that, when given a graph G and a tree-decomposition of G with treewidth at most l, decides whether G has pathwidth at most k and, if so, outputs a path-decomposition of G with pathwidth at most k. □

The constant factor of the algorithm as derived above is very large—much too large for practical purposes. We remark that the analysis used some crude arguments, and it can be expected that the real constant factor of the algorithm is much smaller. However, the algorithm in its present form is probably not practical, even for $k = 4$. An interesting topic for further research is the development of a practical algorithm for the "treewidth $\leq k$" problem. Ideas and techniques in this paper may help to develop such algorithms. For instance, finding I-simplicial vertices is done quite fast and may be a good heuristic.

It is also possible to modify the algorithm such that it uses the algorithm in [12, 22] only on tree-decompositions with treewidth at most $k+1$ at the cost of increasing the running time to $O(n \log n)$. Provided that the algorithm in [12, 22] can be made fast enough, this modification may well be quite practical for small values of k (like $k = 4$ or $k = 5$). The idea is as follows: instead of using the construction of Lemma 3.3, first find a set M' of at least $|M|/(k+1)$ edges in M such that no two vertices that are the result of contracting an edge in M' belong to a common set X_i. (Such a set can be quickly found in $O(n)$ time: the graph H obtained by adding an edge between every pair of vertices in G' that share a common set X_i is a graph with treewidth k and hence is $(k+1)$-colorable. Hence the set of vertices that are a result of an edge contraction contains an independent set in H of size at least $|M|/(k+1)$. Take M', the set of edges corresponding to the vertices in this independent set.) Define $f_{M'}$ as in §2. Now (Y, T) defined by $Y_i = \{v \in v \mid f_{M'} \in X_i\}$ is a tree-decomposition of the graph obtained from G by contracting all edges in M' with treewidth at most $k+1$. Use the algorithm from [12, 22] to find a tree-decomposition of treewidth at most k of this graph. Repeat the process with this last tree-decomposition and edge set $M - M'$ until the edge set is empty. These are at most $O(\log n)$ iterations. (This observation was also made by Jens Lagergren.)

It is possible to implement the algorithm such that it runs on a pointer machine (a correct use of pointers is necessary such that the addressing in the bucket-sort algorithms can be done) and still uses linear time. We omit the (easy) details.

Very recently, using modifications of the techniques of this paper, parallel algorithms with a linear time–processor product were obtained for the "treewidth $\leq k$" problem [11].

Acknowledgments. The presentation of these results was improved considably by comments from Jens Gustedt, Torben Hagerup, Ton Kloks, Dieter Kratsch, Bruce Reed, and an anonymous referee.

REFERENCES

[1] K. R. ABRAHAMSON AND M. R. FELLOWS, *Finite automata, bounded treewidth and well-quasiordering*, in Proc. AMS Summer Workshop on Graph Minors, Graph Structure Theory, Contemporary Mathematics, Vol. 147, American Mathematical Society, Providence, RI, 1993, pp. 539–564.

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

[3] S. ARNBORG, D. G. CORNEIL, AND A. PROSKUROWSKI, *Complexity of finding embeddings in a k-tree*, SIAM J. Algebraic Discrete Meth., 8 (1987), pp. 277–284.

[4] S. ARNBORG, B. COURCELLE, A. PROSKUROWSKI, AND D. SEESE, *An algebraic theory of graph reduction*, J. Assoc. Comput. Mach., 40 (1993), pp. 1134–1164.

[5] S. ARNBORG, J. LAGERGREN, AND D. SEESE, *Easy problems for tree-decomposable graphs*, J. Algorithms, 12 (1991), pp. 308–340.

[6] S. ARNBORG AND A. PROSKUROWSKI, *Linear time algorithms for NP-hard problems restricted to partial k-trees*, Discrete Appl. Math., 23 (1989), pp. 11–24.

[7] H. L. BODLAENDER, *Dynamic programming algorithms on graphs with bounded tree-width*, in Proc. 15th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Comput. Sci. 317, Springer-Verlag, Berlin, 1988, pp. 105–119.

[8] ———, *A tourist guide through treewidth*, Acta Cybernetica, 11 (1993), pp. 1–23.

[9] ———, *Improved self-reduction algorithms for graphs with bounded treewidth*, Discrete Appl. Math., 54 (1994), pp. 101–115.

[10] H. L. BODLAENDER, M. R. FELLOWS, AND M. HALLETT, *Beyond NP-completeness for problems of bounded width: Hardness for the W hierarchy*, in Proc. 26th Annual Symposium on Theory of Computing, IEEE Computer Society Press, Los Alamitos, CA, New York, 1994, pp. 449–458.

