Efficient and constructive algorithms for the pathwidth and treewidth of graphs

Efficient and Constructive Algorithms for the Pathwidth and Treewidth of Graphs

Hans L. Bodlaender and Ton Kloks

RUU-CS-93-27
September 1993

Efficient and Constructive Algorithms for the Pathwidth and Treewidth of Graphs

Hans L. Bodlaender and Ton Kloks

Technical Report RUU-CS-93-27
September 1993

Department of Computer Science
Utrecht University
P.O.Box 80.089
3508 TB Utrecht
The Netherlands

ISSN: 0924-3275

Efficient and Constructive Algorithms for the Pathwidth and Treewidth of Graphs[*]

Hans L. Bodlaender[†] Ton Kloks[‡]

Abstract

In this paper we give, for all constants k, l, explicit algorithms, that given a graph $G = (V, E)$ with a tree-decomposition of G with treewidth at most l, decide whether the treewidth (or pathwidth) of G is at most k, and if so, find a tree-decomposition or (path-decomposition) of G of width at most k, and that use $O(|V|)$ time. In contrast with previous solutions, our algorithms do not rely on non-constructive reasoning, and are single exponential in k and l. This result can be combined with a result of Reed [37], yielding explicit $O(n \log n)$ algorithms for the problem, given a graph G, to determine whether the treewidth (or pathwidth) of G is at most k, and if so, to find a tree- (or path-)decomposition of width at most k (k constant). Also, Bodlaender [13] has used the result of this paper to obtain linear time algorithms for these problems.

We also show that for all constants k, there exists a polynomial time algorithm, that, when given a graph $G = (V, E)$ with treewidth $\leq k$, computes the pathwidth of G and a path-decomposition of G of minimum width.

1 Introduction

The notions of pathwidth and treewidth play an important role in many different fields of computer science, often with different terminologies, e.g.

- Choleski factorization and Gauss elimination. (See e.g. [20].)
- VLSI-layout theory. (See e.g. [34].)
- theory of expert systems. (See e.g. [32].)
- algorithmic graph theory.
- theory of graph grammars. (See e.g. [23].)

[*]A preliminary version of this paper appeared as *Better algorithms for the pathwidth and treewidth of graphs*, in the proceedings of ICALP'91.

[†]Department of Computer Science, Utrecht University, P.O.Box 80.089, 3508 TB Utrecht, the Netherlands. This author is partially supported by the ESPRIT II Basic Research Actions Program of the EC under contract no. 3075 (project ALCOM)

[‡]Department of Mathematics and Computing Science, Eindhoven University of Technology, P.O.Box 513, 5600 MB Eindhoven, The Netherlands. The work of this author was supported by the Foundation for Computer Science (S.I.O.N.) of the Netherlands Organization for Scientific Research (N.W.O.).

In many cases, notations and notions are different from those used in this paper (and from each other). For instance, a graph has treewidth at most k, iff it is a partial k-tree; iff it is the subgraph of a chordal (= triangulated) graph with no clique larger than $k+1$ vertices; iff it has dimension at most k. A graph has pathwidth at most k, iff its vertex separation number is at most k; iff its interval thickness is at most $k+1$; iff its node search number is at most $k+1$; iff it models an instance of the Gate Matrix Layout problem with a solution with at most $k+1$ tracks. There are also equivalent characterizations with help of graph grammars, or k-terminal recursive families of graphs. (See e.g., [2, 8, 22, 46].)

Formally, the treewidth (pathwidth) of a graph is the minimum treewidth (pathwidth) over all tree-decompositions (path-decompositions) of the graph. (See Section 2 for definitions.) When a tree- or path-decomposition is found of a graph G with optimal treewidth, then usually one can easily construct representations of the graph corresponding to the equivalent notions (e.g., chordal graphs with minimum clique size that contain G, optimal node search strategies, optimal solutions to the Gate Matrix Layout problem, etc.) Thus, given a graph G, finding a tree- or path-decomposition of G with minimum treewidth is an important problem.

The notion of treewidth is also interesting because of its vital role in the theory of Graph Minors of Robertson and Seymour [39]. Also, a very large number of intractable graph problems become solvable in polynomial, and even linear time (and belong to the class NC), when restricted to graphs with bounded treewidth, given together with a suitable tree-decomposition. This set of problems includes many well-known NP-complete problems like Hamiltonian Circuit, Independent Set, etc., and even some PSPACE-complete problems (see e.g. [5, 6, 9, 12, 19, 46]). Typically, these algorithms use time polynomial in the number of vertices, but at least exponential in the treewidth of the input graph. Also, researchers in expert system theory have found out that several otherwise time consuming statistical computations can be done quickly when a tree-decomposition (known as: junction tree, or clique tree) with small treewidth is known (see e.g., [32, 45].)

Much research has been done on the problem of determining the treewidth and pathwidth of a graph, and finding tree- or path-decompositions with optimal treewidth or pathwidth. These problems are NP-complete [3]. Research has been done on determining the treewidth and pathwidth of special classes of graphs (see e.g. [18, 17, 24, 27, 29, 28, 26, 36, 44], on approximation algorithms for treewidth and pathwidth (e.g. [15]), and on the case that the parameter k is a fixed constant. (See e.g. [11] for an overview.)

This paper addresses the case that k is a fixed constant. The first known algorithms, solving the treewidth and pathwidth problems for fixed k are based on dynamic programming and use respectively $O(n^{k+2})$ and $O(n^{2k^2+4k+8})$ time [3, 21].

Then, Robertson and Seymour [41] gave a non-constructive proof of the existence of $O(n^2)$ decision algorithms for the problems. Their algorithms consist of two steps. The first step either decides that the treewidth of the input graph G is too large, or finds a tree-decomposition of G of constant bounded but possible non-optimal width[1]. This step takes $O(n^2)$ time. Their second step checks in $O(n)$ time a finite characterization of the graphs with treewidth $\leq k$ or pathwidth $\leq k$. By Robertson and Seymours deep results on graph minors, these characterizations are known to exist; however they are not explicitly known. The linear time is achieved by using the tree-decomposition, found in

[1]To be precise, Robertson and Seymour use the notion of *branch decomposition* instead of tree-decomposition, but this forms an unimportant technical difference.

step one. Thus, these results are non-constructive in two ways: first, only existence of the algorithm is proven, but the algorithm itself is not known, and secondly the algorithm only outputs *yes* or *no*, but no tree- or path-decomposition. Also, the constant factors of these algorithms make them infeasible. With help of a *self-reduction* technique, introduced by Fellows and Langston [22], it is possible to obtain constructive $O(n^2)$ algorithms, but at the cost of a further increase of the constant factors [10].

Matousěk and Thomas [33], Lagergren [30], and Reed [37] improved on the first step. Lagergren [30] gives a parallel algorithm that uses $O(\log^3 n)$ time and $O(n)$ processors on a CRCW PRAM. Reed [37] gives a sequential $O(n \log n)$ algorithm. Arnborg et al. [4] use a slightly different technique, based on graph rewriting, and obtain decision algorithms, that use linear time, but polynomial memory.

This paper addresses the second step. It shows that we do not have to rely on non-constructive arguments, that instead, we give explicitly the algorithms, and our algorithms can also construct tree-decompositions or path-decompositions of width at most k, if existing. Our algorithms use linear time, and need as input, besides G a tree-decomposition of G of constant bounded width. Also, in contrast with the graph minors approach, the constant factor hidden in the O-notation of our algorithms is *only* singly exponential in k.

Recently, Bodlaender [13] used the result of this paper as an important intermediate step to obtain explicit and constructive algorithms that solve the 'treewidth $\leq k$' and 'pathwidth $\leq k$' problems in *linear time* (k fixed).

Results of a similar nature as ours were independently obtained by Lagergren and Arnborg [31] and by Abrahamson and Fellows [1].

It should be noted, that for $k = 1, 2, 3, 4$, linear time and space algorithms based on graph rewriting exist for the 'treewidth $\leq k$' problem [6, 33, 42].

We also solve a different, related problem, with basically the same algorithms: for each constant k, we have a polynomial time algorithm, that when given a graph $G = (V, E)$, computes the pathwidth of G and a path-decomposition of G of minimum width. This solves an open problem from [16]. So far, the only classes of graphs of bounded treewidth for which the complexity of the pathwidth problem was determined (besides classes of graphs with bounded pathwidth) were the trees and the forests: for these the pathwidth can be computed in linear time [21, 34, 43].

2 Definitions and Preliminary Results

The notions of treewidth and pathwidth were introduced by Robertson and Seymour [38, 40].

Definition 2.1
A tree-decomposition of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$ with $\{X_i \mid i \in I\}$ a collection of subsets of V, and $T = (I, F)$ a tree, such that

- $\bigcup_{i \in I} X_i = V$
- for all edges $(v, w) \in E$ there is an $i \in I$ with $v, w \in X_i$
- 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 width 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 $G = (V, E)$ is the minimum width over all tree-decompositions of G.

Definition 2.2
A path-decomposition of a graph $G = (V, E)$ is a sequence $(X_1, X_2, \cdots, X_r)$ of subsets of V, such that

- $\bigcup_{1 \leq i \leq r} X_i = V$
- for all edges $(v, w) \in E$ there is an i, $1 \leq i \leq r$ with $v, w \in X_i$
- for all i, j, k with $1 \leq i < j < k \leq r$: $X_i \cap X_k \subseteq X_j$.

The width of a path-decomposition $(X_1, \cdots, X_r)$ is $\max_{1 \leq i \leq r} |X_i| - 1$. The pathwidth of a graph $G = (V, E)$ is the minimum width over all path-decompositions of G.

We will use (X, T) as a shorthand notation for $(\{X_i \mid i \in I\}, T = (I, F))$. Sometimes we write a path-decomposition as a tree-decomposition, where the tree T has only nodes with degree at most 2. We now introduce some extra terminology, related to tree-decompositions.

Definition 2.3
A *rooted* tree-decomposition is a tree-decomposition $D = (X, T)$ in which T is a rooted tree.

Definition 2.4
Let $D = (X, T)$ be a rooted tree-decomposition for a graph G. For each node i of T, let T_i be the subtree of T, rooted at node i. Define: $V_i = \bigcup_{f \in T_i} X_f$ and let $G_i = G[V_i]$ (so if r is the root of T, $G_r = G$). We call G_i the subgraph of G *rooted at* i.

We can obtain a rooted tree-decomposition $D_i = (X^i, T_i)$ for G_i from D:

Definition 2.5
Let $D = (S, T)$ be a rooted tree-decomposition for a graph G. Let i be a node of T. Let $D_i = (X^i, T_i)$, where T_i is the subtree of T rooted at i, and $X^i = \{X_f \mid f \in T_i\}$. We call D_i the rooted tree-decomposition of G_i *rooted at node* i.

Lemma 2.1 *For each node i, D_i is a tree-decomposition of G_i.*

Proof: For the simple proof, see e.g. [25]. □

In order to describe our algorithms more easily, we introduce a special type of rooted tree-decompositions.

Definition 2.6
A rooted tree-decomposition $D = (S, T)$ with $S = \{X_i \mid i \in I\}$ and $T = (I, F))$ is called a *nice* tree-decomposition, if the following conditions are satisfied:

1. every node of T has at most two children,
2. if a node i has two children j and k, then $X_i = X_j = X_k$
3. if a node i has one child j, then either $|X_i| = |X_j| + 1$ and $X_j \subset X_i$ or $|X_i| = |X_j| - 1$ and $X_i \subset X_j$.

Lemma 2.2 *Every graph G with treewidth k has a nice tree-decomposition of width k. Furthermore, if n is the number of vertices of G then there exists a nice tree-decomposition with at most $4n$ nodes.*

We omit the proof. See e.g. [25]. Also, the following result can be obtained:

Lemma 2.3 *For* constant k, *given a tree-decomposition of a graph G of width k and $O(n)$ nodes, where n is the number of vertices of G, one can find a* nice *tree-decomposition of G of width k and with at most $4n$ nodes in $O(n)$ time.*

Definition 2.7
In a nice tree-decomposition $(\{X_i | i \in I\}, T = (I, F))$ every node is of one of four possible types. We name the types as follows.

"Start" If a node is a leaf, it is called a *start node.*

"Join" If a node has two children, it is called a *join node.*

"Forget" If a node i has one child j and if $|X_i| < |X_j|$, node i is called a *forget node.*

"Introduce" If a node i has one child j and if $|X_i| > |X_j|$, node i is called an *introduce node.*

Notice that every node in the nice tree-decomposition must have one of the four mentioned labels.

We may also assume, that if i is a **start** node, then $|X_i| = 1$: the effect of **start** nodes with $|X_i| > 1$ can be obtained with using a **start** node with a one-vertex set, and then $|X_i| - 1$ **introduce** nodes, that add all other vertices.

Our algorithms roughly work as follows. Given a tree-decomposition of G, we first make a nice tree-decomposition with the same width of G, as indicated by lemma 2.2 and lemma 2.3. We define an equivalence relation on path- or tree-decompositions of subgraphs G_i (determined by the *characteristic* of such 'partial' path- or tree-decompositions). For each node $i \in I$, we compute a table of the 'most relevant' equivalence classes which contain a tree- or path-decomposition of G_i with treewidth or pathwidth $\leq k$. These tables are computed in a bottom-up order, starting with the leaves of tree T, and using the tables of the children of a node to compute the table of the node. The table of the root node is non-empty, if and only if the treewidth or pathwidth of G is at most k.

3 Partial path-decompositions, the interval model and typical sequences

In this section we give many notions and small results that deal with partial path-decompositions and sequences of integers.

Definition 3.1
A *partial path-decomposition rooted at node* $i \in I$ is a path-decomposition for G_i, the subgraph of G rooted at i.

The equivalence class to which a partial path-decomposition rooted at a node $i \in I$ belongs, is described by its *characteristic*, which is a pair of which the first element is the interval model of the path-decomposition, as defined hereafter.

Definition 3.2
Let $Y = (Y_1, \ldots, Y_r)$ be a partial path-decomposition rooted at node i. The *restriction of* Y is the sub-decomposition Y^* of Y for the subgraph induced by X_i, i.e. $Y^* = (Y_1 \cap X_i, \ldots, Y_r \cap X_i)$.

In the restriction Y^* there can be many consecutive elements which are the same. If we remove these duplicates, we obtain the interval model for Y which is, of course, still a path-decomposition for the subgraph induced by X_i.

Definition 3.3
Let $Y^* = (Z_1, \ldots, Z_r)$ be the restriction of a path-decomposition Y rooted at i. Let $1 = t_1 < \ldots < t_{q+1} = r + 1$ be defined by:

$$\forall_{1 \leq i \leq q} \forall_{t_i \leq s < t_{i+1}} [Z_s = Z_{t_i}] \wedge \forall_{1 \leq i < q} [Z_{t_{i+1}} \neq Z_{t_i}]$$

The *interval model* for Y at node i is the sequence $(Z_{t_i})_{1 \leq i \leq q}$.

Notice that *not every* path-decomposition for a subgraph induced by X_p without repeating subsets is an interval model, since an interval model is defined by means of a partial path-decomposition rooted at p. We call a path-decomposition for the subgraph X_p without adjacent subsets that are the same, minimal:

Definition 3.4
A path-decomposition Z for a graph G is called *minimal* if no two consecutive subsets in Z are the same.

The next lemma shows that there are only $\Theta(1)$ different interval models at each node i.

