Available online at www.sciencedirect.com
SCIENCE DIRECT
Journal of Algorithms 56 (2005) 1–24
Journal of Algorithms
www.elsevier.com/locate/jalgor

Cutwidth I: A linear time fixed parameter algorithm[☆,☆☆]

Dimitrios M. Thilikos[a,*], Maria Serna[a], Hans L. Bodlaender[b]

[a] *Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Campus Nord – Edifici Ω, c/Jordi Girona Salgado 1-3, 08034 Barcelona, Spain*
[b] *Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, The Netherlands*

Received 7 August 2000
Available online 21 February 2005

Abstract

The *cutwidth* of a graph G is the smallest integer k such that the vertices of G can be arranged in a linear layout $[v_1, \dots, v_n]$ in such a way that, for every $i = 1, \dots, n-1$, there are at most k edges with one endpoint in $\{v_1, \dots, v_i\}$ and the other in $\{v_{i+1}, \dots, v_n\}$. In this paper we provide, for any constant k, a linear time algorithm that for any input graph G, answers whether G has cutwidth at most k and, in the case of a positive answer, outputs the corresponding linear layout.
© 2005 Elsevier Inc. All rights reserved.

Keywords: Cutwidth; Treewidth; Pathwidth; Graph layout

DOI of related article: 10.1016/j.jalgor.2004.12.003.
[☆] The work of all the authors was supported by the EU project ALCOM-FT (IST-99-14186). The research of the first author was supported by the Spanish CICYT project TIC2000-1970-CE and the Ministry of Education and Culture of Spain (Resolución 31/7/00 – BOE 16/8/00).
[☆☆] This paper is the full version of part of the paper titled "Constructive linear time algorithms for small cutwidth and carving-width" which appeared in: Proceedings of 11th International Conference ISAAC 2000, Lectures Notes in Comput. Sci., vol. 1969, Springer-Verlag, 2000, pp. 192–203.
[*] Corresponding author.
E-mail addresses: sedthilk@lsi.upc.edu (D.M. Thilikos), mjserna@lsi.upc.edu (M. Serna), hansb@cs.uu.nl (H.L. Bodlaender).

1. Introduction

Linear layouts (or vertex orderings) of graphs provide the framework for the definition of several graph theoretic parameters with a wide range of applications [11]. The cutwidth of a layout is the maximum number of edges connecting vertices on opposite sides of any of the "gaps" between successive vertices in the linear layout. The cutwidth of a graph is the minimum cutwidth over all possible layouts of its vertex set. Deciding whether, for a given G and an integer k, $\text{cutwidth}(G) \leqslant k$, is an NP-complete problem known in the literature as the MINIMUM CUT LINEAR ARRANGEMENT problem [14]. Cutwidth has been extensively examined [8,10,12,13,15,17,22] and it is closely related with other graph theoretic parameters like pathwidth, bandwidth, or modified bandwidth [4,8–10,15,16].

The results of this paper concern the fixed parameter tractability of cutwidth which is closely related to immersion properties. Recall that a graph H is said to be immersed to G if a graph isomorphic to H can be obtained from a subgraph of G by a series of *lift* operations. A lift operation replaces two adjacent edges $\{a, b\}$, $\{b, c\}$ by the edge $\{a, c\}$. The main motivation of our research were the results of Robertson and Seymour in their *Graph minors* series where, among others, they prove that any set of graphs contains a finite number of immersion minimal elements (see [18]). As a consequence, we have that for any class $\mathcal{C}$ of graphs the set of graphs *not* in $\mathcal{C}$ contains a *finite* set (we call it immersion obstruction set of $\mathcal{C}$) of immersion minimal elements. Therefore, we have the following finite characterization for $\mathcal{C}$: a graph G is in $\mathcal{C}$ iff none of the graphs in the immersion obstruction set of $\mathcal{C}$ is immersed to G. Combining this observation with the fact that, for any fixed H, there exists a polynomial time algorithm deciding, given G as input, whether a H is immersed to G (see [12,19]), we imply the existence of a polynomial time recognition algorithm for any immersion-closed graph class.

Unfortunately, the result of Robertson and Seymour is *non-constructive* in the sense that it does not provide any method of constructing the corresponding obstruction set. Therefore, it only guarantees the *existence* of a polynomial time algorithm and does not provide one. However, it gives a strong motivation towards identifying the corresponding algorithms for a wide range of graph classes and parameters. So far, it appears that the most popular class (see [12,13]) that is immersion-closed, is the class of graphs with cutwidth bounded by a fixed constant. A direct consequence is that, for any fixed k, there exists a polynomial time algorithm checking whether a graph has cutwidth at most k.

The first algorithm checking whether cutwidth is at most k was given by Makedon and Sudborough in [10] where a $O(n^{k-1})$ dynamic programming algorithm is described. This time complexity has been considerably improved by Fellows and Langston in [13] where they provide, for any fixed k, an algorithm that checks whether a graph has cutwidth at most k in $O(n^3)$. Furthermore, using a technique introduced in [12] (see also [2]) the bound can be further reduced to $O(n^2)$, while in [1] a general method is given to construct a linear time algorithm that decides whether a given graph has cutwidth at most k, for k constant. However the methodology in [1] gives only a decision algorithm: it does not give any method to construct the corresponding layout. In this paper, we give an explicit description, for any $k \geqslant 1$, of a linear time algorithm that checks whether an input graph G has cutwidth at most k and, if this is the case, it further *outputs* a linear layout of G of minimum cutwidth.

At this point, we want to give an informal description of our algorithm. It starts computing a "nice" path decomposition of bounded width of the input graph G (for a formal definition see Section 2.1). The path decomposition allows the definition of an appropriate sequence of subgraphs and the algorithm proceeds from left to right of the path decomposition. First, consider the following exponential algorithm, that builds the set of all layouts of cutwidth at most k. The algorithm goes through the path decomposition from left to right, at each point having in a data structure the set of all layouts of cutwidth at most k of the subgraph induced by the vertices encountered so far. When a new vertex is encountered, we try to insert it at all different spots in all the layouts in the data structure; from the layouts we thus obtain, we keep those whose cutwidth is at most k. Clearly, this algorithm can use exponential time, but solves the problem: G has cutwidth at most k, if and only if the data structure contains at least one layout when the whole path decomposition is scanned (for a more detailed analysis of this procedure, see [4]).

In order to decrease the running time of the algorithm, we modify it as follows. Instead of storing entire layouts, we only store that information of the layout that is necessary to determine whether inserting a vertex at some spot increases the cutwidth to more than k, and to compute the same type of information for the new sequence again. Layouts which have the same such structural information can be deemed to be equivalent, and need only one object to have stored in the data structure. The key notion for this structural information is the *characteristic*, used here in a similar fashion as in linear time algorithms for related parameters, like pathwidth and treewidth in [5], linear-width in [7], and branchwidth in [6]. In a few words, a characteristic serves to filter the main data structure of a parameter to its essential part, a part that can be constructed from node to node of a path decomposition. Characteristics consist of the sequence in which the vertices in the current node of the path decomposition appear in the layout, plus sequences of integers representing numbers of edges crossing certain gaps in the layout. As we will see, the size of the information encoded by a characteristic depends on the width of the path decomposition and, therefore, it is constant for graphs with bounded pathwidth.

A consequence of our result is an algorithm that, for any k, is able to determine the immersion obstruction set for the class of the graphs with cutwidth at most k. We mention that optimal constructive results exist so far only for minor closed parameters such as treewidth and pathwidth [3,5], agile search parameters [7], linear-width [7], and branch-width [6]. Besides the fact that our techniques are motivated by those used in the aforementioned minor-closed parameters, in our knowledge, our results are the first concerning immersion-closed parameters and we believe that our approach is applicable to other parameters as well (e.g., MODIFIED CUTWIDTH, 2-D GRID LOAD FACTOR, or BINARY GRID LOAD FACTOR—see [12]).

The paper is organized as follows. Section 2 contains the definitions of cutwidth and pathwidth and several definitions and results on sequences of sequences, finishing with the definition of characteristics. That way, we provide the theoretical framework for the analysis of the algorithm. Section 3 describes the algorithm for deciding whether the cutwidth of a graph is at most k and Section 4 presents the modification needed to compute also a layout with small cutwidth when there is one. We conclude with some final remarks on Section 5.

2. Definitions and preliminary results

We proceed with a number of definitions and notations, dealing with finite sequences (i.e., ordered sets) of a given finite set $\mathcal{O}$. For our purposes, $\mathcal{O}$ can be a set of numbers, sequences of numbers, vertices, or vertex sets. Let ω be a sequence of elements from $\mathcal{O}$ with length r. We use the notation $[\omega_1, \ldots, \omega_r]$ to represent ω and we define $\omega[i, j]$ as the subsequence $[\omega_i, \ldots, \omega_j]$ of ω (in case $j < i$, the result is the empty subsequence $[\,]$). We also denote as $\omega(i)$ the element of ω indexed by i and by $\omega[i]$ the sequence $[\omega(i)]$.

Given a set S containing elements of $\mathcal{O}$, we denote as $\omega[S]$ the subsequence of ω that contains only the elements of ω that are in S, respecting their relative positions in ω. Given two sequences ω^1, ω^2, defined on $\mathcal{O}$, where $\omega^i = [\omega^i_1, \ldots, \omega^i_{r_i}]$, $i = 1, 2$, we define the *concatenation* of ω^1 and ω^2 as

$$\omega^1 \cdot \omega^2 = \left[\omega^1_1, \ldots, \omega^1_{r_1}, \omega^2_1, \ldots, \omega^2_{r_2}\right].$$

Unless mentioned otherwise, we consider that the first element of a sequence ω is indexed by 1, i.e., $\omega = \omega[1, |\omega|]$.

All the graphs of this paper are finite, undirected, and without loops or multiple edges (our results can be straightforwardly generalized to the case where the last restriction is altered). We denote the vertex (edge) set of a graph G by $V(G)$ $(E(G))$ and set $n = |V(G)|$. As our graphs will not have multiple edges, we represent an edge $e \in E(G)$ by a two vertex subset.

Let G be a graph and $S \subseteq V(G)$. We call the graph $(S, E(G) \cap \{\{x, y\} \mid x, y \in S\})$ the *subgraph of G induced by S* and we denote it by $G[S]$. For any $e \in E(G)$, we set $G - e = (V(G), E(G) - \{e\})$ and for any $N \subseteq V(G)$ and $u \notin V(G)$, the graph $G +^u N$ is obtained by adding to G the new vertex u and the edges $\{\{u, v\} \mid v \in N\}$. We denote by $E_G(S)$ the set of edges of G that have an endpoint in S; we also set $E_G(v) = E_G(\{v\})$ for any vertex v. If $E \subseteq E(G)$ then we denote as $V(E)$ the set of all the endpoints of the edges in E, i.e., we set $V(E) = \bigcup_{e \in E} e$. The neighborhood of a vertex v in graph G is the set of vertices in G that are adjacent to v in G and we denote it as $N_G(v)$, i.e., $N_G(v) = V(E_G(v)) - \{v\}$. If l is a sequence of vertices, we denote the set of its vertices as $V(l)$. If $x \in V(l)$ then we set $l - x = l[V(l) - \{x\}]$. If l is a sequence of all the vertices of G without repetitions, then we will call it *vertex ordering* or *layout* of G. If l is a vertex ordering of G, the *rank* of a vertex $u \in V(l)$ is its position in the ordering, and we denote it by $\mathrm{rank}_l(u)$.