[11] H. L. BODLAENDER AND T. HAGERUP, *Parallel algorithms with optimal speedup for bounded treewidth*, in Proc. 22nd International Colloquium on Automata, Languages, and Programming, Z. Fülöp and F. Gécseg, eds., Lecture Notes in Comput. Sci. 944, Springer-Verlag, Berlin, 1995, pp. 268–279; SIAM J. Comput., to appear.

[12] H. L. BODLAENDER AND T. KLOKS, *Efficient and constructive algorithms for the pathwidth and treewidth of graphs*, J. Algorithms, 1996, to appear.

[13] H. L. BODLAENDER AND R. H. MÖHRING, *The pathwidth and treewidth of cographs*, SIAM J. Discrete Math., 6 (1993), pp. 181–188.

[14] R. B. BORIE, R. G. PARKER, AND C. A. TOVEY, *Automatic generation of linear-time algorithms from predicate calculus descriptions of problems on recursively constructed graph families*, Algorithmica, 7 (1992), pp. 555–581.

[15] B. COURCELLE, *The monadic second-order logic of graphs* I*: Recognizable sets of finite graphs*, Inform. and Comput., 85 (1990), pp. 12–75.

[16] B. COURCELLE AND M. MOSBAH, *Monadic second-order evaluations on tree-decomposable graphs*, Theoret. Comput. Sci., 109 (1993), pp. 49–82.

[17] M. R. FELLOWS AND M. A. LANGSTON, *Nonconstructive advances in polynomial-time complexity*, Inform. Process. Lett., 26 (1987), pp. 157–162.

[18] ———, *Nonconstructive tools for proving polynomial-time decidability*, J. Assoc. Comput. Mach., 35 (1988), pp. 727–739.

[19] ———, *An analogue of the Myhill–Nerode theorem and its use in computing finite-basis characterizations*, in Proc. 30th Annual Symposium on Foundations of Computer Science, IEEE Computer Society Press, Los Alamitos, CA, 1989, pp. 520–525.

[20] ———, *On search, decision and the efficiency of polynomial-time algorithms*, J. Comput. System Sci., 49 (1994), pp. 769–779.

[21] D. S. JOHNSON, *The NP-completeness column: An ongoing guide*, J. Algorithms, 8 (1987), pp. 285–303.

[22] T. KLOKS, *Treewidth: Computations and Approximations*, Lecture Notes in Comput. Sci., 842, Springer-Verlag, Berlin, 1994.

[23] J. LAGERGREN, *Efficient parallel algorithms for graphs of bounded tree-width*, J. Algorithms, 20 (1996), pp. 20–44.

[24] J. LAGERGREN AND S. ARNBORG, *Finding minimal forbidden minors using a finite congruence*, in Proc. 18th International Colloquium on Automata, Languages, and Programming, Lecture Notes in Comput. Sci. 510, Springer-Verlag, Berlin, 1991, pp. 532–543.

[25] J. MATOUŠEK AND R. THOMAS, *Algorithms finding tree-decompositions of graphs*, J. Algorithms, 12 (1991), pp. 1–22.

[26] B. REED, *Finding approximate separators and computing tree-width quickly*, in Proc. 24th

Annual Symposium on Theory of Computing, Association for Computing Machinery, New York, 1992, pp. 221–228.

[27] N. ROBERTSON AND P. D. SEYMOUR, *Graph minors* II: *Algorithmic aspects of tree-width*, J. Algorithms, 7 (1986), pp. 309–322.

[28] ———, *Graph minors* V: *Excluding a planar graph*, J. Combin. Theory Ser. B, 41 (1986), pp. 92–114.

[29] ———, *Graph minors* IV: *Tree-width and well-quasi-ordering*, J. Combin. Theory Ser. B, 48 (1990), pp. 227–254.

[30] ———, *Graph minors* X: *Obstructions to tree-decomposition*, J. Combin. Theory Ser. B, 52 (1991), pp. 153–190.

[31] ———, *Graph minors* XIII: *The disjoint paths problem*, J. Combin. Theory Ser. B, 63 (1995), pp. 65–110.

[32] D. P. SANDERS, *On linear recognition of tree-width at most four*, SIAM J. Discrete Math., 9 (1996), pp. 101–117.

[33] P. SCHEFFLER, *Die Baumweite von Graphen als ein Maß für die Kompliziertheit algorithmischer Probleme*, Ph.D. thesis, Akademie der Wissenschaften der DDR, Berlin, 1989.

[34] J. VAN LEEUWEN, *Graph algorithms*, in Handbook of Theoretical Computer Science A: Algorithms and Complexity Theory, North–Holland, Amsterdam, 1990, pp. 527–631.

[35] T. V. WIMER, *Linear Algorithms on k-Terminal Graphs*, Ph.D. thesis, Department of Computer Science, Clemson University, Clemson, SC, 1987.