Lemma 3.1 *For each node i the number of different interval models at i is bounded by $(2k+3)^{2k+3}$. The number of subsets in any interval model is at most $2k+3$. These bounds hold for the minimal path-decompositions for the subgraph induced by X_i as well.*

Proof: An interval model at node i is a path-decomposition $Z = (Z_1, \ldots, Z_r)$ for $G[X_i]$ which is minimal. We show the bounds hold for the minimal path-decompositions. Let $L(s)$ be the maximal number of subsets in a minimal path-decomposition of a graph with s vertices. We claim that $L(s) \leq 2s + 1$. Clearly, $L(1) = 3$. Now let $s > 1$, and let $Z = (Z_1, \ldots, Z_r)$ be a minimal path-decomposition for a graph $H = (V, E)$ with s vertices. Take any vertex x and let Z_a and Z_b be the first and the last subset of Z containing x. Now remove x from the graph and let $H' = H[V \setminus \{x\}]$. We can obtain a path-decomposition Z' for H' by removing vertex x from all subsets of Z. Notice that Z' can have at most two pairs of duplicate subsets, namely Z'_a can be the same as Z'_{a-1} and Z'_b can be the same as Z'_{b+1}. It follows that the number of subsets of Z is at most two more than the maximal number of subsets in a path-decomposition of a graph with $s - 1$ vertices. Hence $L(s) \leq L(s-1) + 2$. This proves our claim. Since $|X_i| \leq k + 1$, the number of subsets in a minimal path-decomposition of $G[X_i]$ is at most $2k + 3$.

We can find an upper bound for the number of interval models as follows. Notice that an interval model can be characterized by indicating for each vertex the first and last subset where it is contained in. Thus we find an upper bound of i^{2k+2} for the number of interval models with i subsets. Hence we find:

$$\text{number of interval models} \leq \sum_{i=1}^{2k+3} i^{2k+2} \leq (2k+3)^{2k+3}$$

This proves the lemma. □

Next, we define *typical sequences* of integer sequences. We use the term *integer sequence* to denote a sequence of at least one nonnegative integer. (These sequences are used to denote sizes of successive sets in a path-decomposition.) We use the following notations:

- For any integer sequence $a(1 \ldots n)$, let $l(a) = n$ be the length and $max(a)$ be the maximum value: $max(a) = \max_{1 \leq i \leq n} a_i$.
- For two sequences a and b *of the same length* we define the sum $c = a + b$ as the sequence c with
$$\forall_{1 \leq i \leq l(a)} \, [\, c_i = a_i + b_i \,]$$
- For two sequences a and b *of the same length* we write $a \leq b$ if $\forall_i \; a_i \leq b_i$.
- For a *constant* A we write $a + A$ for the sequence with $\forall_i \; (a + A)_i = a_i + A$.

Definition 3.5
For an integer sequence $a(1 \ldots n)$ we define the *typical sequence* $\tau(a)$ as the sequence obtained after iterating the following operations, until none is possible anymore.

- Remove consecutive repetitions of the same element, i.e. if $a_i = a_{i+1}$ then the sequence $a = (a_1, \ldots, a_n)$ is replaced by $(a_1, \ldots, a_i, a_{i+2}, \ldots, a_n)$.
- If the sequence contains two elements a_i and a_j such that $j - i \geq 2$ and $\forall_{i<k<j} \; a_i \leq a_k \leq a_j$ or $\forall_{i<k<j} \; a_i \geq a_k \geq a_j$, then remove the subsequence $a(i+1 \ldots j-1)$, i.e. replace $a = (a_1, \ldots, a_n)$ by $(a_1, \ldots, a_i, \; a_j, \ldots, a_n)$.

We refer to the second operation as the *typical operation.*

We say integer sequence a is a typical sequence, if a is the typical sequence of at least one integer sequence, or, equivalently, if $a = \tau(a)$, i.e., if a is the typical sequence of at least one other sequence.

Lemma 3.2 *For every a, the typical sequence $\tau(a)$ is uniquely defined.*

Proof: We must show that the order in which the typical operations and removal of repetitions are applied does not influence the resulting typical sequence. This can be shown by observing that if a_k can be removed by a typical operation or removal of repetitions, then this remains true under any typical operation or removal of repetition (unless, of course, such an operation removes a_k.) □

Lemma 3.3 *Let $a(1 \ldots n)$ be a sequence of nonnegative integers with $max(a) = L$. Then*
(i) $max(\tau(a)) = L$
(ii) $l(\tau(a)) \leq 2L + 1$.

Proof: (i) Trivial.

(ii) Define $\tilde{T}(R)$ ($\tilde{T}'(R)$) as the maximum length of a typical sequence that contains exactly R different integers, and starts with the smallest (largest) integer, and that does not contain a second occurrence of this smallest (largest) integer. Note that in a typical sequence, starting with the smallest integer, the largest integer cannot occur at any other position than the second one in the sequence (otherwise a typical operation can be applied, removing everything between the smallest and largest integer). Hence $\tilde{T}(R) = \tilde{T}'(R-1)+1$, Similarly, $\tilde{T}'(R) = \tilde{T}(R-1)+1$. As $\tilde{T}(1) = \tilde{T}'(1) = 1$, we have that $\tilde{T}(R) = \tilde{T}'(R) = R$.

Consider the typical sequence $\tau(a)$. Between an occurrence of 0 and of L, there cannot be other integers, otherwise, a typical operation can be applied, removing everything between 0 and L. So, $\tau(a)$ contains at most one 0, or at most one L. In the former case, $\tau(a)$ is of the form $b0c$, with b and c strings that do not contain a 0. So $b0$ and $0c$ have length at most $\tilde{T}(L+1) = L+1$. In the latter case, $\tau(a)$ is of the form bLc, with the length of bL and Lc at most $\tilde{T}'(L+1) = L+1$. In both cases, the length of $\tau(a)$ is at most $2L+1$. □

Remark 3.4 *The bound of lemma 3.3(ii) is sharp: consider the sequences*

$$\cdots (L-2)\ 2\ (L-1)\ 1\ L\ 0\ L\ 1\ (L-1)\ 2\ (L-2) \cdots$$

Lemma 3.5 *The number of different typical sequences of integers in $\{0, 1, \ldots, L\}$ is at most $\frac{8}{3}2^{2L}$.*

Proof: For a set of integers S, define $\tilde{N}(S)$ ($\tilde{N}'(S)$) to be the set of typical sequences, that

- contain each integer in S
- start with the smallest (largest) integer in S
- contain the smallest (largest) integer in S exactly once.

Write $S = \{n_1, n_2, \ldots, n_s\}$, $n_1 < n_2 < \cdots < n_s$. Using induction, one can prove, similar as in the proof of lemma 3.3 that $\tilde{N}(S)$ contains one unique element:

$$n_1\ n_s\ n_2\ n_{s-1}\ n_3\ n_{s-2} \cdots$$

and $\tilde{N}'(S)$ also contains a unique element:

$$n_s\ n_1\ n_{s-1}\ n_2\ n_{s-2}\ n_3 \cdots$$

We will now first count the number of typical sequences in $\{0, 1, \ldots, L\}^*$ that contain their smallest integer m once. There is a unique correspondence between such typical sequences and pairs of subsets S_1, $S_2 \subseteq \{m+1, \ldots, L\}$: S_1 denotes the set of integers appearing before m, and S_2 denotes the set of integers after m. The typical sequence corresponding to pair S_1, S_2 is the string of the form $a\ m\ b$, with $a\ m$ the unique element of $\tilde{N}(S_1)$ in reversed order, and $m\ b$ the unique element of $\tilde{N}(S_2)$. So, the number of typical sequences that contain their smallest element once is

$$\sum_{m=0}^{L} 2^{2m} < \frac{4}{3} 2^{2L}$$

As each typical sequence contains its smallest integer once, or its largest integer once, and those of the latter type can be counted similarly, the result follows. □

For sequences $a(1 \ldots n)$, and $b(1 \ldots n)$, we write $a \leq b$, if for all i, $1 \leq i \leq n$, $a_i \leq b_i$.

Definition 3.6
Let $a(1..n)$ be a sequence. We define $E(a)$ as the set of *extensions* of a:

$$E(a) = \{a^* \mid \exists_{1=t_1<t_2<\ldots<t_{n+1}} \forall_{1 \leq i \leq n} \forall_{t_i \leq k < t_{i+1}} [\, a^*(k) = a(i)\,]\}$$

Hence each element of $E(a)$ is of the form $(a_1, a_1, \ldots, a_2, a_2, \ldots, a_n, \ldots, a_n)$, where each a_i of the original sequence a appears at least once in the extension. For any interval $[\, \alpha, \beta\,]$ with $t_i \leq \alpha \leq \beta < t_{i+1}$ we say that $a(i)$ is *repeated* in this interval (in a^*).

Lemma 3.6 *If $a^* \in E(a)$ then $\tau(a^*) = \tau(a)$.*

Proof: In computing $\tau(a^*)$ we may start by removing all repetitions. □

Definition 3.7
For two integer sequences a and b we write $a \prec b$ if there are $a^* \in E(a)$ and $b^* \in E(b)$ *of the same length* such that $a^* \leq b^*$. If both $a \prec b$ and $b \prec a$ hold we write $a \equiv b$.

Lemma 3.7 *The relation $\prec$ is transitive.*

Proof: Let $a \prec b$ and $b \prec c$. First notice that:

$$b \prec c \wedge b^* \in E(b) \Rightarrow b^* \prec c$$

We show there exist extensions $a^* \in E(a)$ and $b^* \in E(b)$ such that $a^* \leq b^*$. By the remark above, $b^* \prec c$ and hence there are extensions $b^{**} \in E(b^*)$ and $c^* \in E(c)$ such that $b^{**} \leq c^*$. We make an extension $a^{**} \in E(a^*)$ as follows. If an element of b^* is repeated in b^{**} then we let the corresponding element of a^* repeat in a^{**}. Clearly,

$$a^{**} \leq b^{**} \leq c^*$$

and hence $a \prec c$. □

Corollary 3.8 *The relation $\equiv$ is an equivalence relation.*

Lemma 3.9 *If a sequence a' is obtained from a sequence a by a typical operation then $a' \equiv a$. Moreover, there exist extensions a'^* and a'^{**} both of a' such that $a'^* \leq a \leq a'^{**}$.*

Proof: Suppose $a = (a_1, \ldots, a_n)$, and $a' = (a_1, \ldots, a_i, a_j, \ldots, a_n)$. Without loss of generality, suppose that $a_i \leq a_j$. Take

$$a'^* = (a_1, \ldots, a_{i-1}, \underbrace{a_i, \ldots, a_i}_{j-i+1 \text{ times}}, a_j, a_{j+1}, \ldots, a_n)$$

and

$$a'^{**} = (a_1, \ldots, a_{i-1}, a_i, \underbrace{a_j, \ldots, a_j}_{j-i+1 \text{ times}}, a_{j+1}, \ldots, a_n)$$

Clearly, a'^* and a'^{**} are extensions of a' with $a'^* \leq a \leq a'^{**}$. Hence $a' \equiv a$. □

Lemma 3.10 *For any integer sequence a: $\tau(a) \equiv a$. Moreover, there exist extensions a' and a'' of $\tau(a)$ and both of the same length as a such that $a'_i \leq a_i$ and $a''_i \geq a_i$ for all i.*

Proof: Using lemma 3.7, one easily proves with induction to r: if b is obtained from a by r typical operations or removals of repetitions, then $a \equiv b$ and there exists extensions a', a'' of b of the same length as a, such that $a' \leq a$ and $a \leq a''$. □

From Lemma 3.10 and Lemma 3.7 it follows that:

Corollary 3.11 *If a and b are two sequences then $a \prec b$ if and only if $\tau(a) \prec \tau(b)$.*

Definition 3.8
Let $a(1..n)$ and $b(1..m)$ be two integer sequences. The *ringsum* $a \oplus b$ is defined as:

$$a \oplus b = \{a^* + b^* \mid a^* \in E(a) \text{ and } b^* \in E(b) \text{ and } l(a^*) = l(b^*)\}$$

.

Lemma 3.12 *Let $c \in a \oplus b$, and let $a^* \in E(a)$ and $b^* \in E(b)$. Then there exists a sequence $c^* \in E(c)$ such that $c^* \in a^* \oplus b^*$.*

Proof: Let $c = a' + b'$ for some $a' \in E(a)$ and $b' \in E(b)$. Let a_i be repeated p'_i times in a' and p^*_i times in a^*. Let b_i be repeated q'_i times in b' and q^*_i times in b^*. Let $\lambda \geq 1$ be an integer. Make new extensions $a^\circ \in E(a)$ and $b^\circ \in E(b)$, by repeating a_i $\lambda p'_i$ times in a° and b_j $\lambda q'_j$ times in b°. Then a° and b° have the same length. If $c^* = a^\circ + b^\circ$, then $c^* \in E(c)$. If we take λ such that $\lambda p'_i \geq p^*_i$ for all i and $\lambda q'_j \geq q^*_j$ for all j then also $a^\circ \in E(a^*)$ and $b^\circ \in E(b^*)$. □

Next we show that if two sequences can be 'improved', then also the sum can be improved.

Lemma 3.13 *Let a and b be two integer sequences of the same length and let $y = a + b$. Let $a_0 \prec a$ and $b_0 \prec b$. Then there is a sequence $y_0 \in a_0 \oplus b_0$ such that $y_0 \prec y$.*

Proof: There are extensions a^*_0 of a_0 and a^* of a such that $a^*_0 \leq a^*$ and extensions b^*_0 of b_0 and b^* of b such that $b^*_0 \leq b^*$. Assume an element a_i is repeated p_i times in a^* and b_i is repeated q_i times in b^*. Now change the extensions a^* and a^*_0 into a^{**} and a^{**}_0 by repeating a_i $p_i q_i$ times in a^{**} and repeating each corresponding element in a^*_0 q_i times. We then have $a^{**}_0 \leq a^{**}$. In a similar way we obtain new extensions b^{**}_0 and b^{**}. Make an extension y^{**} of y by repeating each element y_i $p_i q_i$ times. Define $y_0 = a^{**}_0 + b^{**}_0$. We now have $y_0 = a^{**}_0 + b^{**}_0 \leq a^{**} + b^{**} = y^{**}$ and $y_0 \in a_0 \oplus b_0$. □

Lemma 3.14 *Let a and b be two integer sequences and let $c \in a \oplus b$. Then there exists an element $c' \in \tau(a) \oplus \tau(b)$ such that $c' \prec c$.*

Proof: This is an immediate consequence of Lemma 3.13. By Definition 3.8 $c = a^* + b^*$ for some extensions a^* of a and b^* of b. By Lemma 3.6 $\tau(a^*) = \tau(a)$ and $\tau(b^*) = \tau(b)$. By Lemma 3.10 $\tau(a^*) \prec a^*$ and $\tau(b^*) \prec b^*$. Hence, by Lemma 3.13, there is a $c^* \in \tau(a) \oplus \tau(b)$ such that $c^* \prec c$. □

Lemma 3.15 *Let a and b be two integer sequences, and $c \in a \oplus b$. Then there exists an integer sequence $c' \in a \oplus b$ with $\tau(c) = \tau(c')$ and $l(c') \leq l(a) + l(b) - 1$.*

Proof: Let a^*, b^* be extensions of a and b, such that $l(a^*) = l(b^*) = m$, and $c = a^* + b^*$. We write $a = (a_1, \ldots, a_n)$, $b = (b_1, \ldots, b_{n'})$, $a^* = (a^*_1, \ldots a^*_m)$, $b^* = (b^*_1, \ldots, b^*_m)$. Let $I = \{i \mid 1 \leq i \leq m \wedge (a^*_i \neq a^*_{i+1} \vee b^*_i \neq b^*_{i+1})\}$. Write $I = \{i_1, \ldots, i_r\}$, $i_1 < i_2 < \cdots < i_r$. As positions in I mark either the last occurrence of a repetion of a value a_j or of a value b_j, we have $|I| \leq n + n' - 2$. Let $c' = (a^*_{i_1} + b^*_{i_1}, a^*_{i_2} + b^*_{i_2}, \ldots, a^*_{i_r} + b^*_{i_r}, a^*_m + b^*_m)$. Note that c is an extension of c', so $\tau(c) = \tau(c')$. The length of c' is $l(c') = |I| + 1 = l(a) + l(b) - 1$. □