2.1. Pathwidth

A *path decomposition* of a graph G is defined as a sequence $X = [X_1, \ldots, X_r]$ of subsets of $V(G)$ satisfying the following properties.

(1) $\bigcup_{i, 1 \leqslant i \leqslant r} X_i \subseteq V(G)$.
(2) $\forall_{\{v,u\} \in E(G)}\ \exists_{i, 1 \leqslant i \leqslant r}\ \{v, u\} \subseteq X_i$.
(3) $\forall_{v \in V(G)}\ \exists_{i, j, 1 \leqslant i \leqslant j \leqslant r}\ \forall_{h, 1 \leqslant h \leqslant r}\ v \in X_h \Leftrightarrow i \leqslant h \leqslant j$.

We call the sets $X_1, \ldots, X_r$, the *nodes* of the path decomposition X. The *width* of X is equal to $\max_{1 \leqslant i \leqslant r}\{|X_i| - 1\}$ and the *pathwidth* of a graph G is the minimum width over all path decompositions of G. We say that a path decomposition $X = [X_1, \ldots, X_r]$ is *nice* if $|X_1| = 1$ and $\forall_{i, 2 \leqslant i \leqslant |X|}\ |(X_i - X_{i-1}) \cup (X_{i-1} - X_i)| = 1$. The following lemma follows directly from the definitions.

Lemma 1. *For some constant k, given a path decomposition of a graph G that has width at most k and $O(|V(G)|)$ nodes, one can find a nice path decomposition of G that has width at most k and has at most $2|V(G)|$ nodes in $O(|V(G)|)$ time.*

We distinguish the three types of nodes in a nice path decomposition $X = \{X_1, \ldots, X_r\}$. We say that X_i is an *introduce* node if $|X_i - X_{i-1}| = 1$, while X_i is a *forget* node if $|X_{i-1} - X_i| = 1$. It is easy to observe that any node X_i, $i \geqslant 2$, of a nice path decomposition is either an *introduce* or a *forget* node. We call the first node X_1 of X, *start* node (recall that $|X_1| = 1$). Notice that if the last node X_r is a forget node, the node can be removed and we still have a path decomposition. Hence we may assume that X_r is an introduce node.

Finally, for each node in the path decomposition we associate a graph G_i, $1 \leqslant i \leqslant r$, as follows. We define $V_i = \bigcup_{j, 1 \leqslant j \leqslant i} X_j$ and $G_i = G[V_i]$. Notice that if X_i, $i > 1$, is an introduce node then $V(G_{i-1}) \neq V(G_i)$, and we will call the unique vertex in $X_i - X_{i-1}$ the *introduced* vertex of G_i. Notice that if X_i, $i > 1$, is a forget node then $V(G_{i-1}) = V(G_i)$, and we will call the unique vertex in $X_{i-1} - X_i$ the *forgotten* vertex of G_i.

2.2. Cutwidth

The cutwidth of a graph G with n vertices is defined as follows. Let $l = [v_1, \ldots, v_n]$ be a layout of $V(G)$. For $i = 1, \ldots, n-1$, we define the *cut at position i*, denoted by $\theta_{l,G}(i)$, as the set of crossover edges of G that have one endpoint in $l[1, i]$ and one in $l[i+1, n]$, i.e., $\theta_{l,G}(i) = E_G(l[1, i]) \cap E_G(l[i+1, n])$. The *cutwidth of a layout l* of $V(G)$ is $\max_{i, 1 \leqslant i \leqslant n-1}\{|\theta_{l,G}(i)|\}$. The *cutwidth* of a graph is the minimum cutwidth over all the vertex orderings of $V(G)$. It is easy to see the following (see also [10]).

Lemma 2. *For any graph G,* $\mathrm{cutwidth}(G) \geqslant \mathrm{pathwidth}(G)$.

If $l = [v_1, \ldots, v_n]$ is a vertex ordering of a graph G, we set

$$\mathbf{Q}_{G,l} = \big[[0], \big[|\theta_{l,G}(1)|\big], \ldots, \big[|\theta_{l,G}(n-1)|\big], [0]\big].$$

The above sequence keeps information on the sequence of cuts at different positions of a layout. We also assume that the indices of the elements of $\mathbf{Q}_{G,l}$ start from 0 and finish on n, i.e., $\mathbf{Q}_{G,l} = \mathbf{Q}_{G,l}[0, n]$. Moreover, in what follows, we will assume that the indices of any sequence of sequences will start from 0. Clearly, $\mathbf{Q}_{G,l}$ is a sequence of sequences of numbers each containing only one element. We insist on the, somewhat overloaded, definition of $\mathbf{Q}_{G,l}$ in order to make its notation compatible with the definition of characteristics in Section 2.4 (for an example of $\mathbf{Q}_{G,l}$, see Fig. 1).

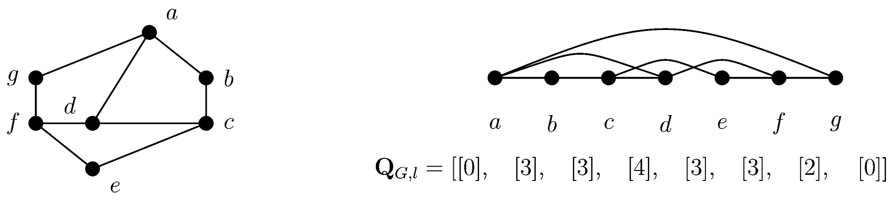
a
b
c
d
e
f
g
a b c d e f g
$\mathbf{Q}_{G,l} = [[0], [3], [3], [4], [3], [3], [2], [0]]$

Fig. 1. An examples of a graph G, a layout and the sequence $\mathbf{Q}_{G,l}$.

2.3. Sequences of integers

We denote as $\mathcal{S}$ the set of all finite sequences of non-negative integers. For any sequence $A = [a_1, \ldots, a_{|A|}] \in \mathcal{S}$ and any integer $t \geqslant 0$ we set $A + t = [a_1 + t, \ldots, a_{|A|} + t]$, and $\max A = \max_{1 \leqslant i \leqslant |A|} a_i$. Also, if $A = [a_1, \ldots, a_r]$ and $B = [b_1, \ldots, b_r]$ are two equal-size sequences of non-negative integers, we define $A + B = [a_1 + b_1, \ldots, a_r + b_r]$. If $A, B \in \mathcal{S}$ and $A = [a_1, \ldots, a_{|A|}]$, we say that $B \sqsubseteq A$ if B is a subsequence of A obtained after applying a number of times (possibly none) the following operations.

(i) If for some i, $1 \leqslant i \leqslant |A| - 1$, $a_i = a_{i+1}$, then set $A \leftarrow A[1, i] \cdot A[i + 2, |A|]$.
(ii) If the sequence contains two elements a_i and a_j such that $j - i \geqslant 2$ and $\forall_{k, i<k<j}\ a_i \leqslant a_k \leqslant a_j$ or $\forall_{k, i<k<j}\ a_i \geqslant a_k \geqslant a_j$, then set $A \leftarrow A[1, i] \cdot A[j, |A|]$.

We define the *compression* $\tau(A)$ of a sequence $A \in \mathcal{S}$, as the unique minimum length element of $\{B \mid B \sqsubseteq A\}$. For example,

$$\tau\big([\mathbf{5}, 5, 6, 7, 7, 7, \mathbf{7}, 4, 4, \mathbf{3}, 5, 4, 6, \mathbf{8}, \mathbf{2}, \mathbf{9}, 3, 4, 6, 7, \mathbf{2}, \mathbf{7}, 5, 4, 4, 6, \mathbf{4}]\big)$$
$$= [5, 7, 3, 8, 2, 9, 2, 7, 4].$$

We call a sequence A *typical* if $A \in \mathcal{S}$ and $\tau(A) = A$.

The following lemma is a direct consequences of the definitions.

Lemma 3. *For $i = 1, 2$, let A_i, B_i such that $A_i \sqsubseteq B_i$. Then,*

(1) *$A_1 \cdot A_2 \sqsubseteq B_1 \cdot B_2$, and*
(2) *if $|A_1| = |A_2|$ and $|B_1| = |B_2|$ then $\tau(A_1 + A_2) = \tau(B_1 + B_2)$.*

The following results have been proved in [5] (Lemmata 3.3 and 3.5 respectively).

Lemma 4. *If $A \in \mathcal{S}$ and $\max A \leqslant k$, then $\tau(A)$ contains at most $2k - 1$ elements.*

Lemma 5. *The number of different typical sequences consisting of integers in $\{0, 1, \ldots, n\}$ is at most $\frac{8}{3}2^{2n}$.*

Notice that $B = \tau(A)$ is a subsequence $[a_{i_1}, \ldots, a_{i_{|B|}}]$ of $A = [a_1, \ldots, a_{|A|}]$ such that for any j, $1 \leqslant j \leqslant |B| - 1$ either $a_{i_j} \leqslant \min\{a_{i_j+1}, \ldots, a_{i_{j+1}-1}\}$ and $\max\{a_{i_j+1}, \ldots, a_{i_{j+1}-1}\} \leqslant a_{i_{j+1}}$ or $a_{i_j} \geqslant \max\{a_{i_j+1}, \ldots, a_{i_{j+1}-1}\}$ and $\min\{a_{i_j+1}, \ldots, a_{i_{j+1}-1}\} \geqslant a_{i_{j+1}}$. In general we may have more than one such subsequence. We fix one of them and define the function $\beta_A : \{1, \ldots, |\tau(A)|\} \to \{1, \ldots, |A|\}$ where $\beta_A(j) = i_j$ is one of the possible original positions in A of the jth element in $\tau(A)$. Consider the sequence of the previous example

$$A = [\mathbf{5}, 5, 6, \underline{7, 7, \mathbf{7}, 7}, 4, 4, \mathbf{3}, 5, 4, 6, \mathbf{8}, \mathbf{2}, \mathbf{9}, 3, 4, 6, 7, \mathbf{2}, \mathbf{7}, 5, 4, 4, 6, \mathbf{4}],$$

then we have

$$\begin{array}{lll} \beta_A(1) = 1, & \beta_A(2) = 6 \text{ (or 4 or 5 or 7)}, & \beta_A(3) = 10, \\ \beta_A(4) = 14, & \beta_A(5) = 15, & \beta_A(6) = 16, \\ \beta_A(7) = 21, & \beta_A(8) = 22, & \beta_A(9) = 27. \end{array}$$

The following lemma is a direct consequence of the definition of β.

Lemma 6. *Let A be any sequence in $\mathcal{S}$. Then, for any i, $1 \leqslant i < |\tau(A)|$, we have $\tau(A[\beta_A(i), \beta_A(i+1)]) = [A(\beta_A(i)), A(\beta_A(i+1))]$.*

Analogously, we define the function $\beta_A^{-1} : \{1, \ldots, |A|\} \to \{1, \ldots, \tau(A)\}$ such that $\beta_A^{-1}(j)$ is the unique i such that $\beta_A(i) = j$.

For any $A \in \mathcal{S}$, we define $\alpha(A)$ in the same way as $\tau(A)$ with the difference that only operation (i) is considered, i.e., we remove repetitions of a number on successive positions in the sequence. If now A is a typical sequence, we define the *set of extensions* of A as

$$\mathcal{E}(A) = \{\tilde{A} \in \mathcal{S} \mid \alpha(\tilde{A}) = A\}.$$

We call a sequence A *dense* if $A \in \mathcal{E}(\tau(A))$. If A is dense then all the sequences in $\mathcal{E}(A)$ are dense. Finally, notice also that if A is dense and $B \sqsubseteq A$ then $B \in \mathcal{E}(\tau(A))$. For example, the sequences $[5, 7, 7, 7, 4, 8, 8, 8, 8]$ and $[1, 7, 2, 6, 4, 4, 4, 4, 4]$ are dense.

Notice that for any typical sequence A, if $B \in \mathcal{E}(A)$, $B(i) \neq B(i+1)$, and $\beta_B^{-1}(i) = j$, then the subsequence $B[i, i+1]$ represents the jth number change in B.

The results in the following two lemmata are direct consequences of the definitions of β and β^{-1}.

Lemma 7. *Let A be a sequence in S, and $B \in \mathcal{E}(\tau(A))$ then, for any i, $1 \leqslant i \leqslant |B|$,*

(1) $A(\beta_A(\beta_B^{-1}(j))) = B(j)$,
(2) $B(j) \neq B(j+1) \Rightarrow \beta_B^{-1}(j+1) = \beta_B^{-1}(j) + 1$, *and*
(3) $B(j) = B(j+1) \Rightarrow \beta_B^{-1}(j+1) = \beta_B^{-1}(j)$.

Let $A = [a_1, \ldots, a_{r_1}]$ and $B = [b_1, \ldots, b_{r_2}]$ be two sequences in $\mathcal{S}$. We say that $A \leqslant B$ if $r_1 = r_2$ and $\forall_{1 \leqslant i \leqslant r_1}\ a_i \leqslant b_i$. In general, we say that $A \prec B$ if there exist extensions $\tilde{A} \in \mathcal{E}(A)$, and $\widetilde{B} \in \mathcal{E}(B)$ such that $\tilde{A} \leqslant \widetilde{B}$. For example if $A = [1, 7, 2, 6, 4]$ and $B = [5, 7, 4, 8]$ then $A \prec B$ because $\widetilde{B} = [5, 7, 7, 7, 4, 8, 8, 8, 8]$ is an extension of B, $\tilde{A} = [1, 7, 2, 6, 4, 4, 4, 4, 4]$ is an extension of A, and $\tilde{A} \leqslant \widetilde{B}$.

The following lemma corresponds to Corollary 3.11 of [5].

Lemma 8. *If A and B are two sequences then $A \prec B$ if and only if $\tau(A) \prec \tau(B)$.*

The following lemma follows directly from Lemmas 8 and 3.13 of [5].

Lemma 9. *Let A and B be two integer sequences where $|A| = |B|$ and let $Y = A + B$. Let $A_0 \prec A$ and $B_0 \prec B$. Then there exists two sequences $A_0^* \in \mathcal{E}(A_0)$ and $B_0^* \in \mathcal{E}(B_0)$ where $\tau(A_0^* + B_0^*) \prec \tau(Y)$.*

The following three lemmata establish basic properties of typical sequences. They follow directly from the definitions.

Lemma 10. *Given $R \in \mathcal{S}$, if we set $A = \tau(R)$ then, for any m, $1 \leqslant m \leqslant |A|$, there exists an i, $1 \leqslant i \leqslant |R|$ such that $A[1, m] = \tau(A[1, i])$ and $A[m, |A|] = \tau(R[i, |R|])$.*

Lemma 11. *Let A_1, A_2 be two typical sequences where $A_2 \prec A_1$. Then, for any m_1, $1 \leqslant m_1 \leqslant |A_1|$, there exists an m_2, $1 \leqslant m_2 \leqslant |A_2|$, such that $A_2[1, m_2] \prec A_1[1, m_1]$ and $A_2[m_2, |A_2|] \prec A_1[m_1, |A_1|]$.*

Lemma 12. *Let A_i, B_i, $i = 1, 2$, be four typical sequences where $A_i \prec B_i$, $i = 1, 2$. Then $\tau(A_1 \cdot A_2) \prec \tau(B_1 \cdot B_2)$.*

The next result reveals a property of typical sequences that is crucial for the correctness of our algorithm.

Lemma 13. *Given $R \in \mathcal{S}$, if we set $A = \tau(R)$ then, for any r, $1 \leqslant r \leqslant |R|$, there exists an integer i, $1 \leqslant i \leqslant |A|$, such that $A[1, i] \prec \tau(R[1, r])$ and $A[i, |A|] \prec \tau(R[r, |R|])$.*

Proof. In the case where, for any r, there exists an integer i, $1 \leqslant i \leqslant |A|$, such that $A[1, i] = \tau(R[1, r])$ and $A[i, |A|] = \tau(R[r, |R|])$ we have a stronger version of the required and we are done.

Otherwise, for any r, $1 \leqslant r \leqslant |R|$, let k and l, $1 \leqslant k < r < l \leqslant |R|$, be the indices of the two local extremes closest to r (i.e., $R[k, l]$ is a maximal monotone subsequence). Let $j = \beta_A^{-1}(k)$, $1 \leqslant j < |A|$, be the local maximum of A corresponding to $R(k)$. Observe that $A[1, j] = \tau(R[1, k])$ and $A[j + 1, |A|] = \tau(R[l, |R|])$. Furthermore, as A is a typical sequence, we have that $A(j) \neq A(j + 1)$ and therefore, $R(k) \neq R(l)$.

Let us show that, in case $R(k) > R(l)$, the lemma holds taking $i = j + 1$. When $R(k) > R(l)$ we have that $A(j) = R(k)$ and that $A[j + 1] = R[l] \prec R[k + 1, r]$. Therefore, $A[j, j + 1] \prec \tau(R[k, r])$ and $A[j + 1] \prec \tau(R[r, l])$. Using the fact that $A[1, j] = \tau(R[1, k])$ and Lemma 12, we get

$$\begin{aligned} A[1, j + 1] &= \tau\bigl(A[1, j] \cdot A[j + 1]\bigr) \prec \tau\bigl(\tau\bigl(R[1, k]\bigr) \cdot \tau\bigl(R[k + 1, r]\bigr)\bigr) \\ &= \tau\bigl(R[1, k] \cdot R[k + 1, r]\bigr) = \tau\bigl(R[1, r]\bigr). \end{aligned}$$

Now using the fact that $A[j + 1, |A|] = \tau(R[l, |R|])$ and Lemma 12, we get

$$\begin{aligned} A\bigl[j + 1, |A|\bigr] &= \tau\bigl(A[j + 1] \cdot A\bigl[j + 1, |A|\bigr]\bigr) \prec \tau\bigl(\tau\bigl(R[r, l]\bigr) \cdot \tau\bigl(R\bigl[l, |R|\bigr]\bigr)\bigr) \\ &= \tau\bigl(R[r, l] \cdot R\bigl[l, |R|\bigr]\bigr) = \tau\bigl(R\bigl[r, |R|\bigr]\bigr). \end{aligned}$$

In the case where $R(k) < R(l)$ a symmetric argument shows that $A[1, j] \prec \tau(R[1, |R|]$ and $A[j, |A|] \prec \tau(R[r, |R|])$, so the lemma holds taking $i = j$. □

As an example illustrating Lemma 13, we consider the sequences

$$R = [\mathbf{2}, 6, 7, \mathbf{8}, 5, 4, 3, 5, \mathbf{2}, 4, \mathbf{6}, 4, \mathbf{4}] \quad \text{and} \quad A = \tau(R) = [2, 8, 2, 6, 4].$$

If we choose $r = 7$, we have that $j = 2$, $k = 4$, and $l = 9$. Notice that,

$$\begin{aligned}
[2, 8] &= \tau\big([2, 6, 7, 8]\big),\\
[8, 2] &\prec \tau\big([8, 5, 4, 3]\big),\\
[2] &\prec \tau\big([3, 5, 2]\big),\\
[2, 6, 4] &= \tau[2, 4, 6, 4, 4],\\
[2, 8, 2] &\prec \tau\big([2, 6, 7, 8, 5, 4, 3]\big),\\
[2, 6, 4] &\prec \tau\big([3, 5, 2, 4, 6, 4, 4]\big).
\end{aligned}$$

Now we extend the definitions to sequences of typical sequences. Suppose now that $\mathbf{A} = [A_0, \ldots, A_r]$ and $\mathbf{B} = [B_0, \ldots, B_r]$ are two sequences of typical sequences (recall that such sequences are indexed starting by 0). We say that $\mathbf{A} \prec \mathbf{B}$ if $\forall_{i, 0 \leqslant i \leqslant r}\ A_i \prec B_i$. For any integer t we set $\mathbf{A} + t = [A_0 + t, \ldots, A_{|\mathbf{A}|} + t]$ and $\max(\mathbf{A}) = \max_{i, 0 \leqslant i \leqslant |\mathbf{A}|}\{\max A_i\}$. Finally, for any sequence of typical sequences $\mathbf{A}$ we set $\tau(\mathbf{A}) = \tau(\mathbf{A}(0) \cdot \cdots \cdot \mathbf{A}(|\mathbf{A}|))$. As an example,

$$\begin{aligned}
\tau\big(\big[[5, 2, 8, 1], [4, 9, 3], [3], [3, 9, 2, 5, 3]\big]\big) &= \tau\big([\mathbf{5}, \mathbf{2}, \mathbf{8}, \mathbf{1}, 4, 9, 3, 3, 3, \mathbf{9}, \mathbf{2}, \mathbf{5}, \mathbf{3}]\big)\\
&= [5, 2, 8, 1, 9, 2, 5, 3].
\end{aligned}$$

Given two typical sequences A, B and an integer j, $1 \leqslant j \leqslant |\tau(A \cdot B)|$, we define

$$\delta(A, B, j) = \begin{cases} (0, \beta_{A \cdot B}(j)) & \text{if } \beta_{A \cdot B}(j) \leqslant |A|,\\ (1, \beta_{A \cdot B}(j) - |A|) & \text{otherwise.} \end{cases}$$

As an example we have that if $A = [1, 3, 2]$ and $B = [8, 5, 9]$, we have that $\tau(A \cdot B) = [1, 9]$, $\delta(A, B, 1) = (0, 1)$, and $\delta(A, B, 2) = (1, 3)$. Function δ will be used in Section 4.

2.4. Characteristics

We now define the notion of *characteristic of a vertex ordering* of a graph with respect to a small subset of its vertices. Characteristics will serve as key-tools for our algorithm. Several other versions of characteristics have been used for the computation of other parameters like pathwidth and treewidth [5], linear-width [7], and branchwidth [20]. As mentioned before, a characteristic serves to filter the main data structure of a parameter to its essential part, a part that is able to be constructed from node to node of the path decomposition. Moreover, the information encoded by a characteristic certifies the existence of small cutwidth layouts, furthermore it depends only on the width of the path decomposition and, therefore, its size is bounded by a fixed constant w.

```
Procedure Com(l, R, S).

Input: A characteristic $(l, \mathbf{R})$ and a set $S$ of elements in $l = [v_1, \dots, v_{|l|}]$.
Output: A characteristic $(\lambda, \mathbf{A})$.

1: $\lambda \leftarrow l[S]$ and assume the notation $\lambda = [v_{i_1}, v_{i_2}, \dots, v_{i_\rho}]$.
2: $\mathbf{A} \leftarrow [\tau(\mathbf{R}[0, i_1 - 1]), \tau(\mathbf{R}[i_1, i_2 - 1]), \dots, \tau(\mathbf{R}[i_{\rho-1}, i_\rho - 1]), \tau(\mathbf{R}[i_\rho, |l|])]$.
3: Output $(\lambda, \mathbf{A})$.
4: End.
```

Fig. 2. Procedure Com.