Lemma 3.16 *Let a be an integer sequence with $l(a) \leq k$. The number of different extensions a^* of a with $l(a^*) = k$ is at most 2^{k-1}.*

Proof: We have $a^*_1 = a_1$. For every i, $2 \leq i \leq k$, there are at most two choices for a^*_i: either we repeat the last element ($a^*_i = a^*_{i-1}$), or we take the next element from a: ($a*_{i-1} = a_j$, $a^*_i = a_{j+1}$ for some j). □

Definition 3.9
Let $a(1 \ldots n)$ and $b(1 \ldots m)$ be two integer sequences. The *concatenation* of a and b, is defined as the sequence:

$$\circ ab = (a_1, \ldots, a_n, b_1, \ldots, b_m)$$

Lemma 3.17 *For two sequences a and b: $\tau(\circ ab) = \tau(\circ\tau(a)\tau(b))$.*

Proof: By Lemma 3.2 we can apply typical operations and removal of duplicates in any order to obtain $\tau(\circ ab)$. Start by applying the typical operations of a to the sequence $\circ ab$ and remove adjacent duplicates from this sublist. The result is the sequence $\circ\tau(a)b$. Next apply all typical operations and removal of duplicates to the sublist b. The result is $\circ\tau(a)\tau(b)$. This proves the lemma. □

Lemma 3.18 *If $a^* \in E(a)$ and $b^* \in E(b)$ then $\circ a^*b^* \in E(\circ ab)$.*

Lemma 3.19 *If $a' \prec a$ and $b' \prec b$, then $\circ a'b' \prec \circ ab$.*

Proof: There are extensions $a'^* \in E(a')$, $a^* \in E(a)$, $b'^* \in E(b')$ and $b^* \in E(b)$ such that $a'^* \leq a^*$ and $b'^* \leq b^*$. Then clearly also: $(\circ a'^*b'^*) \leq (\circ a^*b^*)$. By Lemma 3.18: $\circ a^*b^* \in E(\circ ab)$ and $\circ a'^*b'^* \in E(\circ a'b')$. This proves the lemma. □

Definition 3.10
Let $a(1 \ldots n)$ be an integer sequence ($n > 0$). A *split* of a is a pair (δ_1, δ_2) of integer sequences of one of two types.

1. The *first type* split is such that there exists an index $1 \leq f \leq n$ with: $\delta_1 = (a_1, \ldots, a_f)$ and $\delta_2 = (a_f, \ldots, a_n)$;
2. The *second type* split is such that there is an index $1 \leq f \leq n$ with: $\delta_1 = (a_1, \ldots, a_f)$ and $\delta_2 = (a_{f+1}, \ldots, a_n)$.

Notice that a_f occurs in both elements of the split of the first type. For an integer sequence of length one there can only be a split of the first type, since we assumed that integer sequences always have length at least one.

Lemma 3.20 *Let a be a nonempty sequence such that $a \in E(\tau(a))$. Let (δ_1, δ_2) be a split of $\tau(a)$ of any type. Let (a_1, a_2) be a split of a of the same type such that $a_1 \in E(\delta_1)$ and $a_2 \in E(\delta_2)$ (this split exists). Then $\tau(a_1) = \delta_1$ and $\tau(a_2) = \delta_2$.*

Proof: Write $\tau(a) = (\alpha_1, \ldots, \alpha_s)$ and let (δ_1, δ_2) be a split of the first type with $\delta_1 = (\alpha_1, \ldots, \alpha_f)$ and $\delta_2 = (\alpha_f, \ldots, \alpha_s)$. Make a split of a of the first type such that

$$a_1 = (\alpha_1, \ldots, \alpha_1, \ldots, \alpha_f, \ldots, \alpha_f) \wedge a_2 = (\alpha_f, \ldots, \alpha_f, \ldots, \alpha_s, \ldots, \alpha_s)$$

(with α_f appearing at least once in each a_i). This split clearly is possible since $a \in E(\tau(a))$. Since $a \in E(\tau(a))$, $\tau(a)$ is obtained from a by removing repetitions of elements in a. Clearly, δ_i contains no repetitions, and no typical operation is applicable to it. If the split (δ_1, δ_2) is of the second type, the proof is similar. Hence the lemma follows. □

We extend the results on integer sequences to *lists of integer sequences*. We use the notation $[a]$ to represent a list $(a^{(1)}, a^{(2)}, \ldots, a^{(n)})$ where each $a^{(i)}$ represents an integer sequence. For short, we call a list of integer sequences also a list. We start with some notations.

1. The length of a list is the number of integer sequences in the list.
2. For a list $[a] = (a^{(1)}, \dots, a^{(n)})$ we define $max([a]) = \max_{1 \le i \le n} max(a^{(i)})$.
3. For two lists $[a] = (a^{(1)}, \dots, a^{(n)})$ and $[b] = (b^{(1)}, \dots, b^{(n)})$ of the same length and such that $l(a^{(i)}) = l(b^{(i)})$ for all i, we say that $[a]$ and $[b]$ have the same length *in the strong sense.*
4. For two lists $[a]$ and $[b]$ with the same length in the strong sense we write $[a] \le [b]$ if $a^{(i)} \le b^{(i)}$ for each i.
5. For two such lists with the same length in the strong sense we use the notation $[a] + [b]$ for the list $(a^{(1)} + b^{(1)}, \dots, a^{(n)} + b^{(n)})$.
6. Let $[a] = (a^{(1)}, \dots, a^{(n)})$ be a list. The typical list $\tau[a]$ of $[a]$ is the list
$$\tau[a] = (\tau(a^{(1)}), \dots, \tau(a^{(n)}))$$
7. Let $[a] = (a^{(1)}, \dots, a^{(n)})$ be a list. The set of *extensions* of $[a]$ is defined as:
$$E[a] = \{[b] = (b^{(1)}, \dots, b^{(n)}) \mid \forall_i\, b^{(i)} \in E(a^{(i)})\}$$
8. The *ringsum* of two lists $[a] = (a^{(1)}, \dots, a^{(n)})$ and $[b] = (b^{(1)}, \dots, b^{(n)})$ *of the same length* is defined as
$$[a] \oplus [b] = \{(c^{(1)}, \dots, c^{(n)}) \mid \forall_i\, c^{(i)} \in a^{(i)} \oplus b^{(i)}\}$$
9. For two lists $[a]$ and $[b]$ *of the same length* we write $[a] \prec [b]$ if there exist extensions $[a^*] \in E[a]$ and $[b^*] \in E[b]$ such that $[a^*] \le [b^*]$. If both $[a] \prec [b]$ and $[b] \prec [a]$ we write $[a] \equiv [b]$.

Most results on integer sequences trivially extend to lists of integer sequences. We summarize them in the following lemma.

Lemma 3.21

1. *The relation $\prec$ is transitive for lists and $\equiv$ is an equivalence relation for lists (Lemma 3.7 and Corollary 3.8).*
2. *If $[b] \in E[a]$ then $\tau[b] = \tau[a]$ (Lemma 3.6).*
3. *For two lists $[a]$ and $[b]$* of the same length*: $[a] \prec [b] \Leftrightarrow \tau[a] \prec \tau[b]$ (Corollary 3.11).*
4. *For any list $[a]$: $\tau[a] \equiv [a]$. Moreover there are extensions $[b'] \in E(\tau[a])$ and $[b''] \in E(\tau[a])$ such that $[b'] \le [a] \le [b'']$ (Lemma 3.10).*
5. *Let $[a]$ and $[b]$ be two lists of the same length and let $[c] \in [a] \oplus [b]$. Let $[a^*] \in E[a]$ and $[b^*] \in E[b]$. Then there exists a list $[c^*] \in E[c]$ such that $[c^*] \in [a^*] \oplus [b^*]$ (Lemma 3.12).*
6. *Let $[a]$ and $[b]$ be two list with the same length in the strong sense and let $[y] = [a]+[b]$. Let $[a_0] \prec [a]$ and $[b_0] \prec [b]$. Then there exists a list $[y_0] \in [a_0] \oplus [b_0]$ such that $[y_0] \prec [y]$ (Lemma 3.13).*
7. *Let $[a]$ and b be two lists* of the same length *and let $[c] \in [a] \oplus [b]$. There exists a list $[c'] \in \tau[a] \oplus \tau[b]$ such that $[c'] \prec [c]$ (Lemma 3.14).*

4 A decision algorithm for pathwidth

In this section we give a decision algorithm for the pathwidth $\leq k$ problem for fixed k. We assume we have a nice tree-decomposition (X, T) of the graph $G = (V, E)$ of width at most ℓ. We consider partial path-decompositions, rooted at nodes $i \in I$. We first define the 'characteristic of a partial path-decomposition': this is — in essence — the information of this partial path-decomposition that is sufficient to see whether it can be extended to a path-decomposition of G.

4.1 Characteristic path-decompositions

Definition 4.1
Let Y be a partial path-decomposition rooted at a node i. Let $Z = (Z_{t_j})_{1 \leq j \leq q}$ be the interval model for Y. The *list representation* for Y is the pair $(Z, [Y])$, where Z is the interval model and $[Y] = (Y^{(1)}, Y^{(2)}, \ldots, Y^{(q)})$ is the sequence with $Y^{(m)} = (Y_{t_m}, Y_{t_m+1}, \ldots, Y_{t_{m+1}-1})$ for each $1 \leq m \leq q$.

Definition 4.2
Let $Y = (Y_1, Y_2, \ldots, Y_m)$ be a partial path-decomposition. The set of *extensions* of Y, $E(Y)$, is the set of path-decompositions

$$Z = (Y_1, Y_1, \ldots, Y_1, Y_2, Y_2, \ldots, Y_2, \ldots, Y_m, \ldots, Y_m)$$

where each subset Y_i is repeated at least once.

Definition 4.3
Let Y be a partial path-decomposition with list representation $(Z, [Y])$, with interval model $Z = (Z_{t_j})_{1 \leq j \leq q}$. Let $[y] = (y^{(1)}, y^{(2)}, \ldots, y^{(q)})$ be the list of integer sequences with $y^{(m)} = (|Y_{t_m}|, |Y_{t_m+1}|, \ldots, |Y_{t_{m+1}-1}|)$ for each interval $1 \leq m \leq q$. We call $[y]$ the list of Y and $\tau[y]$ the typical list of Y.

Definition 4.4
Let Y be a partial path-decomposition with list representation $(Z, [Y])$ and let $[y]$ be the list of Y. The *characteristic* of Y is the pair

$$C(Y) = (Z, \tau[y])$$

Lemma 4.1 *The number of different characteristics of possible partial path-decompositions with pathwidth at most k, rooted at i, with $|X_i| \leq \ell + 1$, is at most*

$$(2\ell + 3)^{2\ell+3} \cdot \left(\frac{8}{3} \cdot 2^{2k+2}\right)^{2\ell+3}$$

Proof: This follows directly from lemma 3.1 and lemma 3.5. □

Definition 4.5
For two partial path-decompositions Y and Z rooted at the same node i, *which have the same interval model*, we write $Y \prec Z$ if the corresponding lists satisfy $[y] \prec [z]$. If $Y \prec Z$ and $Z \prec Y$, we write $Y \equiv Z$.

Definition 4.6
A set of characteristics $FS(i)$ of partial path-decompositions rooted at some node i of width at most k is called a *full set of characteristics* if for each partial path-decomposition Y rooted at i of width at most k there is a path-decomposition $Y' \prec Y$ such that the characteristic of Y' is in $FS(i)$.

Lemma 4.2 *If some full set of characteristics at a node i is nonempty, then* every *full set of characteristics at this node is nonempty. A full set of characteristics is nonempty if and only if the pathwidth of G_i is at most k.*

Proof: This follows directly from Definition 4.6. □

An important consequence of lemma 4.2 is that the pathwidth of G is at most k, if and only if any full set of characteristics at the root of the tree-decomposition is non-empty. In the next four subsections we show how to compute a full set of characteristics at a node p in $O(1)$ time, when a full set of characteristics of all the children of p is given.

4.2 A full set for a start node

We may assume that X_p contains one vertex, i.e., $X_p = \{v\}$ for some $v \in V$. By lemma 3.1, a minimal path-decomposition of G_p has at most three nodes. There are four different minimal path-decompositions of G_p: $(\emptyset, \{v\}, \emptyset)$, $(\emptyset, \{v\})$, $(\{v\}, \emptyset)$, and $(\{v\})$. For each of these, we put its characteristic in the full set.

4.3 A full set for a join node

Let p be a **join** node with children $q \in I$ and $r \in I$. By definition $X_p = X_q = X_r$, since we are using a nice tree-decomposition.

Suppose we have a full set of characteristics at node q, $FS(q)$, and a full set of characteristics at node r, $FS(r)$. Recall definition 2.3. Note that $V_p \cap V_r = X_p$, and G_p is obtained from G_q and G_r by identifying the vertices of X_p in X_q and in X_r.

For a characteristic $(Z, \tau[a])$, $Z = (Z_{t_i})_{1 \leq i \leq w}$, $\tau[a] = (\tau(a^{(1)}), \ldots, \tau(a^{(w)}))$, define the list $[a^*] = (\tau(a^{(1)}) - |Z_{t_1}|, \ldots, \tau(a^{(w)}) - |Z_{t_w}|)$.

Theorem 4.3 *Let $FS(q)$ $(FS(r))$ be a full set of characteristics at node q (r), q and r the children of* **join** *node p. Then*

$$\begin{aligned} FS(p) = \quad \{(Z, \tau[c]) \mid \quad & (Z, \tau[a]) \in FS(q) \wedge (Z, \tau[b]) \in FS(r) \wedge \\ & [c] \in [a^*] \oplus \tau[b] \wedge \max([c]) \leq k+1\} \end{aligned}$$

is a full set of characteristics for p.

Proof: We prove this theorem with help of some intermediate results. We first show that an element of $FS(p)$ is indeed a characteristic of a partial path-decomposition at node p. The following lemmas will be useful.

Lemma 4.4 *Let Y be a partial path-decomposition. Let $[y]$ be the list of Y. If $[y^*] \in E[y]$, then there exists a partial path-decomposition $Y^* \in E(Y)$ with list $[y^*]$.*

Proof: If an element of $y_i^{(u)}$ is repeated, we repeat the corresponding subset $Y_i^{(u)}$ the same number of times. □

Lemma 4.5 *Let Y be a partial path-decomposition rooted at p with characteristic $(Z, \tau[y])$. Then there exists a partial path-decomposition Y' rooted at p* with the same characteristic *such that the list $[y']$ of Y' satisfies $[y'] \in E(\tau[y])$.*

Proof: Assume that no integer sequence $y^{(u)}$ of $[y]$ has two consecutive elements that are the same. Recall the proof of Lemma 3.9. Consider a typical operation applied to an integer sequence $y^{(u)}$ of the list $[y]$ of Y. Suppose subsequence $y^{(u)}(i+1 \dots j-1)$ is removed. W.l.o.g., suppose $y_i^{(u)} \geq y_j^{(u)}$. We write $Y_k^{(u)}$ for the set of the partial path-decomposition Y, corresponding to $y_k^{(u)}$. We obtain a path-decomposition Y^* as follows: initialize $Y_k^* = Y_k$ for all sets Y_k of the path-decomposition Y. Iteratively for $k = i+1, \dots, j-1$, add elements of $Y_{k-1}^{*(u)} - Y_k^{*(u)}$ to the set $Y_k^{*(u)}$ until $Y_k^{*(u)}$ contains $|Y_i^{*(u)}|$ elements. It is easy to see that Y^* is a partial path-decomposition with the same characteristic, and that the list $[y^*]$ of Y^* satisfies $[y^*] \in E(\tilde{y})$. The lemma now follows with induction on the number of typical operations. □

Definition 4.7
Let p be a **join** node with children q and r. Let A be a path-decomposition rooted at q and let B be a path-decomposition rooted at r, such that the restrictions of A and B are the same. Then we write $C = A \cup B$ for the path-decomposition rooted at p obtained by $C_i = A_i \cup B_i$ for all i.

Lemma 4.6 *Let p be a* **join** *node with children q and r. Let A (B) be a partial path-decomposition rooted at q (r), such that the restrictions of A and b are the same. Then $C = A \cup B$ is a partial path-decomposition rooted at p.*

Proof: This follows directly from the definitions. □

In the next three results we assume $FS(p)$ is computed from full sets of characteristics $FS(q)$ and $FS(r)$ as described in this theorem.

Lemma 4.7 *Let p be a* **join** *node with children q and r. For each $(Z, \tau[c]) \in FS(p)$ there is a partial path-decomposition rooted at p with this characteristic.*

Proof: Let A be a partial path-decomposition at q with characteristic $(Z, \tau[a]) \in FS(q)$ and let B be a partial path-decomposition at r with characteristic $(Z, \tau[b]) \in FS(r)$ with the same interval model Z. By Lemma 4.5 we may assume that the lists $[a]$ of A and $[b]$ of B satisfy $[a] \in E(\tau[a])$ and $[b] \in E(\tau[b])$. Define

$$[a'] = (a^{(1)} - |Z_{t_1}|, \dots, a^{(w)} - |Z_{t_w}|) \wedge [a^*] = (\tau(a^{(1)}) - |Z_{t_1}|, \dots, \tau(a^{(w)}) - |Z_{t_w}|)$$

Clearly $[a'] \in E[a^*]$ since $[a] \in E(\tau[a])$. Let $[c] \in [a^*] \oplus \tau[b]$ with $max([c]) \leq k+1$. By Lemma 3.21.5 we may conclude that there is a list $[c^\circ] \in E[c]$ such that $[c^\circ] \in [a'] \oplus [b]$. Hence there are extensions $[a^\circ] \in E[a']$ and $[b^\circ] \in E[b]$ such that $[c^\circ] = [a^\circ] + [b^\circ]$. Notice that since $[c^\circ] \in E[c]$ also $max([c^\circ]) \leq k+1$.

Now take extensions $A^\circ \in E(A)$, corresponding with the extension $[a^\circ]$, and $B^\circ \in E(B)$ corresponding with $[b^\circ]$. Define $C^\circ = A^\circ \cup B^\circ$ (since $[a^\circ]$ and $[b^\circ]$ have the same length in the strong sense C° is well defined). By Lemma 4.6 C° is a partial path-decomposition rooted at p. The list of C° is $[c^\circ]$ and hence C° has width at most k. Finally, since $[c^\circ] \in E[c]$: $\tau[c^\circ] = \tau[c]$ (Lemma 3.21.2). Hence the characteristic of C° is $(Z, \tau[c]) \in FS(p)$. □

Lemma 4.8 *Let p be a* **join** *node with children q and r. If Y is a partial path-decomposition rooted at p of width at most k then there is a partial path-decomposition $Y' \prec Y$ such that $C(Y') \in FS(p)$.*

Proof: Let A be the sub-decomposition of Y for G_q and let B be the sub-decomposition of Y for G_r, so $Y = A \cup B$. Since $FS(q)$ and $FS(r)$ are full set of characteristics, we know there exist path-decompositions $A_0 \prec A$ for G_q of which the characteristic is in $FS(q)$ and $B_0 \prec B$ for G_r of which the characteristic is in $FS(r)$. By Lemma 4.5 there exists also a partial path-decomposition A' with the same characteristic as A_0, such that $[a'] \in E(\tau[a'])$. Notice that

$$[a'] \equiv \tau[a'] = \tau[a_0] \equiv [a_0] \prec [a]$$

hence $A' \prec A$. In the same manner we find a partial path-decomposition $B' \prec B$ such that $[b'] \in E(\tau[b'])$. Notice that the interval model of all these path-decompositions is the same, say $(Z_{t_i})_{1 \leq i \leq w}$. Define the list

$$[y^*] = (y^{(1)} + |Z_{t_1}|, \ldots, y^{(w)} + |Z_{t_w}|)$$

(where $[y]$ is the list of Y). Then we have $[y^*] = [a] + [b]$. By Lemma 3.21.6 there exists a list $[y^\circ] \in [a'] \oplus [b']$ such that $[y^\circ] \prec [y^*]$. Hence there are extensions $[a^\circ] \in E[a']$ and $[b^\circ] \in E[b']$ such that $[y^\circ] = [a^\circ] + [b^\circ]$. By Lemma 4.4 there are path-decompositions $A^\circ \in E(A')$ with list $[a^\circ]$ and $B^\circ \in E(B')$ with list $[b^\circ]$. Define $Y^\dagger = A^\circ \cup B^\circ$. Notice that $Y^\dagger$ is a partial path-decomposition rooted at p with list

$$[y^\dagger] = (y^{\circ(1)} - |Z_{t_1}|, \ldots, y^{\circ(w)} - |Z_{t_w}|)$$

Notice that $[y^\dagger] \prec [y]$ (since $[y^\circ] \prec [y^*]$), hence $Y^\dagger \prec Y$. Since $[a'] \in E(\tau[a'])$ and $[a^\circ] \in E[a']$: $[a^\circ] \in E(\tau[a'])$. Also $[b^\circ] \in E(\tau[b'])$. Hence $[y^\circ] \in \tau[a'] \oplus \tau[b']$. If we define

$$[a'^*] = (\tau(a'^{(1)}) - |Z_{t_1}|, \ldots, \tau(a'^{(w)}) - |Z_{t_w}|)$$

we find $[y^\dagger] \in [a'^*] \oplus \tau[b']$, hence $C(Y^\dagger) \in FS(p)$. □

This proves theorem 4.3. □

Note that theorem 4.3 implies that a full set of characteristics for a join node p can be computed in $O(1)$ time, given the full sets of children p and q.

4.4 A full set for a forget node

Let p be a **forget** node with child q. Then $G_p = G_q$ and by Definitions 2.6 and 2.7 $X_p \subset X_q$ and X_q contains exactly one vertex, say x, which is not in X_p. We call x the *forgotten* element of p. We first show how to compute the full set of characteristics $FS(p)$ from the full set of characteristics $FS(q)$.

Let $(Z, \tau[y])$ be a characteristic in $FS(q)$, with interval model $Z = (Z_{t_j})_{1 \leq j \leq w}$. Since Z is a path-decomposition for the subgraph induced by X_q there is a consecutive number of subsets in Z which contain the forgotten element x. We remove x from these sets, and remove consecutive subsets which are now the same. Obviously, the following lemma holds.

Lemma 4.9 *Z' is an interval model for p.*

Let $i \leq j$ be such that Z_{t_i} is the first subset Z which contains x and Z_{t_j} is the last subset containing x. Notice that the number of subsets in Z is at most two more than the number of subsets of Z', namely Z_{t_i} can become the same as $Z_{t_{i-1}}$ after the removal of x and Z_{t_j} can become the same as $Z_{t_{j+1}}$. Consider the following four cases.

1. If the number of subsets of Z' is the same as the number of subsets in Z, then we put $(Z', \tau[y])$ in $FS(p)$.
2. If only the subset $Z_{t_i} \setminus \{x\}$ is the same as $Z_{t_{i-1}}$ then let
$$\tau^* = \tau(\circ\tau(y^{(i-1)})\tau(y^{(i)}))$$
and change the typical list $\tau[y]$ into the list:
$$[y'] = (\tau(y^{(1)}), \ldots, \tau(y^{(i-2)}), \tau^*, \tau(y^{(i+1)}), \ldots, \tau(y^{(w)}))$$
i.e. we concatenate the typical sequences $\tau(y^{(i-1)})$ and $\tau(y^{(i)})$ and compute the typical sequence of the result. We put $(Z', [y'])$ in $FS(p)$.
3. If only $Z_{t_j} \setminus \{x\} = Z_{t_{j+1}}$ then compute
$$\tau(\circ\tau(y^{(j)})\tau(y^{(j+1)}))$$
and change the typical list $\tau[y]$ into $[y'']$ as in the former case. Put $(Z', [y''])$ into $FS(p)$.
4. Finally, if both $Z_{t_{i-1}} = Z_{t_i} \setminus \{x\}$ and $Z_{t_j} \setminus \{x\} = Z_{t_{j+1}}$ then let
$$\tau_1 = \tau(\circ\tau(y^{(i-1)})\tau(y^{(i)})) \wedge \tau_2 = \tau(\circ\tau(y^{(j)})\tau(y^{(j+1)}))$$
and change the typical list $\tau[y]$ into the list:
$$[y^*] = (\tau(y^{(1)}), \ldots, \tau(y^{(i-2)}), \tau_1, \tau(y^{(i+1)}), \ldots$$
$$\ldots, \tau(y^{(j-1)}), \tau_2, \tau(y^{(j+2)}), \ldots, \tau(y^{(w)}))$$
We put $(Z', [y^*])$ in $FS(p)$.

Notice that if $i = j$ we compute in this last case the typical sequence of: $\circ\tau(y^{(i-1)})\circ\tau(y^{(i)})\tau(y^{(i+1)})$.

$FS(p)$ is obtained by carrying out the above for each element of $FS(q)$. Below we assume $FS(p)$ is computed in this way from $FS(q)$. To prove the correctness we first show that an element of $FS(p)$ is a characteristic of a partial path-decomposition rooted at p.

Theorem 4.10 *For each $(Z', [c]) \in FS(p)$ there is a partial path-decomposition rooted at p with this characteristic.*

Proof: Let $(Z, \tau[y])$ be the corresponding characteristic in $FS(q)$ (i.e. $(Z', [c])$ is computed from $(Z, \tau[y])$ by the algorithm described above). There exists a partial path-decomposition Y rooted at q, with this characteristic. Y is also a partial path-decomposition rooted at p. By Lemma 4.9 the interval model of Y at node p is Z'. We prove that the typical list is computed correctly. This is clearly the case when $Z' = Z$. Consider the second case: the number of subsets in Z' is one less and $Z_{t_{i-1}} = Z_{t_i} \setminus \{x\}$. Let $[y]$ be the list of Y. In this case the list of Y changes into

$$(y^{(1)}, \dots, y^{(i-2)}, \circ y^{(i-1)}y^{(i)}, y^{(i+1)}, \dots, y^{(w)})$$

By Lemma 3.17: $\tau(\circ y^{(i-1)}y^{(i)}) = \tau(\circ\tau(y^{(i-1)})\tau(y^{(i)}))$, hence the typical list is computed correctly. The other cases are similar. □

The next theorem shows that $FS(p)$ is indeed a full set of characteristics.

Theorem 4.11 *If Y is a partial path-decomposition rooted at p of width at most k, then there is a partial path-decomposition $Y' \prec Y$ such that $C(Y') \in FS(p)$.*

Proof: Y is also a partial path-decomposition rooted at q, since $G_q = G_p$. Hence there is a partial path-decomposition rooted at q, $Y' \prec Y$, of which the characteristic is in $FS(q)$. In the proof of Theorem 4.10 it is shown that the characteristic of Y' is computed correctly for node p. We only have to show that $Y' \prec Y$ still holds for node p (recall Definition 4.5: the interval model may have changed!). This however is proved in Lemma 3.19. □

Corollary 4.12 *$FS(p)$ is a full set of characteristics for the* **forget** *node p.*

So, the full set of a forget node can be computed in $O(1)$ node, given the full set of its child.

4.5 A full set for an introduce node

In this subsection, we consider the case in which p is an **introduce** node with child q. Now $V_p = V_q \cup \{x\}$ for some vertex $x \notin V_q$. We call the vertex x *introduced* at p. Note that all neighbors of x in G_p belong to X_p.

Suppose we have a full set of characteristics $FS(q)$ for q. We give a procedure to compute a set $FS(p)$, and then we prove that this set $FS(p)$ is a full set of characteristics for p.

The computation of $FS(p)$ is certainly not the most efficient one possible, but is given here for a somewhat simpler presentation.

We first make a list of all feasible interval models for the for the node p (a minimal path-decomposition for the subgraph induced by X_p is a feasible interval model for p). We then check, for each feasible interval model, if there is a characteristic in $FS(q)$ which can be 'extended' to a characteristic for X_p with this interval model. Clearly, this algorithm might not be the most efficient one to compute $FS(p)$, since there could be many feasible interval models which are in fact not interval models.

Algorithm

Step 1 Make a list Q of all minimal path-decompositions for the subgraph induced by X_p (Definition 3.4).

Step 2 Make a new list Q^* as follows. For each minimal path-decomposition $Z \in Q$ compute Z': Remove the introduced vertex x from all subsets of Z and after that remove repetitions of subsets. Notice that Z' is a minimal path-decomposition for the subgraph induced by X_q. Put the pair (Z, Z') in Q^*.

Step 3 If for some pair $(Z, Z') \in Q^*$ there is *no* characteristic in the full set of characteristics for q, $FS(q)$, which has Z' as an interval model, then remove the pair (Z, Z') from the list Q^*.

Step 4 Initialize $FS(p) = \emptyset$. For each pair (Z, Z') in Q^* and for each characteristic $(Z', \tau[c]) \in FS(q)$ with Z' as an interval model do the following. Let $Z = (Z_{t_s})_{1 \leq s \leq w}$. Let $i \leq j$ be such that Z_{t_i} is the first subset of Z containing the introduced vertex x and Z_{t_j} the last subset of Z containing x. Notice that the number of subsets of Z is at most two more than the number of subsets of Z'. Namely, after the removal of x Z_{t_i} can become the same as $Z_{t_{i-1}}$ and Z_{t_j} can become the same as $Z_{t_{j+1}}$. Hence one of the following four different cases is applicable:

1. If the number of subsets of Z' is the same as the number of subsets in Z, we change the typical list $\tau[c]$ into:

$$\begin{aligned}[c^\circ] = (\tau(c^{(1)}), \ldots, \tau(c^{(i-1)}), 1 + \tau(c^{(i)}), \ldots \\ \ldots, 1 + \tau(c^{(j)}), \tau(c^{(j+1)}), \ldots, \tau(c^{(w)}))\end{aligned}$$

   i.e. we add one to all typical sequences $\tau(c^{(u)})$ with $i \leq u \leq j$. If $max([c^\circ]) \leq k + 1$, then we put $(Z, [c^\circ])$ in $FS(p)$.

2. If the number of subsets in Z' is one less than the number of subsets in Z, and $Z_{t_i} \setminus \{x\} = Z_{t_{i-1}}$: For convenience we write:

$$\tau[c] = (\tau(c^{(1)}), \ldots, \tau(c^{(i-2)}), \tau(c^{(i)}), \tau(c^{(i+1)}), \ldots, \tau(c^{(w)}))$$

   Consider *all splits of both types* (δ_1, δ_2) of $\tau(c^{(i)})$ (Definition 3.10). For each such split change the typical list $\tau[c]$ into:

$$\begin{aligned}[c'] = (\tau(c^{(1)}), \ldots, \tau(c^{(i-2)}), \delta_1, 1 + \delta_2, 1 + \tau(c^{(i+1)}), \ldots \\ \ldots, 1 + \tau(c^{(j)}), \tau(c^{(j+1)}), \ldots, \tau(c^{(w)}))\end{aligned}$$

   If $max([c']) \leq k + 1$ then we put $(Z, [c'])$ in $FS(p)$.