We call a *characteristic* any pair $(\lambda, \mathbf{A})$ where λ is a layout of a graph and $\mathbf{A}$ is a sequence of typical sequences in $\mathcal{S}$ such that $|\mathbf{A}| = |\lambda| + 1$. Notice that for any graph G and any vertex ordering l of $V(G)$ the pair $(l, \mathbf{Q}_{G,l})$ is a characteristic.

To give an example of a characteristic, consider the pair

$$\big([a, b, c, d, e], \big[[0, 3], [4], [3, 7, 2], [3], [8, 1, 3], [3, 8, 4, 6]\big]\big).$$

We also use a different notation for characteristics, e.g.:

$$[0, 3]\, \boldsymbol{a}\, [4]\, \boldsymbol{b}\, [3, 7, 2]\, \boldsymbol{c}\, [3]\, \boldsymbol{d}\, [8, 1, 3]\, \boldsymbol{e}\, [3, 8, 4, 6].$$

The procedure Com, described in Fig. 2, defines the *compression* of a characteristic relative to a subset of vertices. Intuitively, it compresses the concatenations of the sequences of $\mathbf{R}$ delimited by the positions of the elements of S (recall that the indices of the sequences of $\mathbf{R}$ start from 0).

The compression of the previous example of characteristic to the set $S = \{a, c\}$ is the characteristic

$$[0, 3]\, \boldsymbol{a}\, [4, 3, 7, 2]\, \boldsymbol{c}\, [3, 8, 1, 8, 4, 6].$$

Let us start by defining a *characteristics* associated to a vertex ordering of a graph. Given a graph G with n vertices, a vertex ordering l of G and $S \subseteq V(G)$, the S-*characteristic* of l is $C_S(G, l) = \mathsf{Com}(l, \mathbf{Q}_{G,l}, S)$. Notice that, from the definition of the S-characteristic of a vertex ordering l of a graph G we have $C_{V(G)}(G, l) = (l, \mathbf{Q}_{G,l})$, taking into account that $\mathsf{Com}(l, \mathbf{Q}_{G,l}, V(G)) = (l, \mathbf{Q}_{G,l})$.

As an example we mention that, for the graph and ordering l given in Fig. 5, taking $N = \{b, e, g\}$, the characteristics $C_N(G, l)$ and $C_{V(G)-N}(G, l)$ are:

$$[0, 3]\, \boldsymbol{b}\, [3, 4, 3]\, \boldsymbol{e}\, [3, 2]\, \boldsymbol{g}\, [0], \qquad [0]\, \boldsymbol{a}\, [3]\, \boldsymbol{c}\, [4]\, \boldsymbol{d}\, [3]\, \boldsymbol{f}\, [2, 0].$$

Given the S-characteristics $(\lambda^i, \mathbf{A}^i)$, $i = 1, 2$, of two different vertex orderings l_1 and l_2 of G we say that $(\lambda^1, \mathbf{A}^1) \prec (\lambda^2, \mathbf{A}^2)$ when $\lambda^1 = \lambda^2$ and $\mathbf{A}^1 \prec \mathbf{A}^2$.

The following result is an easy consequence of the definition of compression and the obvious fact that for a sequence of typical sequences $\mathbf{A}$, if $\mathbf{A} = \tau(\mathbf{R})$, then $\mathbf{A} + 1 = \tau(\mathbf{R} + 1)$.

Lemma 14. *Given a graph G, a vertex ordering l of G, and a vertex subset S. Assume that $C_S(G, l) = (\lambda, \mathbf{A})$. For any two vertices a, b of S, let $i = \mathrm{rank}_l(a)$, $i' = \mathrm{rank}_\lambda(a)$, $j = \mathrm{rank}_l(b)$, and $j' = \mathrm{rank}_\lambda(b)$, then we have*

$$\begin{aligned}&\big(\lambda, \mathbf{A}\big[1, i'-1\big]\cdot\big(\mathbf{A}\big[i', j'-1\big]+1\big)\cdot\mathbf{A}\big[j', |\mathbf{A}|\big]\big)\\&\quad=\mathsf{Com}\big(l, \mathbf{Q}_{G,l}[1, i-1]\cdot\big(\mathbf{Q}_{G,l}[i, j-1]+1\big)\cdot\mathbf{Q}_{G,l}\big[j, |\mathbf{Q}_{G,l}|\big], S\big).\end{aligned}$$

Given a graph G and a vertex subset S, we say that a characteristic $(\lambda, \mathbf{A})$ is a *S-characteristic* when $(\lambda, \mathbf{A}) = C_S(l, G)$ for some ordering l of the vertices of G.

3. A decision algorithm for cutwidth

In this section, we give for any pair of integer constants k, w, an algorithm that, given a graph G and a nice path decomposition $X = [X_1, \dots, X_r]$ of width at most w, decides whether G has cutwidth at most k.

Using now Lemma 5 and working in a similar way as in the proof of Lemma 3.1 in [5] we can prove that for any i, the number of X_i-characteristics of G_i depends only on k and w.

Lemma 15. *Let G be a graph and let $X = [X_1, \dots, X_r]$ be a nice path decomposition of G with width at most w. Let X_i, $1 \leqslant i \leqslant r$, be some node in X. The number of different X_i-characteristics of all possible vertex orderings of $G_i = G[\bigcup_{j=1}^{i} X_j]$ with cutwidth at most k, is bounded by $(w+1)!(\frac{8}{3}2^{2k})^{w+1}$.*

Proof. Let $(\lambda, \mathbf{A})$ be a X_i-characteristic of some vertex ordering of G_i. Clearly, $V(\lambda) = X_i$ and, as $|X_i| \leqslant w+1$, there are at most $(w+1)!$ ways to choose λ. For each one of them, there are $\leqslant w+1$ typical sequences in $\mathbf{A}$ to be chosen. From Lemma 5, there are at most $\frac{8}{3}2^{2k}$ different ways to choose each of these sequences and the lemma follows. □

Finally, we introduce the latest component of our algorithm. Assume from now on that w and k are fixed constants and that we have a graph G and that $X = [X_1, \dots, X_r]$ is a nice path decomposition of G, with width at most w. A *full set of characteristics* for G_i is a set $FS(i)$ of X_i-characteristics of vertex orderings of the graph G_i with cutwidth at most k, such that for each vertex ordering l of G_i with cutwidth at most k, there is a vertex ordering l' of G_i such that $C_{X_i}(G_i, l') \prec C_{X_i}(G_i, l)$ and $C_{X_i}(G_i, l') \in FS(i)$, i.e., the X_i-characteristic of l' is in $FS(i)$. The following lemma can be derived directly from the definitions.

Lemma 16. *A full set of characteristics for the graph G_i is non-empty if and only if the cutwidth of G_i is at most k. If some full set of characteristics for G_i is non-empty, then any full set of characteristics for G_i is non-empty.*

An important consequence of Lemma 16 is that the cutwidth of G is at most k, if and only if any full set of characteristics of $G_r = G$ is non-empty. The general lines of the algorithm Check-Cutwidth are given in Fig. 3. In what follows, we complement the schema showing how to compute a full set of characteristics at a node X_i in $O(1)$ time, when a full set of characteristics for G_{i-1} is given ($i \geqslant 2$). Therefore the algorithm Check-Cutwidth finishes in $O(n)$ time.

```
Algorithm Check-Cutwidth($G, X, k$).

Input: A graph $G$, a path decomposition $X$ of $G$ with width $w$, and an integer $k$.
Output: Whether cutwidth($G$) $\leqslant k$.

  1: Compute a full set $F_1$ of $X_1$-characteristics for $G_1$.
  2: For any $i$, $1 < i \leqslant r$, compute a full set of $X_i$-characteristics $F_i$ for $G_i$ using $F_{i-1}$ and $X_i$'s type.
  3: If $F_r \neq \emptyset$ then output "cutwidth($G$) $\leqslant k$", otherwise output "cutwidth($G$) $> k$".
  4: End.
```

Fig. 3. Algorithm Check-Cutwidth.

3.1. A full set for a start node

We first give a full set of characteristics for G_1. Clearly, G_1 consists only of the unique vertex in $X_1 = \{x_{\text{start}}\}$ and a full set of characteristics is $\{[x_{\text{start}}], [[0], [0]]\}$.

3.2. A full set for an introduce node

We will now consider the case where X_i is an *introduce* node. The procedure Ins, depicted in Fig. 4, is the basis to compute a X_i-characteristic after the insertion of the introduced vertex and the additional edges that appear in G_i at some point j of a layout for G_{i-1}.

The following lemma is a direct consequence of the definitions of $\mathbf{Q}_{G,l}$, $\mathbf{Q}_{G',l'}$ and the insertion procedure. Recall that the sequences in $\mathbf{Q}_{G,l}$ and $\mathbf{Q}_{G',l'}$ are sequences consisting of only one element counting the number of "crossing edges" in the "gaps" of l and l' respectively.

Lemma 17. *Let G be a graph, let l be a layout of G, and let γ be an integer where $0 \leqslant \gamma \leqslant |l|$. If $G' = G +^u N$ for some $N \subseteq V(G)$ and $u \notin V(G)$, then* $\mathsf{Ins}(G, u, V(G), N, l, \mathbf{Q}_{G,l}, \gamma, 1)$ *is the $V(G')$-characteristic of the layout $l' = l[1,\gamma] \cdot [u] \cdot l[\gamma+1, |l|]$ of G', that is* $\mathsf{Ins}(G, u, V(G), N, l, \mathbf{Q}_{G,l}, \gamma, 1) = (l', \mathbf{Q}_{G',l'})$*, where* Ins *is the procedure described in Fig.* 4.

```
Procedure Ins($G, u, S, N, \lambda, \mathbf{A}, j, m$).

Input: A graph $G$, a vertex $u \notin V(G)$, two sets $S$ and $N$, $N \subseteq S \subseteq V(G)$, a $S$-characteristic $(\lambda, \mathbf{A})$ of some layout
    of $G$, an integer $j$, $0 \leqslant j \leqslant |\lambda|$, and an integer $m$, $1 \leqslant m \leqslant |\mathbf{A}(j)|$.
Output: An $(S \cup \{u\})$-characteristic $(\lambda', \mathbf{A}')$ of some vertex ordering of $G' = G +^u N$ where $0 \leqslant \gamma \leqslant |l|$.

  Assume that $\lambda = [u_1, \ldots, u_\rho]$, and $\lambda[N] = [u_{j_1}, \ldots, u_{j_\sigma}]$.

1: (Insertion of $u$) Set $\lambda' = \lambda[1, j] \cdot [u] \cdot \lambda[j+1, \rho]$ and $\mathbf{A}' = \mathbf{A}[0, j-1] \cdot [\mathbf{A}(j)[1, m]] \cdot [\mathbf{A}(j)[m, |\mathbf{A}(j)|]] \cdot$
   $\mathbf{A}[j+1, \rho]$.
2: (Insertion of the edges from $u$) for $h = 1$ to $\sigma$ do
     (i) If $j_h \leqslant j$ then set $\mathbf{A}' \leftarrow \mathbf{A}'[0, j_h - 1] \cdot (\mathbf{A}'[j_h, j] + 1) \cdot \mathbf{A}'[j+1, \rho+1]$.
     (ii) If $j_h \geqslant j+1$ then set $\mathbf{A}' \leftarrow \mathbf{A}'[0, j] \cdot (\mathbf{A}'[j+1, j_h] + 1) \cdot \mathbf{A}'[j_h+1, \rho+1]$.
3: Output $(\lambda', \mathbf{A}')$.
4: End.
```

Fig. 4. Procedure Ins.

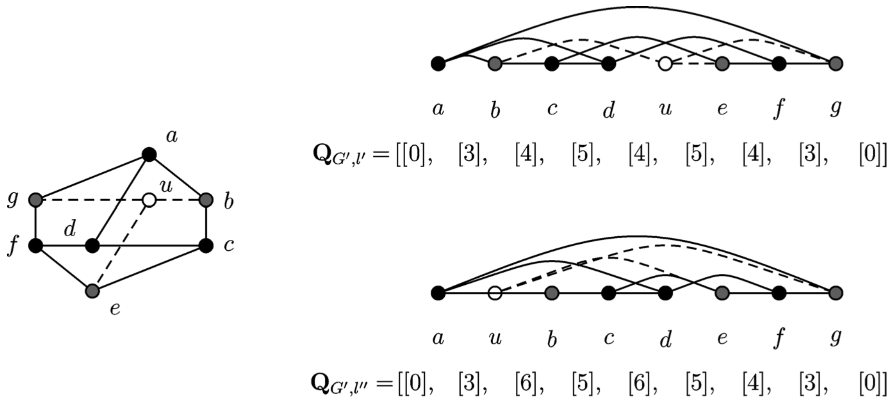
a
u
g
b
d
f
c
e
a b c d u e f g
$\mathbf{Q}_{G',l'} = [[0], [3], [4], [5], [4], [5], [4], [3], [0]]$
a u b c d e f g
$\mathbf{Q}_{G',l''} = [[0], [3], [6], [5], [6], [5], [4], [3], [0]]$

Fig. 5. Examples of Lemma 17 where G is the graph depicted in Fig. 1, $G' = G +^u N$, $l = [a, b, c, d, e, f, g]$, and $N = \{b, e, g\}$. Layouts l' and l'' correspond to $\gamma = 4$ and $\gamma = 1$ respectively.

In Fig. 5 we give two examples illustrating Lemma 17 for $\gamma = 4$ (see layout l') and $\gamma = 1$ (see layout l'').

The following lemma supports the correctness of the output of the procedure Ins and constitutes the main tool for the construction of a full set of characteristics for an *introduce* node.

Lemma 18. *Let G be a graph, $N \subseteq S$ be two subsets of $V(G)$, l be a vertex ordering of G, $(\lambda, \mathbf{A}) = C_S(l, \mathbf{Q}_{G,l})$ be the S-characteristic of l and $G' = G +^u N$ where $u \notin V(G)$. Then the following hold.*

(i) *For any j, $0 \leqslant j \leqslant |\lambda|$, and m, $1 \leqslant m \leqslant |\mathbf{A}(j)|$, there exists an integer γ, $0 \leqslant \gamma \leqslant |l|$, such that*

$$\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}, j, m) = \mathsf{Com}\big(\mathsf{Ins}\big(G, u, V(G), N, l, \mathbf{Q}_{G,l}, \gamma, 1\big), S \cup \{u\}\big).$$

(ii) *For any γ, $0 \leqslant \gamma \leqslant |l|$, there exist two integers j, $0 \leqslant j \leqslant |\lambda|$, and m, $1 \leqslant m \leqslant |\mathbf{A}(j)|$, such that*

$$\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}, j, m) \prec \mathsf{Com}\big(\mathsf{Ins}\big(G, u, V(G), N, l, \mathbf{Q}_{G,l}, \gamma, 1\big), S \cup \{u\}\big).$$

Proof. Let us start by proving part (i). Recall that $(\lambda, \mathbf{A}) = \mathsf{Com}(l, \mathrm{Q}_{G,l}, S)$. Let $l = [v_1, \ldots, v_{|l|}]$ and $\lambda = [v_{i_1}, \ldots, v_{i_\rho}]$. From the definition of the procedure Com we have that $\forall_{h, 0 \leqslant h \leqslant \rho}\ \mathbf{A}(h) = \tau(\mathrm{Q}_{G,l}[i_h, i_{h+1} - 1])$ (for convenience, we set $i_0 = 0$). It is now easy to verify that

$$\mathbf{A}[0, j-1] = \big[\tau\big(\mathbf{Q}_{G,l}[0, i_1 - 1]\big), \tau\big(\mathbf{Q}_{G,l}[i_1, i_2 - 1]\big), \ldots, \tau\big(\mathbf{Q}_{G,l}[i_{j-1}, i_j - 1]\big)\big], \tag{1}$$

$$\mathbf{A}(j) = \tau\big(\mathbf{Q}_{G,l}[i_j, i_{j+1} - 1]\big), \tag{2}$$

$$\mathbf{A}[j+1,\rho] = \bigl[\tau\bigl(\mathbf{Q}_{G,l}[i_{j+1}, i_{j+2}-1]\bigr), \ldots, \tau\bigl(\mathbf{Q}_{G,l}[i_{\rho-1}, i_\rho - 1]\bigr),$$
$$\tau\bigl(\mathbf{Q}_{G,l}\bigl[i_\rho, |l|\bigr]\bigr)\bigr]. \tag{3}$$

Applying now Lemma 10 to (2), we have that there exists γ, $i_j \leqslant \gamma < i_{j+1}$, such that

$$\mathbf{A}(j)[1,m] = \tau\bigl(\mathbf{Q}[i_j,\gamma]\bigr), \tag{4}$$
$$\mathbf{A}(j)\bigl[m, \bigl|\mathbf{A}(j)\bigr|\bigr] = \tau\bigl(\mathbf{Q}[\gamma, i_{j+1}-1]\bigr). \tag{5}$$

Observe that, as $i_j \leqslant \gamma < i_{j+1}$, the following hold

$$l[1,\gamma][S] = \lambda[1,j], \tag{6}$$
$$l\bigl[\gamma+1,|l|\bigr][S] = \lambda[j+1,\rho]. \tag{7}$$

Let $(\lambda', \mathbf{A}')$ and $(l', \mathbf{R})$ be the characteristics constructed by step **1** of the procedures $\mathsf{Ins}(G,u,S,N,\lambda,\mathbf{A},j,m)$ and $\mathsf{Ins}(G,u,V(G),N,l,\mathbf{Q}_{G,l},\gamma,1)$ respectively. We will prove now that $(\lambda',\mathbf{A}') = \mathsf{Com}((l',\mathbf{R}),S)$. Notice first that $\lambda' = \lambda[1,j]\cdot[u]\cdot\lambda[j+1,|\rho|]$ and $l' = l[1,\gamma]\cdot[u]\cdot l[\gamma+1,|l|]$. Using now (6) and (7), it follows that $\lambda' = l'[S\cup\{u\}] = [v_{i_1},\ldots,v_{i_j},u,v_{i_{j+1}},\ldots,v_{i_\rho}]$. Notice now that

$$\mathbf{A}' = \mathbf{A}[0,j-1]\cdot\bigl[\mathbf{A}(j)[1,m]\bigr]\cdot\bigl[\mathbf{A}(j)\bigl[m,\bigl|\mathbf{A}(j)\bigr|\bigr]\bigr]\cdot\mathbf{A}[j+1,\rho], \tag{8}$$
$$\mathbf{R} = \mathbf{Q}_{G,l}[0,i_j-1]\cdot\mathbf{Q}_{G,l}[i_j,\gamma]\cdot\mathbf{Q}_{G,l}[\gamma,i_{j+1}-1]\cdot\mathbf{Q}_{G,l}\bigl[i_{j+1},|l|\bigr]. \tag{9}$$

Taking now in mind (9) and the fact that $\lambda' = [v_{i_1},\ldots,v_{i_j},u,v_{i_{j+1}},\ldots,v_{i_\rho}]$, we can conclude that the sequence of typical sequences in the output of $\mathsf{Com}(l,\mathbf{R},S\cup\{u\})$ is

$$\bigl[\tau\bigl(\mathbf{Q}_{G,l}[0,i_1-1]\bigr),\ldots,\tau\bigl(\mathbf{Q}_{G,l}[i_{j-1},i_j-1]\bigr),\tau\bigl(\mathbf{Q}_{G,l}[i_j,\gamma]\bigr),$$
$$\tau\bigl(\mathbf{Q}_{G,l}[\gamma,i_{j+1}-1]\bigr),\ldots,\tau\bigl(\mathbf{Q}_{G,l}\bigl[i_\rho,|l|\bigr]\bigr)\bigr]. \tag{10}$$

Using now (1), (3), (4), and (5) we have that the sequence of typical sequences in (10) is equal to $\mathbf{A}'$ in the form it is presented in (8).

So far, we have seen that $(\lambda',\mathbf{A}') = \mathsf{Com}((l',\mathbf{R}),S\cup\{u\})$. In what follows we will prove that this relation is invariant under the transformations applied to $(\lambda',\mathbf{A}')$ and $(l',\mathbf{R})$ during step **2** of the insertion procedure.

Notice that during step **2**, no vertex is introduced, only new edges are taken into account and added, therefore the respective vertex orderings do not change. We will use the notation $(\lambda',\mathbf{A}^{(h)})$ and $(l',\mathbf{R}^{(h)})$ for the contents of $\mathbf{A}$ and $\mathbf{R}$ at the end of the hth execution of the loop in step **2** of $\mathsf{Ins}(G,u,S,N,\lambda,\mathbf{A},j,m)$ and $\mathsf{Ins}(G,u,V(G),N,l,\mathbf{Q}_{G,l},\gamma,1)$ respectively. And for convenience we set $(\lambda',\mathbf{A}') = (\lambda',\mathbf{A}^{(0)})$, and $(l',\mathbf{R}) = (l',\mathbf{R}^{(0)})$. To see that for any h $(\lambda',\mathbf{A}^{(h)}) = \mathsf{Com}((l',\mathbf{R}^{(h)}),S\cup\{u\})$ we proceed by induction.

Suppose that $(\lambda',\mathbf{A}^{(h)}) = \mathsf{Com}(l',\mathbf{R}^{(h)},S\cup\{u\})$ for any h, $0 < h < \xi$. It remains to prove that $(\lambda',\mathbf{A}^{(\xi)}) = \mathsf{Com}(l',\mathbf{R}^{(\xi)},S\cup\{u\})$.