3. If the number of subsets in Z' is one less than the number of subsets in Z, and $Z_{t_j} \setminus \{x\} = Z_{t_{j+1}}$: This case is similar to the second case. In this case we make all splits of $\tau(c^{(j)})$.
4. If the number of subsets in Z' is two less than the number of subsets in Z: Let

$$\tau[c] = (\tau(c^{(1)}), \ldots, \tau(c^{(i-2)}), \tau(c^{(i)}), \ldots, \tau(c^{(j)}), \tau(c^{(j+2)}), \ldots, \tau(c^{(w)}))$$

In this case consider all splits (α_1, α_2) of $\tau(c^{(i)})$ and all splits (β_1, β_2) of $\tau(c^{(j)})$. For each pair of such splits (α_1, α_2) and (β_1, β_2) change the typical list $\tau[c]$ into

$$\begin{aligned}[c^\dagger] = (\tau(c^{(1)}), \ldots, \tau(c^{(i-2)}), \alpha_1, 1 + \alpha_2, 1 + \tau(c^{(i+1)}), \ldots \\ \ldots, 1 + \beta_1, \beta_2, \tau(c^{(j+2)}), \ldots, \tau(c^{(w)}))\end{aligned}$$

If $max([c^\dagger]) \leq k + 1$ then put $(Z, [c^\dagger])$ in $FS(p)$.

Notice that in the last case, if $i = j$ then we split $\tau(c^{(i)})$ into *three* parts; i.e first split it into two parts and then split the second part again.

Step 5 Stop. The computation of $FS(p)$ is completed.

We prove that $FS(p)$ is a full set of characteristics for p in two stages. First we demonstrate that every element in $FS(p)$ is a characteristic of a partial path-decomposition rooted at p.

Theorem 4.13 *For each $(Z, [d]) \in FS(p)$ there is a partial path-decomposition rooted at p with this characteristic.*

Proof: Let $(Z', \tau[y])$ be the corresponding characteristic in $FS(q)$, i.e. $(Z, [d])$ is computed from this characteristic by the algorithm described above. There is a partial path-decomposition Y rooted at q, with $C(Y) = (Z', \tau[y])$. Let $[y]$ be the list for Y. By Lemma 4.5 we may assume that $[y] \in E(\tau[y])$. Let $(Z', [Y])$ be the list representation of Y.

First consider the case with $|Z'| = |Z|$ (the same number of subsets). If $i \leq j$ are the first and last subset of Z containing x, we change the path-decomposition Y into Y°, by adding x to all subsets of $Y^{(u)}$, for all $i \leq u \leq j$. Clearly, Y° is a partial path-decomposition for G_p. Since $\tau(y^{\circ(u)}) = \tau(1 + y^{(u)}) = 1 + \tau(y^{(u)})$ for all $i \leq u \leq j$, the typical list $\tau[y^\circ]$ for Y° equals the list $[d]$ as computed by the algorithm. Hence Y° is a partial path-decomposition with characteristic $(Z, [d])$.

Now consider the case where the number of subsets in Z is one more than the number of subsets in Z' and $Z_{t_i} \setminus \{x\} = Z_{t_{i-1}}$ (the second case). In this case the typical sequence $\tau(y^{(i)})$ is split by the algorithm into, say, (δ_1, δ_2) in order to obtain $[d]$. The integer sequence $y^{(i)}$ is an extension of $\tau(y^{(i)})$. We make a split of $y^{(i)}$, say $(y_1^{(i)}, y_2^{(i)})$ such that $y_1^{(i)} \in E(\delta_1)$ and $y_2^{(i)} \in E(\delta_2)$. By Lemma 3.20 this split is always possible and $\tau(y_1^{(i)}) = \delta_1$ and $\tau(y_2^{(i)}) = \delta_2$. Take a similar 'split' of $Y^{(i)}$ into $(Y_1^{(i)}, Y_2^{(i)})$. Add the vertex x to all subsets of $Y_2^{(i)}$, and to all subsets of $Y^{(u)}$ with $i + 1 \leq u \leq j$. Call the obtained path-decomposition Y° (notice that Y° is indeed a partial path-decomposition rooted at p of width at most k). Since the typical sequence for $y_1^{(i)}$ is δ_1 and the typical sequence for $y_2^{(i)}$ is δ_2, the characteristic of Y° is indeed $(Z, [d])$, where $[d]$ is the typical list of Y° as computed by the algorithm.

The other two cases are similar. □

Theorem 4.14 *For every partial path-decomposition Y rooted at p of width at most k there exists a partial path-decomposition $Y' \prec Y$, such that $C(Y') \in FS(p)$.*

Proof: Let Y° be the sub-decomposition of Y for G_q. Since $FS(q)$ is a full set of characteristics, there exists a partial path-decomposition $Y_0 \prec Y^\circ$ of which the characteristic is in $FS(q)$. By Lemma 4.5 we know there exists a partial path-decomposition Y' with the same characteristic as Y_0, such that $[y'] \in E(\tau[y'])$ (where $[y']$ is the list of Y' and $\tau[y']$ is the typical list for Y'). Notice that $[y'] \equiv \tau[y'] = \tau[y_0] \equiv [y_0] \prec [y^\circ]$ (Lemma 3.21 and Definition 4.5) hence $Y' \prec Y^\circ$. So there are extensions $Y'^* \in E(Y')$ and $Y^{\circ *} \in E(Y^\circ)$ such that the respective lists satisfy $[y'^*] \leq [y^{\circ *}]$ (Lemma 4.4). Since Y and Y° have the same length we can take an extension $Y^* \in E(Y)$ corresponding with $Y^{\circ *}$ (i.e. if some subset Y_f° is repeated r times in $Y^{\circ *}$ then we repeat Y_f also r times in Y^*). Notice that now we have three partial path-decompositions of the same length Y^*, $Y^{\circ *}$ and Y'^* and that $Y^{\circ *}$ and Y'^* have the same interval model.

Make a partial path-decomposition $Y^\dagger$ rooted at p by changing Y'^* as follows. Add x to $Y_f'^*$ whenever $x \in Y_f^*$. *Notice that the interval model of $Y^\dagger$ is the same as the interval model of* Y and $[y^\dagger] \leq [y^*]$, hence $Y^\dagger \prec Y$.

We now show that the characteristic of $Y^\dagger$ is in $FS(p)$. Clearly, the characteristic of Y'^* is in $FS(q)$. Let $(Z', [Y'^*])$ be the list representation for Y'^* and let $(Z, [Y^\dagger]$ be the list representation for $Y^\dagger$. Write $Z = (Z_{t_s})_{1 \leq s \leq w}$. Let $i \leq j$ be the first and last interval of Z containing the vertex x.

Consider the case where the number of intervals of Z is the same as the number of intervals of Z'. Then x is added to *all* subsets of $Y'^{*(u)}$ for all $i \leq u \leq j$ (otherwise at least one interval of Z' would have been split). It follows that the characteristic of $Y^\dagger$ is computed in the first case of step 4 of the algorithm: $y^{\dagger(u)} = 1 + y'^{*(u)}$ for all $i \leq u \leq j$ hence $\tau(y^{\dagger(u)}) = 1 + \tau(y'^{*(u)})$.

Next consider the case where the number of intervals of Z' is one less than the number of intervals of Z and $Z_{t_i} \setminus \{x\} = Z_{t_{i-1}}$. For convenience we write:

$$[y'^*] = (y'^{*(1)}, \dots, y'^{*(i-2)}, y'^{*(i)}, \dots, y'^{*(j)}, y'^{*(j+1)}, \dots, y'^{*(w)})$$

So $y'^{*(i)} = \circ y^{\dagger(i-1)}(y^{\dagger(i)} - 1)$. Now $[y'^*] \in E(\tau[y'])$. Write $\tau(y'^{(i)}) = (\alpha_1, \dots, \alpha_s)$. Then it follows that either

$$y^{\dagger(i-1)} = (\alpha_1, \dots, \alpha_1, \dots, \alpha_f, \dots, \alpha_f) \ \wedge \ y^{\dagger(i)} = 1 + (\alpha_f, \dots, \alpha_f, \dots, \alpha_s, \dots, \alpha_s) \text{ or}$$
$$y^{\dagger(i-1)} = (\alpha_1, \dots, \alpha_1, \dots, \alpha_f, \dots, \alpha_f) \ \wedge \ y^{\dagger(i)} = 1 + (\alpha_{f+1}, \dots, \alpha_{f+1}, \dots, \alpha_s, \dots, \alpha_s)$$

It follows that the characteristic of $Y^\dagger$ is computed by the algorithm in the second case of step 4.

The other cases are similar. □

Corollary 4.15 *The set $FS(p)$ computed by the algorithm is a full set of characteristics for the* **introduce** *node p.*

Again, we have a procedure that computes a full set from a full set for the child node in $O(1)$ time.

4.6 The decision algorithm

From the previous sections, it now easily follows that we can decide whether 'pathwidth(G) $\leq k$', given the nice tree-decomposition of G of width $\leq \ell$, in time, proportionally to the number of nodes of this tree-decomposition, which is linear in the number of vertices of G. We compute for all nodes of T a full set of characteristics, starting at the leaves, and going up in the tree. The pathwidth of G is at most k, if and only if the full set for the root node is non-empty.

As we spend $O(1)$ time per node of T, the total time is linear in the number of nodes of T, i.e., linear in the number of vertices of G.

5 A decision algorithm for treewidth

In this section, we give an explicit algorithm, that given a graph $G = (V, E)$ with a nice tree-decomposition of G of width $\leq \ell$, decides whether the treewidth of G is at most k. The algorithm uses time, linear in the number of nodes of the nice tree-decomposition, which is $O(|V|)$.

The method we use is an extension of the method we used in the previous section for the pathwidth problem. We use one main new concept here: the *trunk* of the tree in the tree-decomposition (of width $\leq k$).

5.1 The characteristic of a tree-decomposition and full sets

Note that we work with two types of tree-decompositions: the nice tree-decomposition $NT = (\{X_i \mid i \in I\}, T = (I, F))$ of G of width $\leq \ell$, and the tree-decompositions of G or subgraphs of G of width $\leq k$. In this section, we define the characteristic of a tree-decomposition. First we note that we may restrict ourselves to tree-decompositions which are minimal in some sense.

Definition 5.1
Let $D = (S, T)$ be a tree-decomposition for a graph G. D is called *non-trivial* if for every pair of adjacent nodes x and y in T the corresponding subsets S_x and S_y are different.

Clearly, if D is a tree-decomposition, it can be transformed to a tree-decomposition which is non-trivial.

Definition 5.2
Let $D = (S, T)$ be a tree-decomposition. Let x be a leaf of T and let S_x be the corresponding subset. The leaf x is called *maximal* if S_x contains a vertex v which is not an element of any other subset of S.

Notice that if x is a leaf and if y is the father of x, then x is exactly maximal if S_x is *not* a subset of S_y.

Definition 5.3
Let $D = (S, T)$ be a tree-decomposition for a graph G. D is called *minimal* if the following two conditions are satisfied:

1. D is non-trivial, and
2. all leafs of T are maximal.

Lemma 5.1 *Let G be a graph with treewidth k. There exists a minimal tree-decomposition of G of width k.*

Proof: Take any tree-decomposition $D = (S, T)$ of G of width k. We transform D into a minimal tree-decomposition D' as follows. First, recursively remove leafs of T which are not maximal and remove the corresponding subsets from S. If $e = (x, y)$ is an edge of T such that $S_x = S_y$, then contract the edge in T and replace the subsets S_x and S_y by one new subset. It is easy to see that D' obtained in this way is a minimal tree-decomposition. □

We want to show an upper bound on the number of nodes in a minimal tree-decomposition.

Lemma 5.2 *Let G be a graph with n vertices. Then the number of nodes in a minimal tree-decomposition is at most $(2n-1)^2$.*

Proof: Let $D = (S, T)$ be a minimal tree-decomposition for a graph G with n vertices. For each leaf node $i \in I$, X_i must contain a vertex v_i that is not contained in the set X_j, j the neighbor of i in T, and hence is not contained in any set X_j, $j \in I$. So T has at most n leaves. Using standard arguments, it follows that the number of nodes of degree at least 3 in T is at most $n-1$. Since adjacent subsets are different, we can use Lemma 3.1 to see that each path in T with all vertices of degree two can have length at most $2n-1$. It follows that T has at most $(2n-2)(2n-1)$ vertices of degree two. Hence, the total number of nodes in T is at most $(2n-1)^2$. □

The node $i \in I$ in the definition below denotes a node in the nice tree-decomposition NT of G of width $\leq \ell$. Recall definition 2.3 of a rooted subgraph.

Definition 5.4
A *partial tree-decomposition* rooted at a node $i \in I$ is a tree-decomposition for G_i, i.e., the subgraph rooted at i.

Definition 5.5
Let $Y = (SY, TY)$ be a partial tree-decomposition rooted at a node i. The *restriction* of Y is the sub-decomposition $Y^* = (SY^*, TY)$ for the subgraph induced by X_i, i.e. if we define for each $S \in SY$, $S' = S \cap X_i$, then $SY^* = \{S' \mid S \in SY\}$.

The characteristic of a partial tree-decomposition consists of three parts. We call the first part the trunk of the tree-decomposition.

Definition 5.6
Let $Y = (SY, TY)$ be a partial tree-decomposition rooted at a node i. The *trunk* of Y is a tree $\mathcal{T}$ defined as follows. First take the restriction of Y, say $Y^* = (SY^*, TY)$. Next, recursively remove leafs of TY for which the corresponding subsets of SY^* are not maximal in SY^*. Finally, remove those vertices of the tree which have degree two and make the two neighbors adjacent. The *filled trunk* is the set of all trunk nodes and all nodes on a path between trunk nodes in TY.

Lemma 5.3 *Let Y be a partial tree-decomposition rooted at a node i. The number of vertices of the trunk of Y is at most $2k$.*

Proof: Every leaf node of the trunk of Y contains at least one unique vertex, not in any other trunk node. So the trunk has at most $k+1$ leaf nodes, and hence it cannot have more than $k-1$ nodes of degree ≥ 3. □

We now define the tree model of a partial tree-decomposition in analogue of the interval model defined in Definition 3.3. Let $Y = (SY, TY)$ be a partial tree-decomposition 3 at a node i and let $\mathcal{T}$ be the trunk. For each edge e in the trunk, consider the corresponding path in TY with all internal vertices of degree 2. Let SY_e be the corresponding subsets of SY. We use the notation Y_e for the pair (SY_e, TY_e). Notice that Y_e is a path-decomposition for the subgraph induced by the vertices in $\cup_{S \in SY_e} S$.

Definition 5.7
Let $Y = (SY, TY)$ be a partial tree-decomposition rooted at a node i. The *tree model* of Y is a pair

$$(\mathcal{T}, (Z_e)_{e \in \mathcal{T}})$$

where $\mathcal{T}$ is the trunk of Y and Z_e is the interval model of Y_e for each edge e of $\mathcal{T}$.

Recall Definition 4.1 of the list representation of of a path-decomposition.

Definition 5.8
Let Y be a partial tree-decomposition rooted at a node i. The *trunk-representation* is:

$$(\mathcal{T}, (Z_e)_{e \in \mathcal{T}}, ([Y_e])_{e \in \mathcal{T}})$$

where $(Z_e, [Y_e])$ is the list representation for Y_e (for each edge e in the trunk).

Recall Definition 4.3 of a typical list of a path-decomposition.

Definition 5.9
Let Y be a partial tree-decomposition rooted at a node i with tree model $(\mathcal{T}, (Z_e)_{e \in \mathcal{T}})$. The *characteristic* of Y is the triple:

$$C(Y) = (\mathcal{T}, (Z_e)_{e \in \mathcal{T}}, (\tau[y_e])_{e \in \mathcal{T}})$$

where $\tau[y_e]$ is the typical list of Y_e.

Notice that the characteristic is of constant size by Lemma 5.3 and Lemma 4.1.

Definition 5.10
Let Y and Z be two partial tree-decompositions rooted at the same node i, which have *the same tree model* (i.e., the same trunk and for each edge of the trunk the same interval model). Then $Y \prec Z$ if for *each edge e in the trunk* the corresponding lists satisfy $[y_e] \prec [z_e]$. If $Y \prec Z$ and $Z \prec Y$ then we write $Y \equiv Z$.

Recall Definition 4.6 for the full set of characteristics. We define the full set of characteristics for partial tree-decompositions.

Definition 5.11
A set of characteristics $FS(i)$ of partial tree-decompositions rooted at some node i of width at most k is called a *full set of characteristics* if for each partial tree-decomposition Y rooted at i and of width at most k either its characteristic is in $FS(i)$ or there is a partial tree-decomposition Y' with $Y' \prec Y$ and the characteristic of Y' is in $FS(i)$.

In the following sections we show how to compute a full set of characteristics for each node from the full sets of characteristics of the children of the node.

5.2 A full set for a start node

Again, we may assume that X_p contains one vertex, i.e., $X_p = \{v\}$ for some $v \in V$. Note from lemma 5.2 that a minimal tree-decomposition of G_p has one node. So, there is a unique, one node minimal tree-decomposition of G_p: $(\{X_{i_0}\}, T = (\{i_0\}, \emptyset))$, with $X_{i_0} = \{v\}$. So, it is straightforward to compute the full set of characteristics rooted at p in $O(1)$ time: this set contains only the characteristic of this minimal tree-decomposition.

5.3 A full set for a join node

Let p be a **join** node with children q and r. By definition, $X_p = X_q = X_r$. Suppose we have full sets of characteristics $FS(q)$ and $FS(r)$ for the nodes q and r. We give a procedure to compute a set $FS(p)$, and then prove that $FS(p)$ is a full set fro p.

For every pair $(\mathcal{T}, (Z_e)_{e \in \mathcal{T}}, (\tau[a_e])_{e \in \mathcal{T}}) \in FS(q)$ and $(\mathcal{T}, (Z_e)_{e \in \mathcal{T}}, (\tau[b_e])_{e \in \mathcal{T}}) \in FS(r)$, (i.e., with the same tree model), compute for every edge e of the trunk the list

$$[a_e^*] = (\tau(a_e^{(1)}) - |Z_e^{(1)}|, \tau(a_e^{(2)}) - |Z_e^{(2)}|, \ldots) \tag{1}$$

and the set

$$\theta(e) = \{\tau[c_e] \mid [c_e \in [a_e^*] \oplus \tau[b_e] \wedge \max([c_e]) \leq k + 1\}$$

Put all triples $(\mathcal{T}, (Z_e)_{e \in \mathcal{T}}, (d_e)_{e \in \mathcal{T}})$ with for all $e \in \mathcal{T}$, $d_e \in \theta(e)$, in the set $FS(p)$.

Clearly, these computations can be done in time, only depending on the sizes of $FS(q)$ and $FS(r)$, hence in constant time. We now prove that this construction indeed computes a full set of characteristics for p. The following notion of the join of two partial tree-decompositions will be useful.

Let p be a **join** node with children q and r. Let $A = (SA, TA)$ be a partial tree-decomposition rooted at q and let $B = (SB, TB)$ be a partial tree-decomposition rooted at r. Let $A' = (SA', TA)$ be the restriction of A and let $B' = (SB', TB)$ be the restriction of B. From the trees TA and TB recursively remove leafs which are not maximal in the restriction. Call these new trees TA^* and TB^*, and let SA^* and SB^* be those subsets of SA' and SB' corresponding with nodes in TA^* and TB^* respectively. Assume that:

$$(SA^*, TA^*) = (SB^*, TB^*) \tag{2}$$

We define a tree-decomposition C rooted at p as follows. Define a tree TC by taking the union of TA and TB and by *identifying* the nodes of TA^* and TB^*. For each node x of TC, define a subset SC_x as:

$$SC_x = \begin{cases} SA_x \cup SB_x & \text{if } x \in TA^* \\ SA_x & \text{if } x \in TA \setminus TA^* \\ SB_x & \text{if } x \in TB \setminus TB^* \end{cases}$$

Note that TA^* consists of the full trunk of TA. The same relation holds for TB^* and TB. The resulting tree-decomposition C is written as $C = A \cup B$. Note that $A \cup B$ is not always defined, (namely only when (2) holds). It follows, that when $C = A \cup B$, then the trunk of A, B and C are the same.

Lemma 5.4 *$C = A \cup B$ is a partial tree-decomposition rooted at p.*

Proof: The conditions are easily verified. □

Lemma 5.5 *Let p be a* **join** *node with children q and r. Let*

$$C = (\mathcal{T}, (Z_e)_{e \in \mathcal{T}}, (\tau[y_e])_{e \in \mathcal{T}}) \in FS(p)$$

There exists a partial tree-decomposition of width at most k and rooted at p with this characteristic.

Proof: Let

$$\begin{aligned} C(A) &= (\mathcal{T}, (Z_e)_{e \in \mathcal{T}}, (\tau[a_e])_{e \in \mathcal{T}}) \in FS(q) \\ C(B) &= (\mathcal{T}, (Z_e)_{e \in \mathcal{T}}, (\tau[b_e])_{e \in \mathcal{T}}) \in FS(r) \end{aligned}$$

Let C be obtained from $C(A)$ and $C(B)$ by the algorithm:

$$\forall_{e \in \mathcal{T}} \, [[y_e] \in [a_e^*] \oplus \tau[b_e] \wedge \max([y_e]) \leq \ell + 1]$$

where $[a_e^*]$ as defined in (1). It is easy to verify that the proof of Lemma 4.7 generalizes to obtain the following result. There exist partial tree-decompositions A° rooted at q and B° rooted at r with characteristics $C(A)$ and $C(B)$ respectively, such that $C^\circ = A^\circ \cup B^\circ$ is well defined and has characteristic C. □

Lemma 5.6 *Let p be a* **join** *node with children q and r. If Y is a partial tree-decomposition rooted at p of width at mist ℓ then there exists a partial tree-decomposition Y' such that $Y' \prec Y$ and such that $C(Y') \in FS(p)$.*

Proof: Let A be the sub-decomposition of Y for G_q and let B be the sub-decomposition for G_r. Notice that $\bar{Y} = A \cup B$ is well defined and satisfies $C(\bar{Y}) = C(Y)$. The result of Lemma 4.8 can easily be generalized and we obtain the following result. There exist partial tree-decompositions A° rooted at q and B° rooted at r, such that $C(A^\circ) \in FS(q)$, $C(B^\circ) \in FS(r)$ and $Y^\dagger = A^\circ \cup B^\circ$ is well defined and satisfies $Y^\dagger \prec \bar{Y}$. Moreover we may assume that for each edge of the trunk $[a^\circ_e] \in E(\tau[a^\circ_e])$ and $[b^\circ_e] \in E(\tau[b^\circ_e])$. Hence $[y^\dagger_e] \in [a^{\circ *}_e] \oplus \tau[b^\circ_e]$, with $[a^{\circ *}_e]$ computed from $\tau[a^\circ_e]$ as in (1). This proves the theorem. □

It follows that $FS(p)$ is a full set of characteristics for the **join** node p, which can be computed in $O(1)$ time, given $FS(p)$ and $FS(q)$.

5.4 A full set for a forget node

Let p be a **forget** node with child q, and let x be the 'forgotten' element i.e. $X_p = X_q - \{x\}$. Again, we first give a procedure to compute a set $FS(p)$, given a full set of characteristics $FS(q)$ for q, and then we prove that $FS(p)$ is a full set of characteristics for p.

For every $(\mathcal{T}, (Z_e)_{e\in\mathcal{T}}, (\tau[y_e])_{e\in\mathcal{T}})$ in $FS(q)$, do the following: Remove from all sets $Z^{(i)}_e$ the vertex x. Compute the new trunk $\mathcal{T}^*$. (See below.) Remove repetitions from the interval models Z_e and for each edge $e \in \mathcal{T}^*$ let Z^*_e be the new interval model. Finally, for each edge $e \in \mathcal{T}^*$ change the typical list $\tau[y_e]$ into $\tau[y^*_e]$ as described in section 4.4. Put $(\mathcal{T}^*, (Z^*_e)_{e\in\mathcal{T}^*}, (\tau[y^*_e])_{e\in\mathcal{T}^*})$ in $FS(p)$.

Clearly, this computation takes constant time. Consider an $(\mathcal{T}, (Z_e)_{e\in\mathcal{T}}, (\tau[y_e])_{e\in\mathcal{T}})$ from $FS(q)$ and its corresponding $(\mathcal{T}^*, (Z^*_e)_{e\in\mathcal{T}}, (\tau[y_e])_{e\in\mathcal{T}})$. Let $Y = (ST, TY)$ be the tree-decomposition, rooted at q, corresponding with $(\mathcal{T}, (Z_e)_{e\in\mathcal{T}}, (\tau[y_e])_{e\in\mathcal{T}})$.

Lemma 5.7 *If the trunk $\mathcal{T}^*$ differs from the trunk $\mathcal{T}$, then there is exactly one subset $Z^{(i)}_e$ that contains the vertex x, and at least one end node of e must be a leaf of $\mathcal{T}$.*

Proof: Consider how trunks $\mathcal{T}$ and $\mathcal{T}^*$ are made. Let $Y^* = (ST^*, TY)$ be the restriction of Y at node q, and let $Y^* - \{x\} = (ST^* - \{x\}, TY)$ be the restriction of Y at node p, i.e., remove x from every subset in SY^*. Remove leaves from TY for 0 the corresponding subsets of SY^* are not maximal in Y^*. Let TY^* be the resulting tree. If $\mathcal{T}^*$ differs from $\mathcal{T}$, then there must be a leaf node w in TY^* whose set $SY^*_w - \{x\}$ is not maximal in $Y^* - \{x\}$. As SY^*_w is maximal, SY^*_w contains a vertex z, not in the set $SY^*_{w'}$, with w' the neighbor of w in TY^*. Necessarily, $z = x$. So, the edge e in $\mathcal{T}$ with w as one end node is the only edge with $\exists i : x \in Z^{(i)}_e$, and moreover, only one such i exists. □

Lemma 5.8 *The tree model for Y at p is $(\mathcal{T}^*, (Z_e)_{e\in\mathcal{T}^*})$.*

Theorem 5.9 *For each $(\mathcal{T}^*, (Z^*_e)_{e\in\mathcal{T}^*}, (\tau[y^*_e])_{e\in\mathcal{T}^*}) \in FS(p)$ there is a partial tree-decomposition Y rooted at p with this characteristic.*

Proof: Let $(\mathcal{T}, (Z_e)_{e\in\mathcal{T}}, (\tau[y_e])_{e\in\mathcal{T}})$ be the corresponding characteristic in $FS(q)$. Then there is a partial tree-decomposition Y rooted at q with this characteristic. Then Y is also a partial tree-decomposition rooted at p since $G_p = G_q$. By Lemma 5.8, $(\mathcal{T}^*, (Z_e)_{e\in\mathcal{T}^*})$ is the tree model for Y at p. By Theorem 4.10 for each edge e in $\mathcal{T}^*$ the typical list $\tau[y^*_e]$ is computed correctly. □

Theorem 5.10 *If Y is a partial tree-decomposition rooted at p of width at most k then there exists a partial tree-decomposition $Y' \prec Y$ such that $C(Y') \in FS(p)$.*

Proof: Y is also a partial tree-decomposition rooted at q. Since $FS(q)$ is a full set of characteristics, there is a partial tree-decomposition Y' with $Y' \prec Y$ such that $C(Y') \in FS(q)$. It is shown that the characteristic of Y' is computed correctly for node p. We have to show that $Y' \prec Y$ holds for node p. Since Y and Y' have the same tree model at q, they also have the same tree model at p (see Lemma 5.8; the tree model at p is computed from the tree model at q). Using Lemma 3.19 it follows that for each edge $e \in \mathcal{T}^*$ we have $[y'_e] \prec [y_e]$. Hence $Y' \prec Y$ for node p. □

Corollary 5.11 *$FS(p)$ is a full set of characteristics for the forget node p.*

Again, we can compute the full set for a forget node in $O(1)$ time, given the full set for its child.

5.5 A full set for an introduce node

Let p be an **introduce** node with child q. Let x be the vertex introduced at p, i.e. $X_p = X_q \cup \{x\}$. We first give a procedure that computes a set $FS(p)$, given a full set $FS(q)$ for q, and then prove that $FS(p)$ is a full set of characteristics for p. For reasons of simplicity we apply the same method as in section 4.5: First we compute all minimal tree-decompositions for X_p. Notice that by Lemma 5.2 this can be done in constant time. Remove the vertex x from all subsets, obtaining a tree-decomposition for X_q. Next compute the tree model of this tree-decomposition. Check if there is a characteristic in $FS(q)$ with this tree model, and change this into a new characteristic for $FS(p)$.

Algorithm

Step 1 Make a list Q of all tree models of minimal tree-decompositions for the graph induced by X_p of width at most k.

Step 2 Make a new list Q' as follows. For each element T^* of Q remove x from all subsets. Compute the tree model $T = (\mathcal{T}, (Z_e)_{e \in \mathcal{T}})$ of the result. If there is a characteristic in $FS(q)$ with this tree model, then put the pair (T^*, T) in the list Q'.

Step 3 Let $(T^*, T) \in Q'$, and let $C = (\mathcal{T}, (Z_e)_{e \in \mathcal{T}}, ([\tau[y_e]])_{e \in \mathcal{T}}) \in FS(q)$ such that $T = (\mathcal{T}, (Z_e)_{e \in \mathcal{T}})$. Let $\mathcal{T}^*$ be the trunk of T^*. We show how to compute characteristics $C^* = (\mathcal{T}^*, (Z^*_e)_{e \in \mathcal{T}^*}, (\tau[y^*_e])_{e \in \mathcal{T}^*})$ for $FS(p)$. We consider two cases.