Assume that $\mu_1 = l[N\cap V(l[1,\gamma])]$ and $\mu_2 = l[N\cap V(l[\gamma+1,|l|])]$. Then we have

$$\begin{aligned}
\mathrm{rank}_l(v) &= \mathrm{rank}_{l'}(v), && \text{for } v\in V(\mu_1),\\
\mathrm{rank}_l(v)+1 &= \mathrm{rank}_{l'}(v), && \text{for } v\in V(\mu_2),\\
\mathrm{rank}_\lambda(v) &= \mathrm{rank}_{\lambda'}(v), && \text{for } v\in V(\mu_1),\\
\mathrm{rank}_\lambda(v)+1 &= \mathrm{rank}_{\lambda'}(v), && \text{for } v\in V(\mu_2),
\end{aligned}$$

$$\begin{aligned}
\mathrm{rank}_{l'}(u) &= \gamma + 1,\\
\mathrm{rank}_{\lambda'}(u) &= j + 1,\\
\lambda &= \mu_1 \cdot \mu_2,\\
\lambda' &= \mu_1 \cdot [u] \cdot \mu_2.
\end{aligned}$$

Assume that the vertex to be taken from N in this step has rank j_ξ in l and rank ξ in λ. In the case that $j_\xi \leqslant j$ we have that

$$\mathbf{R}^{(\xi)} = \mathbf{R}^{(\xi-1)}[0, j_\xi - 1] \cdot \big(\mathbf{R}^{(\xi-1)}[j_\xi, \gamma] + 1\big) \cdot \mathbf{R}^{(\xi-1)}\big[\gamma + 1, |l| + 1\big], \tag{11}$$

$$\mathbf{A}^{(\xi)} = \mathbf{A}^{(\xi-1)}[0, \xi - 1] \cdot \big(\mathbf{A}^{(\xi-1)}[\xi, j] + 1\big) \cdot \mathbf{A}^{(\xi-1)}[j + 1, \rho + 1]. \tag{12}$$

As $j_\xi = \mathrm{rank}_{l'}(u), \gamma + 1 = \mathrm{rank}_{\lambda'}(u), \xi = \mathrm{rank}_{l'}(u_{j_\xi})$ and $j + 1 = \mathrm{rank}_{\lambda'}(u_{j_\xi})$, Lemma 14 implies that $(\lambda^{(\xi)}, \mathbf{A}^{(\xi)}) = \mathsf{Com}(l^{(\xi-1)}, \mathbf{R}^{(\xi)}, S \cup \{u\})$. In the case $j_\xi \geqslant j + 1$ we have that

$$\mathbf{R}^{(\xi)} = \mathbf{R}^{(\xi-1)}[0, \gamma] \cdot \big(\mathbf{R}^{(\xi-1)}[\gamma + 1, j_\xi]\big) \cdot \mathbf{R}^{(\xi-1)}\big[j_\xi + 1, |l| + 1\big], \tag{13}$$

$$\mathbf{A}^{(\xi)} = \mathbf{A}^{(\xi-1)}[0, j] \cdot \big(\mathbf{A}^{(\xi-1)}[j + 1, \xi] + 1\big) \cdot \mathbf{A}^{(\xi-1)}[\xi + 1, \rho + 1]. \tag{14}$$

As $\gamma + 1 = \mathrm{rank}_{l'}(u)$, $j_\xi + 1 = \mathrm{rank}_{l'}(u_{j_\xi})$, $j + 1 = \mathrm{rank}_{\lambda'}(u)$, and $\xi + 1 = \mathrm{rank}_{\lambda'}(u_{j_\xi})$, Lemma 14 implies that $(\lambda^{(\xi)}, \mathbf{A}^{(\xi)}) = \mathsf{Com}(l^{(\xi-1)}, \mathbf{R}^{(\xi)}, S \cup \{u\})$ and this completes the proof of (i). The arguments to prove (ii) are exactly the same as in the proof of (i) with the difference that now weaker versions of (1), (3), (4), and (5) are required. As $(\lambda, \mathbf{A}) = \mathsf{Com}(l, \mathrm{Q}_{G,l}, S)$, we can assume again that $l = [v_1, \ldots, v_{|l|}]$ and $\lambda = [v_{i_1}, \ldots, v_{i_\rho}]$. From the definition procedure Com there exists j, $0 \leqslant j \leqslant |\rho|$, such that $i_j \leqslant \gamma < i_{j+1}$ (for convenience, we set $i_0 = 0$ and $i_{\rho+1} = |l| + 1$). Notice now that for this choice of j, (1)–(3) hold and (1) and (3) can be rewritten in the following weaker form.

$$\begin{aligned}
\mathbf{A}[0, j - 1] \prec \big[&\tau\big(\mathbf{Q}_{G,l}[0, i_1 - 1]\big), \tau\big(\mathbf{Q}_{G,l}[i_1, i_2 - 1]\big), \ldots,\\
&\tau\big(\mathbf{Q}_{G,l}[i_{j-1}, i_j - 1]\big)\big],
\end{aligned} \tag{15}$$

$$\begin{aligned}
\mathbf{A}[j + 1, \rho] \prec \big[&\tau\big(\mathbf{Q}_{G,l}[i_{j+1}, i_{j+2} - 1]\big), \ldots, \big(\mathbf{Q}_{G,l}[i_{\rho-1}, i_\rho - 1]\big),\\
&\tau\big(\mathbf{Q}_{G,l}\big[i_\rho, |l|\big]\big)\big].
\end{aligned} \tag{16}$$

From Lemma 13 there exists an integer m, $1 \leqslant m \leqslant \mathbf{A}(j)$, where

$$\mathbf{A}(j)[1, m] \prec \tau\big(\mathbf{Q}[i_j, \gamma]\big), \tag{17}$$

$$\mathbf{A}(j)\big[m, \big|\mathbf{A}(j)\big|\big] \prec \tau\big(\mathbf{Q}[\gamma, i_{j+1} - 1]\big). \tag{18}$$

Notice that (15), (16), (17), and (18) are the same as (1), (3), (4), and (5) with the difference that "=" has been replaced by "$\prec$". From this fact, using the same proof as in case (i), it follows that $\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}, j, m) \prec \mathsf{Com}(\mathsf{Ins}(G, u, V(G), N, l, \mathbf{Q}_{G,l}, \gamma, 1), S \cup \{u\})$. $\square$

As an example for Lemma 18, we consider the graphs G and G' and their vertex orderings l, l' and l'' as depicted in Fig. 5. If we set $S = N \cup \{d\} = \{b, d, e, g\}$ we have that $C_S(G, l) = (\lambda, \mathbf{A})$ is the characteristic $[0, 3]$ ***b*** $[3, 4]$ ***d*** $[3]$ ***e*** $[3, 2]$ ***g*** $[0]$. Notice that in l', if $j = 2$ and $m = 1$, Lemma 18(i) gives $\gamma = 4$. Moreover, in l'', if $\gamma = 1$, Lemma 18(ii) gives $j = 0$ and $m = 2$.

We will now prove some monotonicity properties of the insertion procedure.

Lemma 19. *Let $(\lambda, \mathbf{A}_i)$, $i = 1, 2$, be two characteristics of a graph G where $(\lambda, \mathbf{A}_2) \prec (\lambda, \mathbf{A}_1)$. Let N, S be subsets of $V(G)$ and let $G' = G +^u N$ for some $u \notin V(G)$. Then for any j, $0 \leqslant j \leqslant |\mathbf{A}_1|$, and any m_1, $1 \leqslant m_1 \leqslant |\mathbf{A}_1(j)|$, there exists m_2, $1 \leqslant m_2 \leqslant |\mathbf{A}_2(j)|$, such that*

$$\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}_2, j, m_2) \prec \mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}_1, j, m_1).$$

Proof. As $(\lambda, \mathbf{A}_2) \prec (\lambda, \mathbf{A}_1)$, for any j, we have $\mathbf{A}_2(j) \prec \mathbf{A}_1(j)$. Therefore, Lemma 11 implies that there exists m_2, $1 \leqslant m_2 \leqslant |\mathbf{A}_2(j)|$, such that

$$\mathbf{A}_2(j)[1, m_2] \prec \mathbf{A}_1(j)[1, m_1] \quad \text{and} \quad \mathbf{A}_2(j)\bigl[m_2, |\mathbf{A}_2(j)|\bigr] \prec \mathbf{A}_1(j)\bigl[m_1, |\mathbf{A}_1(j)|\bigr].$$

Moreover, $\forall_{i, 0 \leqslant i \leqslant j-1}\ \mathbf{A}_2(i) \prec \mathbf{A}_1(i)$ and $\forall_{i, j+1 \leqslant i \leqslant \rho}\ \mathbf{A}_2(i) \prec \mathbf{A}_1(i)$. Therefore, if

$$\mathbf{A}'_i = \mathbf{A}_i[0, j-1] \cdot \bigl[\mathbf{A}_i(j)[1, m_i]\bigr] \cdot \bigl[\mathbf{A}_i(j)\bigl[m_1, |\mathbf{A}_i(j)|\bigr]\bigr] \cdot \mathbf{A}_i[j+1, \rho],$$

for $i = 1, 2$, then $\mathbf{A}'_2 \prec \mathbf{A}'_1$. Observe that, $\mathbf{A}'_1$ and $\mathbf{A}'_2$ are the sequences of typical sequences created after step **1** of the computation of $\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}_1, j, m_1)$ and $\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}_2, j, m_2)$ respectively. Moreover, the vertex orderings constructed after step **1** of $\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}_1, j, m_1)$ and $\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}_2, j, m_2)$ are both identical and we denote them λ'. It now remains to prove that after step **2**, $\mathbf{A}'_2 \prec \mathbf{A}'_1$. We proceed by induction.

For $i = 1, 2$ let $\mathbf{A}_i^{(h)}$ denote the value of $\mathbf{A}'_i$ after the hth execution of the loop in step **2** of the computation of $\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}_i, j, m_i)$, $i = 1, 2$, and set $\mathbf{A}_i^{(0)} = \mathbf{A}'_i$. Suppose that $\mathbf{A}_2^{(h)} \prec \mathbf{A}_1^{(h)}$ for any h, $0 < h < \xi$. It remains to prove that $\mathbf{A}_2^{(\xi)} \prec \mathbf{A}_1^{(\xi)}$. Assume that the vertex of N considered in this step has rank j_ξ in l and ξ in λ'. We examine only the case where $j_\xi \leqslant j$ as the case where $j_\xi \geqslant j + 1$ is analogous. By induction hypothesis $\mathbf{A}_2^{(\xi-1)} \prec \mathbf{A}_1^{(\xi-1)}$, therefore

$$\mathbf{A}_2^{(\xi-1)}[0, j_\xi - 1] \prec \mathbf{A}_1^{(\xi-1)}[0, j_\xi - 1], \tag{19}$$

$$\mathbf{A}_2^{(\xi-1)}[j_\xi, j] \prec \mathbf{A}_1^{(\xi-1)}[j_\xi, j], \tag{20}$$

$$\mathbf{A}_2^{(\xi-1)}[j, \rho + 1] \prec \mathbf{A}_1^{(\xi-1)}[j, \rho + 1]. \tag{21}$$

Clearly, (20) is equivalent to

$$\mathbf{A}_2^{(\xi-1)}[j_\xi, j] + 1 \prec \mathbf{A}_1^{(\xi-1)}[j_\xi, j] + 1. \tag{22}$$

Taking into account that, for $i = 1, 2$,

$$\mathbf{A}_i^{\xi} = \mathbf{A}_i^{(\xi-1)}[0, j_\xi - 1] \cdot \bigl(\mathbf{A}_i^{(\xi-1)}[j_\xi, j] + 1\bigr) \cdot \mathbf{A}_i^{(\xi-1)}[j, \rho + 1],$$

we conclude $\mathcal{A}_2^{(\xi)} \prec \mathcal{A}_1^{(\xi)}$ as a consequence of (19), (22), and (21). $\square$

We now give an algorithm (algorithm $\mathsf{Introduce\text{-}Node}$) given in Fig. 6 that, for any introduce node X_i, computes a full set of characteristics for the graph G_i, given a full set of characteristics for the graph G_{i-1}. Now we prove the correctness of the previous algorithm.

Algorithm Introduce-Node.

Input: A full set of characteristics $FS(i-1)$ for G_{i-1}.
Output: A full set of characteristics $FS(i)$ for G_i.

```
1: Initialize $FS(i)=\emptyset$ and set $\rho=|X_{i-1}|$, $\{u\}=X_i-X_{i-1}$, and $N=N_{G_i}(u)$.
2: For any $X_{i-1}$-characteristic $(\lambda,\mathbf{A})\in FS(i-1)$ do
3:     for $j=0$ to $\rho$ do
4:         for $m=1$ to $|\mathbf{A}(j)|$ do
5:             Let $(\lambda',\mathbf{A}')=\mathsf{Ins}(G_i,u,X_{i-1},N,\lambda,\mathbf{A},j,m)$.
               If $\max(\mathbf{A}')\leqslant k$, then set $FS(i)\leftarrow FS(i)\cup\{(\lambda',\mathbf{A}')\}$.
6: Output $FS(i)$.
7: End.
```

Fig. 6. The algorithm to compute a full set of characteristics for an introduce node.

Lemma 20. *Given a nice path decomposition $X=[X_1,\dots,X_r]$ of a graph G. If X_i is an introduce node and $FS(i-1)$ is a full set of characteristics for the graph $G_{i-1}=G[\bigcup_{1\leqslant j\leqslant i-1}X_j]$, then the set $FS(i)$ constructed by the* Introduce-Node *algorithm is a full set of characteristics for the graph $G_i=G[\bigcup_{1\leqslant j\leqslant i}X_j]$.*

Proof. We prove first that $FS(i)$ is a set of X_i-characteristics for the graph G_i. We show that for any $(\lambda',\mathbf{A}')\in FS(i)$ there exists a vertex ordering l' of G_i where $(\lambda',\mathbf{A}')=C_{X_i}(G_i,l')$. Clearly, as $(\lambda',\mathbf{A}')$ was constructed by the algorithm Introduce-Node, there must be a characteristic $(\lambda,\mathbf{A})\in FS(i-1)$ and two integers j, $0\leqslant j\leqslant|\lambda|$, and m, $1\leqslant m\leqslant|\mathbf{A}(j)|$, such that

$$\bigl(\lambda',\mathbf{A}'\bigr)=\mathsf{Ins}(G_{i-1},u,X_{i-1},N,\lambda,\mathbf{A},j,m). \tag{23}$$

As $(\lambda,\mathbf{A})$ is a X_{i-1}-characteristic for G_{i-1}, there exists a vertex ordering l of G_{i-1} of cutwidth at most k where $(\lambda,\mathbf{A})=C_{X_{i-1}}(G_{i-1},l)$. From part (i) in Lemma 18, we have that there exists an integer γ, $0\leqslant\gamma\leqslant|l|$, such that

$$\begin{aligned}&\mathsf{Ins}(G_{i-1},u,X_{i-1},N,\lambda,\mathbf{A},j,m)\\&\quad=\mathsf{Com}\bigl(\mathsf{Ins}\bigl(G_{i-1},u,V(G_{i-1}),N,l,\mathbf{Q}_{G_{i-1},l},\gamma,1\bigr),X_{i-1}\cup\{u\}\bigr).\end{aligned} \tag{24}$$

From Lemma 17, we have that $\mathsf{Ins}(G_{i-1},u,V(G_{i-1}),N,l,\mathbf{Q}_{G_{i-1},l},\gamma,1)$ is the $V(G_i)$-characteristic of $l=l(1,\gamma)\cdot[u]\cdot l(\gamma+1,|l|)$ and therefore,

$$\mathsf{Com}\bigl(\mathsf{Ins}\bigl(G_{i-1},u,V(G_{i-1}),N,l,\mathbf{Q}_{G_{i-1},l},\gamma,1\bigr),X_i\bigr)=\mathsf{Com}\bigl(l',\mathbf{Q}_{G',l'},X_i\bigr). \tag{25}$$

Combining now (23), (24), and (25), we have that $(\lambda',\mathbf{A}')=\mathsf{Com}(l',\mathbf{Q}_{G',l'},X_i)=C_{X_i}(G_i,l')$. It remains to prove that $FS(i)$ is a full set of characteristics for G_i. Let l' be a vertex ordering of G_i with cutwidth at most k. Let us show that there exists a vertex ordering l'_* of G_i such that $C_{X_i}(G_i,l'_*)\prec C_{X_i}(G_i,l')$ and $C_{X_i}(G_i,l'_*)\in FS(i)$. Setting $\gamma=\mathrm{rank}_{l'}(u)-1$ and $l=l'[1,\gamma]\cdot l'[\gamma+2,|l|]$, from Lemma 17, we have that

$$\mathsf{Ins}\bigl(G_{i-1},u,V(G_{i-1}),N,l,\mathbf{Q}_{G_{i-1},l},\gamma,1\bigr)=\bigl(l',\mathbf{Q}_{G_i,l'}\bigr)$$

and therefore,

$$\begin{aligned}&\mathsf{Com}\big(\mathsf{Ins}\big(G_{i-1}, u, V(G_{i-1}), N, l, \mathbf{Q}_{G_{i-1},l}, \gamma, 1\big), X_i\big)\\ &\quad = \mathsf{Com}(l, \mathbf{Q}_{G_i,l'}, X_i) = C_{X_i}\big(G_i, l'\big).\end{aligned} \tag{26}$$

Set now $(\lambda, \mathbf{A}) = C_{X_i}(G_i, l)$. From part (ii) of Lemma 18 we have that there are values j and m, $0 \leqslant j \leqslant |\lambda|$ and $1 \leqslant m \leqslant |\mathbf{A}(j)|$, such that

$$\begin{aligned}&\mathsf{Ins}(G_{i-1}, u, X_{i-1}, N, \lambda, \mathbf{A}, j, m)\\ &\quad \prec \mathsf{Com}\big(\mathsf{Ins}\big(G_{i-1}, u, V(G_{i-1}), N, l, \mathbf{Q}_{G_{i-1},l}, \gamma, 1\big), X_{i-1} \cup \{u\}\big).\end{aligned} \tag{27}$$

As $FS(i-1)$ is a full set of characteristics, there exists a vertex ordering l_* of $V(G_{i-1})$ for which

$$C_{X_{i-1}}(G_{i-1}, l_*) \prec C_{X_{i-1}}(G_{i-1}, l) \quad \text{and} \quad C_{X_{i-1}}(G_{i-1}, l_*) \in FS(i-1).$$

Let $(\lambda_*, \mathbf{A}_*) = C_{X_{i-1}}(G_{i-1}, l_*)$. From Lemma 19, we have that there exists a m_* such that

$$\mathsf{Ins}(G_{i-1}, u, X_{i-1}, N, \lambda_*, \mathbf{A}_*, j, m_*) \prec \mathsf{Ins}(G_{i-1}, u, X_{i-1}, N, \lambda, \mathbf{A}, j, m). \tag{28}$$

From part (i) of Lemma 18, there exists γ_*, $0 \leqslant \gamma_* \leqslant |l_*|$, such that

$$\begin{aligned}&\mathsf{Com}\big(\mathsf{Ins}\big(G_{i-1}, u, V(G_{i-1}), N, l_*, \mathbf{Q}_{G_{i-1},l_*}, \gamma_*, 1\big), X_i\big)\\ &\quad = \mathsf{Ins}(G_{i-1}, u, X_{i-1}, N, \lambda_*, \mathbf{A}_*, j, m_*).\end{aligned} \tag{29}$$

Defining $l'_* = l_*[1, \gamma_*] \cdot [u] \cdot l_*[\gamma_* + 1, |l_*|]$ and applying Lemma 17 we have that

$$\big(l'_*, \mathbf{Q}_{G_i,l'_*}\big) = \mathsf{Ins}\big(G_{i-1}, u, V(G_{i-1}), N, l_*, \mathbf{Q}_{G_{i-1},l_*}, \gamma_*, 1\big)$$

and therefore,

$$\begin{aligned}C_{X_i}\big(G_i, l'_*\big) &= \mathsf{Com}\big(l'_*, \mathbf{Q}_{G_i,l'_*}, X_i\big)\\ &= \mathsf{Com}\big(\mathsf{Ins}\big(G_{i-1}, u, V(G_{i-1}), N, l_*, \mathbf{Q}_{G_{i-1},l_*}, \gamma_*, 1\big), X_i\big).\end{aligned} \tag{30}$$

From (29) and (30) we have that $C_{X_i}(G_i, l'_*) = \mathsf{Ins}(G_{i-1}, u, X_{i-1}, N, \lambda_*, \mathbf{A}_*, j, m_*)$. Since $(\mathbf{Q}_{G_{i-1}}, l_*) \in FS(i-1)$ we obtain $C_{X_i}(G_i, l'_*) \in FS(i)$. Finally, combining relations (26)–(30) we conclude that $C_{X_i}(G_i, l'_*) \prec C_{X_i}(G_i, l')$. □

3.3. A full set for a forget node

We now consider the case where X_i is a *forget* node. We provide an algorithm that given a full set of characteristics $FS(i-1)$ for X_{i-1}, computes a full set of characteristics $FS(i)$ for X_i. We start by defining the deletion procedure Del (see Fig. 7) that operates inversely to the insertion procedure Ins.

The following lemma is a direct consequence of the definitions of the procedures Com and Del.

Lemma 21. *Let* $(l, \mathbf{R})$ *be a characteristic of a given graph* G *and let* $V \subseteq V(l)$. *Then, for any* $v \in V$, $\mathsf{Com}(l, \mathbf{R}, V - \{v\}) = \mathsf{Del}(\mathsf{Com}(l, \mathbf{R}, V), v)$.

Observe that Lemma 21 provides an alternative recursive definition of the procedure Com, based on the procedure Del.

The following monotonicity result is a direct consequence of Lemma 12.

Procedure Del$(G, v, S, \lambda, \mathbf{A})$.

Input: A graph G, a vertex v, a set $S \subseteq V(G)$ an S-characteristic $(\lambda, \mathbf{A})$, where $v \in V(\lambda)$, of some vertex ordering of G.
Output: An S-characteristic $(\lambda', \mathbf{A}')$.

Assume that $\lambda = [u_1, \ldots, u_\rho]$ and $j = \mathrm{rank}_\lambda(v)$.

```
1: $\lambda' \leftarrow \lambda(1, j-1) \cdot \lambda(j+1, \rho)$.
2: $\mathbf{A}' \leftarrow \mathbf{A}[0, j-2] \cdot [\tau(\mathbf{A}(j-1) \cdot \mathbf{A}(j))] \cdot \mathbf{A}[j+1, \rho]$.
3: Output $(\lambda', \mathbf{A}')$.
4: End.
```

Fig. 7. Procedure Del.

Lemma 22. *Let $(l_i, \mathbf{R}_i)$, $i = 1, 2$, be two characteristic of a given graph G. If $(l_2, \mathbf{R}_2) \prec (l_1, \mathbf{R}_1)$, then for any $u \in V(l_1)$, $\mathsf{Del}(l_2, \mathbf{R}_2, u) \prec \mathsf{Del}(l_1, \mathbf{R}_1, u)$.*

Now we can give an algorithm (algorithm Forget-Node given in Fig. 8) that, for any forget node X_i, computes a full set of characteristics for the graph G_i, given a full set of characteristics for the graph G_{i-1}.