1. The trunks $\mathcal{T}$ and $\mathcal{T}^*$ are the same. In this case we can proceed as described in section 4.5. For each edge of the trunk we change the typical list as described in step 4 of the algorithm given in section 4.5.
2. The trunks are different. In that case, the trunk $\mathcal{T}^*$ contains a leaf a, which is not a leaf of the trunk $\mathcal{T}$. Let b be the neighbor of a in $\mathcal{T}^*$. In general, when b is of degree three in $\mathcal{T}^*$, b is not a node in $\mathcal{T}$. In that case let c and d be the other neighbors of b in $\mathcal{T}^*$. Notice that c and d are adjacent in $\mathcal{T}$. Let Z_b be the interval corresponding with node b in T^*. Let

$$Z_e = (Z_e^{(c)}, \ldots, Z_e^{(b)}, \ldots, Z_e^{(d)})$$

be the interval model for $e = (c, d)$ in T. In T^* this interval model is split in two parts

$$\begin{array}{ll} Z^*_{e_1} = (Z^{(c)}_{e_1}, Z^{(c+1)}_{e_1}, \ldots, Z^{(b)}_{e_1}) & \text{for } e_1 = (c, b) \\ Z^*_{e_2} = (Z^{(b)}_{e_2}, Z^{(b+1)}_{e_2}, \ldots, Z^{(d)}_{e_2}) & \text{for } e_2 = (b, d) \end{array}$$

Consider the typical list for e in T:

$$\tau[y_e] = (\tau(y^{(c)}_e), \ldots, \tau(y^{(b)}_e), \ldots, \tau(y^{(d)}_e))$$

The typical sequence

$$\tau(y^{(c)}_e) = (\alpha_1, \ldots, \alpha_s)$$

for the interval $Z^{(b)}_e$, is split into two parts *in all possible ways* (each split gives a characteristic for $FS(p)$):

$$\tau_1 = (\alpha_1, \ldots, \alpha_f) \wedge \tau_2 = (\alpha_f, \ldots, \alpha_s)$$

Notice that, by Lemma 3.20, τ_1 and τ_2 are typical sequences. The typical lists for the edges (c, b) and (b, d) in C^* are

$$\begin{array}{ll} \tau[y^*_{e_1}] = (\tau(y^{(c)}_e), \ldots, \tau(y^{(b-1)}_e), \tau_1) & \text{for the edge } (c, b) \\ \tau[y^*_{e_2}] = (\tau_2, \tau(y^{(b+1)}_e), \ldots, \tau(y^{(d)}_e)) & \text{for the edge } (b, d) \end{array}$$

When the node b is a node in $\mathcal{T}$, the interval model and typical list are not split.

Finally, we have to describe the typical sequence for the edge (a, b) in $f = (a, b)$. Consider the interval model $Z_f = (Z^{(1)}_f, \ldots, Z^{(r)}_f)$ for this edge in T. The typical sequence $\tau(y^{*(i)}_f) = (|Z^{(i)}_f|)$ (i.e. consists of one element).

Step 5 Stop. The computation of $FS(p)$ is completed.

In Step 3 of the algorithm, we claim that the trunks $\mathcal{T}$ and $\mathcal{T}^*$ differ only in some specified way. We start by proving this.

Lemma 5.12 *If the trunks $\mathcal{T}$ and $\mathcal{T}^*$ are different, then there is exactly one leaf a of $\mathcal{T}^*$ which is not a leaf of $\mathcal{T}$. Let b be the neighbor of a in $\mathcal{T}^*$. If b is of degree three in $\mathcal{T}^*$ then b is not a node in $\mathcal{T}$. In this case the two other neighbors of b are adjacent in $\mathcal{T}$.*

Proof: Let Y^* be a minimal tree-decomposition for the subgraph induced by X_p with tree-model $(\mathcal{T}^*, (Z_e)_{e \in \mathcal{T}})$. The trunk $\mathcal{T}$ is obtained by removing x from all subsets and then computing the trunk of the result. Assume the trunks are not the same. Then clearly, there must be a leaf in $\mathcal{T}^*$ which is not a node of $\mathcal{T}$. Hence the subset corresponding with a is not maximal in $\mathcal{T}$. It follows that x is contained only in this subset. □

Theorem 5.13 *Each element $C^* = (\mathcal{T}^*, (Z^*_e)_{e \in \mathcal{T}^*}, (\tau[y^*_e])_{e \in \mathcal{T}^*}) \in FS(p)$ is the characteristic of a partial tree-decomposition rooted at node p.*

Proof: Let $C = (\mathcal{T}, (Z_e)_{e \in \mathcal{T}}, (\tau[y_e])_{e \in \mathcal{T}})$ be the characteristic in $FS(q)$ from which C^* is computed by the algorithm. Since $FS(q)$ is a full set of characteristics for q, there exists a partial tree-decomposition $Y = (SY, TY)$ rooted at q with characteristic C. By Lemma 4.5 we may assume that $[y_e] \in E(\tau[y_e])$ for every edge $e \in \mathcal{T}$. We show how to compute a partial tree-decomposition rooted at p with characteristic C^*. If $\mathcal{T} = \mathcal{T}^*$ the result follows from Theorem 4.13. Hence assume the trunks are different. Let a be the leaf of $\mathcal{T}^*$ which is not in $\mathcal{T}$ and let b be the neighbor of a. Assume b is not an element of $\mathcal{T}$. Then b has two other neighbors c and d which are adjacent in $\mathcal{T}$. Let e be the edge (c,d) in $\mathcal{T}$. The algorithm splits the characteristic sequence $\tau(y_e^{(b)})$ into two parts:

$$\tau_1 = (\alpha_1, \ldots, \alpha_f) \wedge \tau_2 = (\alpha_f, \ldots, \alpha_s)$$

Since $[y_e] \in E(\tau[y_e])$ we know that $y_e^{(c)} \in E(\tau(y_e^{(c)}))$. Hence we can split the sequence $Y_e^{(c)}$ in two parts $Y_{e_1}^*$ and $Y_{e_2}^*$ such that τ_1 is the characteristic sequence of $y_{e_1}^*$ and τ_2 is the characteristic sequence of $y_{e_2}^*$. Let b_0 be the node in TY, corresponding with the split of $Y_e^{(c)}$. We make a new tree TY^* by making a new path P adjacent to b_0. Let f be the edge (a,b) in $\mathcal{T}^*$. Each node i in P corresponds to a subset $Z_f^{*(i)}$. The corresponding subset in Y^* is equal to $Z_f^{*(i)}$. It is easily checked that Y^* is a partial tree-decomposition rooted at p with characteristic C^*. □

Theorem 5.14 *Let Y be a partial tree-decomposition rooted at p of width at most k. There exists a partial tree-decomposition Y' such that $Y' \prec Y$ and such that $C(Y') \in FS(p)$.*

Proof: We may assume Y is minimal. Let Y^0 be the sub-decomposition of Y rooted at q. Since $FS(q)$ is a full set of characteristics there exists a partial tree-decomposition $Y_0 \prec Y^0$ such that $C(Y_0) \in FS(q)$. Clearly we may assume that Y_0 is minimal. Let $C(Y_0) = (\mathcal{T}, (Z_e)_{e \in \mathcal{T}}, (\tau[y_e])_{e \in \mathcal{T}})$. We may assume that $[y_e] \in E(\tau[y_e])$ for all edges $e \in \mathcal{T}$. Since $Y_0 \prec Y^0$, there exist partial tree-decompositions Y_0^* and Y^{0*} such that $Y_{0e}^* \in E(Y_{0e})$, $Y_e^{0*} \in E(Y_e^0)$ and $[y_{0e}^*] \leq [y_e^{0*}]$ for every edge $e \in \mathcal{T}$. Since Y^0 is the restriction of Y, there exists a partial tree-decomposition Y^* with the same characteristic as Y such that $[y_e^*] \in E([y_e])$ and such that $[y_e^*]$ and $[y_e^{0*}]$ have the same length in the strong sense for every edge $e \in \mathcal{T}$. Assume the trunk of Y and Y^0 are different. Since Y and Y_0 are minimal there is a simple path P in the tree TY of Y which is not present in the tree TY_0 of Y_0. Let i be the node in TY_0^* which is adjacent to a node of P in TY^*. Make P adjacent to i in TY_0. □

We now have shown that $FS(p)$, which can be computed in $O(1)$ time from $FS(q)$ is a full set of characteristics for p.

5.6 The decision algorithm

Again, it now directly follows that we can decide in $O(n)$ time whether the treewidth of input graph G is at most k, given a nice tree-decomposition of G of width $\leq \ell$. This is done similarly as for the pathwidth problem, cf. section 4.6.

6 Turning decision algorithms into construction algorithms

In the previous sections, we showed how to obtain decision algorithms for the 'pathwidth $\leq k$' and 'treewidth $\leq k$' problems. We now show that also, if existing, corresponding path- and tree-decompositions of width $\leq k$ can be constructed, in linear time.

The first step of the construction algorithms is to run the decision algorithms. Clearly, if these output that the treewidth or pathwidth of the input graph is larger than k, then we are done. Otherwise, the full set of characteristics of the root node of the nice tree-decomposition is non-empty. Take an arbitrary characteristic from the full set of the root node.

We will describe a recursive procedure, that for a characteristic c^p from a full set of characteristics at node p computes a certain representation of a tree-decomposition or path-decomposition $(\{Y_i \mid i \in I\}, T = (I,F))$ of G_p with treewidth $\leq k$ and with characteristic c^p, and some pointers between c^p and this implicit representation.

To be more specific, we have the following representation and pointers for path-decompositions $(Y_1, \ldots, Y_r)$:

- for each $i \in I$, we have two sets L_i, R_i, with
  - $L_i = \{v \in Y_i \mid i \text{ is the first node with } v \in Y_i\}$.
  - $R_i = \{v \in Y_i \mid i \text{ is the last node with } v \in Y_i\}$.
- If $c^p = ((Z_{t_j})_{1\leq j\leq q}, \tau(y^{(j)})_{1\leq j\leq q})$, we have
  - for each j, $1 \leq j \leq q$, a pointer from the appearance from Z_{t_j} in c^p to the first and last nodes $i \in I$ with $X_p \cap Y_i = Z_{t_j}$.
  - for each j, $1 \leq j \leq q$, each each j', $1 \leq j' \leq l(\tau(y^{(j)}))$, a pointer from the integer variable $\tau(y^{(j)}))_i$ to the corresponding set Y_α (with $|Y_\alpha| = \tau(y^{(j)}))_i$, cf. section 4.1.
- the pairs (L_i, R_i) are put in a doubly linked list, in the order in which the nodes i appear in the path-decomposition $(Y_1, \ldots, Y_r)$.

Note that the collection of L_i's, R_i's and their order exactly determine the path-decomposition $(Y_1, \ldots, Y_r)$.

For tree-decompositions $(\{Y_i \mid i \in I\}, T = (I,F))$, we have the following representation:

- For each edge (a,b) in the trunk $\mathcal{T}$, we have for the set of nodes on the path between a and b in $\mathcal{T}$ a representation, similar as for path-decompositions, and we have similar pointers between these nodes and $(Z_{(a,b)}, \tau[y_{(a,b)}])$ (cf. definition 5.9).
- we have pointers from trunk nodes where appearing in c^p, to the corresponding nodes i in the representation of the tree-decomposition.
- for all nodes $i \in I$, not on the filled trunk, we have a set Y_i.
- each node, either on a path between trunk nodes, or not on the filled trunk, also carries a linked list of pointers to neighboring nodes, such that for every pair of adjacent nodes, at least one of them has a pointer to the other in this linked list.
- we have a bag, containing a pointer to each node $i \in I$.

Note that again, the information above is sufficient to construct the tree-decomposition, in $O(\sum_{i\in I} |X_i|) = O(n)$ time.

We now give a recursive procedure to compute this representation for a characteristic c^p, rooted at p. Again, we consider four cases, depending on the type of node of p.

p is a start node. This case is obvious.

p is a forget node. Note that during the computation of the full set of characteristics for p, the characteristic c^p is made from one unique characteristic c^q from the child node q of p. Moreover, a path- or tree-decomposition with characteristic c^q is also a path- or tree-decomposition with characteristic c^p. Recursively, compute the representation of such a path- or tree-decomposition for c^q. Then, we update the representation. When the trunk is changed, the forgotten vertex x must be placed in a number of subsets Y_i with i no longer a part of the trunk. The total time over all forget nodes of this work is bounded by $O(\sum_{i \in I} |Y_i|) = O(n)$. It is easy to see that all other modifications can be made in $O(1)$ time per forget node.

p is an introduce node. Again, the characteristic c^p is made from a unique characteristic c^q from the child node q of p. Recursively, compute the representation of the path- or tree-decomposition corresponding to c^q. From sections 4.5 and 5.5 it follows that the path- or tree-decomposition can be modified in a rather straightforward way to a path- or tree-decomposition with characteristic c^p. (Checking all the details is easy, but tedious, and is omitted here.) The modifications to the representations can be done in $O(1)$ time. (E.g., note that the representations with sets L_i, R_i make that (for every edge of the trunk) we must add x to one set L_i and one set R_i. In the case that the trunk changes, the interval model for the new trunk edge (a, b) (cf. section 5.5) $(Z_f^{(1)}, \ldots, Z_f^{(r)})$ equals the sequence of the new sets Y_i in the tree-decomposition, corresponding to this edge, and necessarily, $r \leq k + 1$.)

p is a join node Suppose q and r are the children of p. Observe that c^p is made from one characteristic c^q at q and one characteristic c^r at r.

First, we look at the pathwidth problem. We can write: $c^p = (Z, \tau[c])$, $(Z, \tau[a]) \in FS(q)$, $(Z, \tau[b]) \in FS(r)$, $[c] \in [\tau[a] - Z] \oplus \tau[b]$.

Recursively, compute the representations of the path-decompositions with characteristic $(Z, \tau[a])$, $(Z, \tau[b])$. As in the proof of lemma 4.7, we can now compute the representation for a path-decomposition with characteristic c^p. Again, because of the representation by sets L_i, R_i, $O(1)$ pointer modifications are sufficient to do this.

In case of the treewidth problem, we do the same as described above for each trunk edge. In some cases, we must link two linked lists of pointers to one linked list, which also costs $O(1)$ pointer operations.

The total work to compute a representation for a tree-decomposition with characteristic in the full set of the root node is linear in the sum of $|V|$ and the number of nodes in the given nice tree-decomposition of G of width $\leq \ell$. Note that this tree-decomposition is a tree-decomposition of G of width $\leq k$.

Theorem 6.1 *(i) For all $k, \ell \geq 1$, there exists an algorithm, that when given a graph $G = (V, E)$ and a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G of width $\leq \ell$, computes whether the treewidth of G is at most k, and if so, finds a tree-decomposition of G of width $\leq k$, and that uses $O(|V| + |I|)$ time.*
(ii) For all $k, \ell \geq 1$, there exists an algorithm, that when given a graph $G = (V, E)$ and a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G of width $\leq \ell$, computes whether the

pathwidth of G is at most k, and if so, finds a path-decomposition of G of width $\leq k$, and that uses $O(|V| + |I|)$ time.

Reed [37] has found an $O(n \log n)$ algorithm, that when given a graph $G = (V, E)$, either decides that the treewidth of G is at most k, or finds a tree-decomposition of G of width $O(k)$ (for fixed k). Combining this result with theorem 6.1 directly gives $O(n \log n)$ algorithms for the 'treewidth $\leq k$' and 'pathwidth $\leq k$' problems.

7 Computing the pathwidth of graphs with bounded treewidth

In this section, we show that the algorithms, given in this paper can also be used to compute the pathwidth of graphs with bounded treewidth in polynomial time. So, we assume a fixed upper bound on the treewidth of the input graphs, but the pathwidth of the input graphs, which must be computed, is not a-priory bounded by some constant. Most important for our discussions here are lemma 3.5, and the following fact.

Theorem 7.1 *[15] Let $G = (V, E)$ be a graph with treewidth $\leq \ell$. Then the pathwidth of G is at most $(\ell + 1) \log |V|$.*

In the remainder of this section, we suppose that ℓ is a fixed constant, and we want to decide whether a given graph $G = (V, E)$ with treewidth at most ℓ has pathwidth at most a given integer k. By theorem 7.1, we may assume that $k \leq (l + 1) \log |V|$, otherwise the problem is trivially solvable. We write $n = |V|$.

The first step of our algorithm is to find a nice tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of G with treewidth $\leq k$. Clearly, this can be done in polynomial time, and even in linear time, using the algorithm from [13] and lemma 2.3.

Proposition 7.2 *For each $p \in I$, a full set of characteristics rooted at p contains a polynomial number of characteristics. Each characteristic contains $O(1)$ integer sequences of length at most $2k + 3 = O(\log n)$.*

Proof: See lemmas 4.1. Note that $l = O(1)$, and $k = O(\log n)$. By lemma 3.1, a characteristic contains at most $2\ell + 3 = O(1)$ typical sequences. Each of these contains numbers between 0 and $k + 1$, so by lemma 3.3, they have length $O(k) = O(\log n)$. □

We now show that full sets of characteristics can be computed in polynomial time for a node $p \in I$, when the full sets of characteristics of the children of p are known. In each case, we use the same algorithm as was used in earlier sections for the case that k is a fixed constant. We need to show that the computations can be done in polynomial time.

p is a start node. This case is obvious.

p is a join node. Let q, r be the children of p, and let $FS(p)$, $FS(q)$ be full sets of characteristics at p and r. We use the same notations as in section 4.3.

From proposition 7.2 and lemma 3.15, it follows that we may restrict the lists $[c]$ in theorem 4.3 to lists with each integer sequence in the list with length at most $4k + 5 =$

$O(\log n)$. For each i, $1 \leq i \leq w$, we have to consider the extensions of $\tau(a^{(i)}) - |Z_{t_i}|$ with length at most $4k+5$, and the extensions of $\tau(b^{(i)})$ with length at most $4k+5$. By lemma 3.16, the number of such extensions is polynomially bounded in n. As the $w = O(1)$ (proposition 7.2), it follows that we can compute the full set $FS(p)$ in polynomial time.

p is a forget node. One can directly observe that the computation of $FS(p)$, as done in section 4.4 is linear in the product of the number of elements in a full set and the size of a characteristic (total number of integers and vertices appearing). By proposition 7.2, the former number is polynomial in n, and the latter is $O(\log n)$.

p is an introduce node. Consider the procedure, given in section 4.5. Note that steps 1 and 2 can be done in constant time, as $|X_p| \leq \ell+1$. Step 3 can be done in time, polynomial in the size of $FS(q)$, hence in polynomial time. Step 4 also can be done in polynomial time: for each of the constant bounded many pairs (Z, Z'), and the polynomially bounded many $(Z, \tau[c])$, we must consider $O(\log^2 n)$ cases: the number of different splits of a sequence $\tau(c^{(i)})$ is at most $2 \cdot l(\tau(c^{(i)})) = O(\log n)$. Each of the $O(\log n)$ cases can be done in time, linear in $\sum_{i=1}^{w} l(\tau(c^{(i)})) = O(\log n)$. It follows that the computation of a full set at p takes polynomial time.

So, for each $p \in I$, we can compute a full set of characteristics $FS(p)$ in polynomial time, given full sets for the children of p. So, in polynomial time, we have a full set for the root node of T. As before, the pathwidth of G is at most k, if and only if the full set of the root node is non-empty.

It is not hard to see that one can turn the decision algorithm into a polynomial time algorithm that also constructs the path-decompositions of minimum width, using the approach, described in section 6.

Theorem 7.3 *For every k, there exists a polynomial time algorithm, that, when given a graph $G = (V, E)$ with treewidth at most k, computes the pathwidth of G and a path-decomposition of G of minimum width.*

Corollary 7.4 *The pathwidth, vertex separation number and node search number can be computed in polynomial time for each of the following classes of graphs: cacti, outerplanar graphs, k-outerplanar graphs (k-fixed), Halin graphs, series-parallel graphs, almost graphs with parameter k (k constant).*

Proof: Pathwidth, vertex separation number, and node search number are equivalent notions (see e.g. [34]). Each of the classes mentioned has a constant upper bound on the treewidth of the graphs in the class. □

The pathwidth of trees and forests can be computed in linear time (see [21, 34, 43]).

Note that the running time of our algorithm is quite large: already the bound on the size of the full sets in lemma 3.5 is $\Omega(n^{4\ell+3})$.

8 Final Remarks

8.1 On the constant factors

The algorithms, discussed in section 7 can be considered to be of only theoretical interest. The algorithms discussed in this paper for the case that k is a constant (sections 4, 5 and 6), are more practical. The constant factors of these algorithms are still quite large, although much better than those of previous solutions, and 'only' singly exponential in a the treewidth of the graphs involved. Note that, due to the NP-completeness of the decision problems TREEWIDTH and PATHWIDTH [3], we should not expect to do better than worst case running time exponential in k. We believe that our algorithms are probably 0 for small values of k, e.g., $k = 3$, 4, or 5. For instance, it might well be that the following approach would yield the most practical algorithm for the 'pathwidth ≤ 3' problem:

- Find a tree-decomposition of width ≤ 3 of the input graph with the algorithm from Arnborg and Proskurowski [6], or decide that the treewidth and hence the pathwidth is larger than 3.
- Apply theorem 6.1(ii), with some further optimizations.

Several optimizations to our algorithms seem possible. Most obvious this is for the computations of full sets for **introduce** nodes. Other possible approaches for optimizations are:

- Use modified characterizations and/or modified graph building rules.
- Use the partial ordering $\prec$ on characterizations in full sets, or stronger forms of such partial orderings, and try to remove many characteristics from full sets that are 'dominated' by other 'better' characteristics in that full set.
- Use (Myhill-Nerode) state reduction techniques. See e.g., [7, 1].
- Use 'memoization'; i.e., do not compute full sets at once, but always try to find characteristics that can be included in a (not yet full) set of a node i that is as close to the root as possible. As soon as we find a characteristic in the set of the root of T we are done. This approach may help to significantly decrease the average case time for inputs with a positive answer to the decision problem, but it does not significantly increase the worst case time. However, it also will not improve the time for 'negative' inputs.
- Try to start with inputs with ℓ as small as possible. When combining the result of this paper with results in [30] or [37], or with the approach taken in [13], we still can have $\ell \geq 2k + 1$, and this is probably not practical for most values of k. It might be that the variant of the algorithm mentioned in [13], that uses $O(n \log n)$ time, and only involves tree-decompositions of width at most $k + 1$ has most practical value.
- Combine the results of this paper with graph rewriting techniques, as in [6, 4].

8.2 Related results and open problems

Recently, the algorithms given in this paper have been used in [13] to obtain a linear time algorithm, that given a graph G, decides whether G has treewidth at most k, and if so, find a tree-decomposition of width $\leq k$ (k fixed). Clearly, a similar results holds for pathwidth (cf. theorem 6.1).

Very recent results show that similar techniques can be employed to solve the fixed parameter variant of several other related problems in linear time: minimum cut linear arrangement, modified minimum cut linear arrangement, directed minimum cut linear arrangement, search number, register sufficiency, and others [14]. It seems likely that for some of these (e.g., search number), also polynomial time algorithms for computing the number of graphs with bounded treewidth exist. For others of these problems, this is unlikely: e.g. minimum cut linear arrangement is NP-complete for graphs with treewidth 2 [35].

The results in this paper resolved some interesting open problems concerning the complexity of computing the treewidth or pathwidth for special classes of graphs. We mention some other interesting, remaining open problems in this area: what are the complexities of computing the treewidth of planar graphs, the pathwidth of circular arc graphs, and the treewidth or pathwidth of line graphs. Also, it seems an interesting research subject to try to get more efficient algorithms for computing the pathwidth of special classes of graphs with bounded treewidth, like the outerplanar graphs, Halin graphs, or graphs with treewidth 2 or 3.

References

[1] K. R. Abrahamson and M. R. Fellows. Finite automata, bounded treewidth and well-quasiordering. In *Graph Structure Theory, Contemporary Mathematics vol. 147*, pages 539–564. American Mathematical Society, 1993.

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

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

[4] S. Arnborg, B. Courcelle, A. Proskurowski, and D. Seese. An algebraic theory of graph reduction. In H. Ehrig, H. Kreowski, and G. Rozenberg, editors, *Proceedings of the Fourth Workshop on Graph Grammars and Their Applications to Computer Science*, pages 70–83. Springer Verlag, Lecture Notes in Computer Science, vol. 532, 1991. To appear in J. ACM.

[5] S. Arnborg, J. Lagergren, and D. Seese. Easy problems for tree-decomposable graphs. *J. Algorithms*, 12:308–340, 1991.

[6] S. Arnborg and A. Proskurowski. Characterization and recognition of partial 3-trees. *SIAM J. Alg. Disc. Meth.*, 7:305–314, 1986.

[7] M. W. Bern, E. L. Lawler, and A. L. Wong. Linear time computation of optimal subgraphs of decomposable graphs. *J. Algorithms*, 8:216–235, 1987.

[8] H. L. Bodlaender. Classes of graphs with bounded treewidth. Technical Report RUU-CS-86-22, Dept. of Computer Science, Utrecht University, Utrecht, the Netherlands, 1986.

[9] H. L. Bodlaender. Dynamic programming algorithms on graphs with bounded treewidth. In *Proceedings of the 15th International Colloquium on Automata, Languages and Programming*, pages 105–119. Springer Verlag, Lecture Notes in Computer Science, vol. 317, 1988.

[10] H. L. Bodlaender. Improved self-reduction algorithms for graphs with bounded treewidth. In *Proc. 15th Int. Workshop on Graph-theoretic Concepts in Computer Science WG'89*, pages 232–244. Springer Verlag, Lect. Notes in Computer Science, vol. 411, 1990. To appear in: Annals of Discrete Mathematics.

[11] H. L. Bodlaender. A tourist guide through treewidth. Technical Report RUU-CS-92-12, Department of Computer Science, Utrecht University, Utrecht, 1992. To appear in Acta Cybernetica.

[12] H. L. Bodlaender. Complexity of path-forming games. *Theor. Comp. Sc.*, 110:215–245, 1993.

[13] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the 25th Annual Symposium on Theory of Computing*, pages 226–234. ACM Press, 1993.

[14] H. L. Bodlaender and M. R. Fellows, 1993. Work in progress.

[15] H. L. Bodlaender, J. R. Gilbert, H. Hafsteinsson, and T. Kloks. Approximating treewidth, pathwidth, and minimum elimination tree height. In G. Schmidt and R. Berghammer, editors, *Proceedings 17th International Workshop on Graph-Theoretic Concepts in Computer Science WG'91*, pages 1–12. Springer Verlag, Lecture Notes in Computer Science, vol. 570, 1992.

[16] H. L. Bodlaender and J. Gustedt. A conjecture on the pathwidth of k-trees. Contemp. Math. 147. In section "Open Problems", editor N. Dean, 1993.

[17] H. L. Bodlaender, T. Kloks, and D. Kratsch. Treewidth and pathwidth of permutation graphs. In *Proceedings 20th International Colloquium on Automata, Languages and Programming*, pages 114–125, Berlin, 1993. Springer Verlag, Lecture Notes in Computer Science, vol. 700.

[18] H. L. Bodlaender and R. H. Möhring. The pathwidth and treewidth of cographs. *SIAM J. Disc. Meth.*, 6:181–188, 1993.

[19] 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*, 7:555–582, 1992.

[20] I. S. Duff and J. K. Reid. The multifrontal solution of indefinite sparse symmetric linear equations. *ACM Transactions on Mathematical Software*, 9:302–325, 1983.

[21] J. A. Ellis, I. H. Sudborough, and J. Turner. The vertex separation and search number of a graph. To appear in Information and Computation, 1994.

[22] M. R. Fellows and M. A. Langston. On search, decision and the efficiency of polynomial-time algorithms. In *Proceedings of the 21rd Annual Symposium on Theory of Computing*, pages 501–512, 1989.

[23] A. Habel. *Hyperedge Replacement: Grammars and Languages.* PhD thesis, Univ. Bremen, 1988.

[24] M. Habib and R. H. Möhring. Treewidth of cocomparability graphs and a new order-theoretic parameter. Technical Report 336/1992, Fachbereich Mathematik, Technische Universität Berlin, 1992.

[25] T. Kloks. *Treewidth.* PhD thesis, Utrecht University, Utrecht, the Netherlands, 1993.

[26] T. Kloks. Treewidth of circle graphs. Technical Report RUU-CS-93-12, Department of Computer Science, Utrecht University, Utrecht, 1993.

[27] T. Kloks, H. Bodlaender, H. Müller, and D. Kratsch. Computing treewidth and minimum fill-in: All you need are the minimal separators. To appear in: proceedings 1st European Symposium on Algorithms, ESA'93, 1993.

[28] T. Kloks and D. Kratsch. Finding all minimal separators of a graph. Manuscript, 1993.

[29] T. Kloks and D. Kratsch. Treewidth of chordal bipartite graphs. In P. Enjalbert, A. Finkel, and K. W. Wagner, editors, *Proceedings Symp. Theoretical Aspects of Computer Science, STACS'93*, pages 80–89, Berlin, 1993. Springer Verlag, Lecture Notes in Computer Science, vol. 665.

[30] J. Lagergren. Efficient parallel algorithms for tree-decomposition and related problems. In *Proceedings of the 31rd Annual Symposium on Foundations of Computer Science*, pages 173–182, 1990.

[31] J. Lagergren and S. Arnborg. Finding minimal forbidden minors using a finite congruence. In *Proceedings of the 18th International Colloquium on Automata, Languages and Programming*, pages 533–543. Springer Verlag, Lecture Notes in Computer Science, vol. 510, 1991.

[32] S. J. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *The Journal of the Royal Statistical Society. Series B (Methodological)*, 50:157–224, 1988.

[33] J. Matousěk and R. Thomas. Algorithms finding tree-decompositions of graphs. *J. Algorithms*, 12:1–22, 1991.

[34] R. H. Möhring. Graph problems related to gate matrix layout and PLA folding. In E. Mayr, H. Noltemeier, and M. Sysło, editors, *Computational Graph Theory, Comuting Suppl. 7*, pages 17–51. Springer Verlag, 1990.

[35] B. Monien and I. H. Sudborough. Min cut is NP-complete for edge weighted trees. *Theor. Comp. Sc.*, 58:209–229, 1988.

[36] A. Rajaraman, H. Balakrishnan, and C. Pandu Rangan. Modular decomposition techniques for distance-hereditary graphs. Unpublished Manuscript, 1993.

[37] B. Reed. Finding approximate separators and computing tree-width quickly. In *Proceedings of the 24th Annual Symposium on Theory of Computing*, pages 221–228, 1992.

[38] N. Robertson and P. D. Seymour. Graph minors. I. Excluding a forest. *J. Comb. Theory Series B*, 35:39–61, 1983.

[39] N. Robertson and P. D. Seymour. Graph minors — a survey. In I. Anderson, editor, *Surveys in Combinatorics*, pages 153–171. Cambridge Univ. Press, 1985.

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

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

[42] D. P. Sanders. On linear recognition of tree-width at most four. Manuscript, 1992.

[43] P. Scheffler. A linear algorithm for the pathwidth of trees. In R. Bodendiek and R. Henn, editors, *Topics in combinatorics and graph theory*, pages 613–620, Heidelberg, 1990. Physica-Verlag.

[44] R. Sundaram, K. Sher Singh, and C. Pandu Rangan. Treewidth of circular-arc graphs. Manuscript, to appear in SIAM J. Disc. Math., 1991.

[45] L. C. van der Gaag. *Probability-Based Models for Plausible Reasoning.* PhD thesis, University of Amsterdam, 1990.

[46] T. V. Wimer. *Linear Algorithms on k-Terminal Graphs.* PhD thesis, Dept. of Computer Science, Clemson University, 1987.