Lemma 23. *If $FS(i-1)$ is a full set of X_{i-1}-characteristics then the set $FS(i)$ constructed by the* Forget-Node *algorithm is a full set of X_i-characteristics for G_i.*

Proof. Let $u \in X_{i-1}$ be the forgotten vertex. As $G_i = G_{i-1}$ we will use the unifying notation $\widehat{G}$ for both of them. We start proving that $FS(i)$ is a set of X_i-characteristics for $\widehat{G}$. We need to prove that, for any $(\lambda', \mathbf{A}') \in FS(i)$, there exists a vertex ordering l of $\widehat{G}$ where

$$C_{X_i}(l, \widehat{G}) = \mathsf{Com}(l, \mathbf{Q}_{\widehat{G},l}, X_i) = (\lambda', \mathbf{A}').$$

As $(\lambda', \mathbf{A}')$ has been constructed by the procedure Forget-Node there must exists a X_{i-1}-characteristic $(\lambda, \mathbf{A}) \in FS(i-1)$ such that

$$(\lambda', \mathbf{A}') = \mathsf{Del}(\lambda, \mathbf{A}, u). \tag{31}$$

As $(\lambda, \mathbf{A}) \in FS(i-1)$, there exists a vertex ordering l of $\widehat{G}$ such that

$$(\lambda, \mathbf{A}) = \mathsf{Com}(l, \mathbf{Q}_{\widehat{G},l}, X_{i-1}) \tag{32}$$

Algorithm Forget-Node.

Input: A full set of characteristics $FS(i-1)$ for G_{i-1}.
Output: A full set of characteristics $FS(i)$ for G_i.

```
1: Initialize $FS(i) = \emptyset$ and let $u$ be the forget vertex of $G_i$.
2: For any $(\lambda, \mathbf{A}) \in FS(i-1)$ do
3:     $FS(i) \leftarrow FS(i) \cup \{\mathsf{Del}(\lambda, \mathbf{A}, u)\}$.
4: Output $FS(i)$.
5: End.
```

Fig. 8. The algorithm to compute a full set of characteristics for a forget node.

and therefore, from (31) and (32) we have

$$\bigl(\lambda', \mathbf{A}'\bigr) = \mathsf{Del}\bigl(\mathsf{Com}(l, \mathbf{Q}_{\widehat{G},l}, X_{i-1}), u\bigr) \tag{33}$$

and using (33) and Lemma 21 we have that $C_{X_i}(l, \widehat{G}) = \mathsf{Com}(l, \mathbf{Q}_{\widehat{G},l}, X_i) = (\lambda', \mathbf{A}')$.

We will now prove that $FS(i)$ is a full set of X_i-characteristics for $\widehat{G}$. Let l be a vertex ordering of $\widehat{G}$ of cutwidth at most k. We will show that there exists a vertex ordering l_* of $\widehat{G}$ such that

$$C_{X_i}\bigl(\widehat{G}, l_*\bigr) \prec C_{X_i}\bigl(\widehat{G}, l\bigr) \quad \text{and} \quad C_{X_i}\bigl(\widehat{G}, l_*\bigr) \in FS(i).$$

From Lemma 21 we have that

$$C_{X_i}\bigl(\widehat{G}, l\bigr) = \mathsf{Com}(l, \mathbf{Q}_{\widehat{G},l}, X_i) = \mathsf{Del}\bigl(\mathsf{Com}(l, \mathbf{Q}_{\widehat{G},l}, X_{i-1}), u\bigr). \tag{34}$$

As $FS(i-1)$ is a full set of characteristics, there exists a vertex ordering l_* of $V(\widehat{G})$ such that $C_{X_{i-1}}(\widehat{G}, l_*) \in FS(i-1)$ and $C_{X_{i-1}}(\widehat{G}, l_*) \prec C_{X_{i-1}}(\widehat{G}, l)$ or, equivalently,

$$\mathsf{Com}(l_*, \mathbf{Q}_{\widehat{G},l_*}, X_{i-1}) \prec \mathsf{Com}(l, \mathbf{Q}_{\widehat{G},l}, X_{i-1}). \tag{35}$$

Using now Lemma 22 we can rewrite (35) as follows.

$$\mathsf{Del}\bigl(\mathsf{Com}(l_*, \mathbf{Q}_{\widehat{G},l_*}, X_{i-1}), u\bigr) \prec \mathsf{Del}\bigl(\mathsf{Com}(l, \mathbf{Q}_{\widehat{G},l}, X_{i-1}), u\bigr). \tag{36}$$

Applying again Lemma 21 we have that

$$C_{X_i}\bigl(\widehat{G}, l_*\bigr) = \mathsf{Com}(l_*, \mathbf{Q}_{\widehat{G},l_*}, X_i) = \mathsf{Del}\bigl(\mathsf{Com}(l_*, \mathbf{Q}_{\widehat{G},l_*}, X_{i-1}), u\bigr). \tag{37}$$

Combining now (34), (36), and (37), we have that $C_{X_i}(\widehat{G}, l_*) \prec C_{X_i}(\widehat{G}, l)$. Finally as

$$C_{X_{i-1}}\bigl(\widehat{G}, l_*\bigr) = \mathsf{Com}(l_*, \mathbf{Q}_{\widehat{G},l_*}, X_{i-1}) \in FS(i-1),$$

the output of $\mathsf{Del}(\mathsf{Com}(l_*, \mathbf{Q}_{\widehat{G},l_*}, X_{i-1}), u)$ will be one of the characteristics included in $FS(i)$. And we conclude $C_{X_i}(\widehat{G}, l_*) \in FS(i)$. $\square$

4. Computing a vertex ordering

Suppose now that, given a path decomposition $X = [X_1, \ldots, X_r]$ of G with bounded width, after running the algorithm Check-Cutwidth, described in the previous section, we know that a graph G has cutwidth at most k, i.e., the computed set $FS(r)$ is not empty. We describe now a way to further construct a vertex ordering of G with cutwidth at most k, and the modifications and additions needed to perform to the algorithm Check-Cutwidth, we refer to the extended algorithm as Layout-Cutwidth. By observing the execution of the Check-Cutwidth algorithm, it follows that there exist a sequence of characteristics, $(\lambda_1, \mathbf{A}_1), (\lambda_2, \mathbf{A}_2), \ldots, (\lambda_r, \mathbf{A}_r)$, that we call a *witness path*, such that

(1) $(\lambda_1, \mathbf{A}_1) = ([x_{\text{start}}], [[0], [0]])$ is the unique characteristic of the unique vertex ordering of G_1.
(2) $(\lambda_r, \mathbf{A}_r)$ is some characteristic in $FS(r)$, and

(3) for any h, $1 \leqslant h \leqslant n-1$, the characteristic $(\lambda_{h+1}, \mathbf{A}_{h+1})$ was constructed after a call of either Introduce-Node or Forget-Node with input $(\lambda_h, \mathbf{A}_h)$.

Let us show how to compute a layout with cutwidth at most k, in linear time, given a witness path

$$(\lambda_1, \mathbf{A}_1), (\lambda_2, \mathbf{A}_2), \dots, (\lambda_r, \mathbf{A}_r).$$

For any h, $1 < h < r$, let l_h be a vertex ordering such that $C_{X_h}(G_h, l_h) = (\lambda_h, \mathbf{A}_h)$. Assume that

$$\begin{aligned} l_h &= \bigl[v_1^h, \dots, v_{|l_h|}^h\bigr] \quad \text{and} \\ \mathbf{A}_h &= \bigl[A_0^h, \dots, A_{|\mathbf{A}_h|}^h\bigr] \quad \text{where } A_j^h = \bigl[a_1^{h,j}, \dots, a_{|A_h|}^{h,j}\bigr]. \end{aligned}$$

Notice that any element $a_m^{h,j}$ of $A_0^h \cdot \cdots \cdot A_{|\mathbf{A}_h|}^h$ is determined by a pair (j, m) of indices where $0 \leqslant j \leqslant |\mathbf{A}_h|$ and $1 \leqslant m \leqslant |A_j|$. We denote as $\mathcal{P}_h$ the set containing all these pairs.

Let κ be the minimum number such that $h < \kappa \leqslant r$ and X_κ is an *introduce node* (κ is well defined as X_r is an introduce node and $h < r = |X|$). We set $\{u^h\} = X_k - X_h$ and $N_h = N_{G_\kappa}(u^h)$. Now we define a mapping $\phi_h : \mathcal{P}_h \to \{0, 1, \dots, |l_h|\}$ such that $\phi_h(j, m) = \gamma$ implies that

$$\begin{aligned} &\mathsf{Ins}\bigl(G_h, u^h, X_h, N_h, \lambda_h, \mathbf{A}_h, j, m\bigr) \\ &\quad = \mathsf{Com}\bigl(\mathsf{Ins}\bigl(G_h, u^h, V(G_h), N_h, l_h, \mathbf{Q}_{G_h, l_h}, \gamma, 1\bigr), X_h\bigr). \end{aligned} \tag{38}$$

Our definition is recursive. We assume that for some h, $1 \leqslant h \leqslant r-1$, l_h and ϕ_h are known. We will show that l_{h+1}, ϕ_{h+1} can be defined recursively and computed in $O(1)$ time.

We first examine the case where $(\lambda_{h+1}, \mathbf{A}_{h+1})$ was computed after a call of Introduce-Node. This means that $\kappa = h + 1$ and that $\{u^h\} = X_{h+1} - X_h$ and $N_h = N_{G_{h+1}}(u^h)$. Clearly, $(\lambda_{h+1}, \mathbf{A}_{h+1}) = \mathsf{Ins}(G_h, u^h, X_h, N_h, \lambda_h, \mathbf{A}_h, j, m)$ for some choice of j and m where $0 \leqslant j \leqslant |\lambda_h|$ and $1 \leqslant m \leqslant |\mathbf{A}(j)|$. From (38) and the proof of Lemma 20, we derive the following:

If $\gamma = \phi_h(j, m)$, then by setting $l_{h+1} = l_h[1, \gamma] \cdot [u^h] \cdot l_h[\gamma + 1, |l_h|]$, we get $C_{X_{h+1}}(G_{h+1}, l_{h+1}) = (\lambda_{h+1}, \mathbf{A}_{h+1})$.

If we now take in mind the rearrangement of the indices occuring during step **1** of the procedure Ins as it is applied on $(\lambda_h, \mathbf{A}_h)$ and $(l_h, \mathbf{Q}_{G_h, l_h})$ to (38) we have that

$$\begin{aligned} \mathcal{P}_{h+1} = {} & \bigl\{(0,1), \dots, \bigl(0, \bigl|A_0^h\bigr|\bigr)\bigr\} \cup \cdots \cup \bigl\{(j-1, 1), \dots, \bigl(j-1, \bigl|A_{j-1}^h\bigr|\bigr)\bigr\} \\ & \cup \bigl\{(j,1), \dots, (j,m)\bigr\} \cup \bigl\{(j+1, 1), \dots, \bigl(j+1, \bigl|A_j^h\bigr| - m + 1\bigr)\bigr\} \\ & \cup \bigl\{(j+2, 1), \dots, \bigl(j+2, \bigl|A_{j+1}^h\bigr|\bigr)\bigr\} \cup \cdots \\ & \cup \bigl\{\bigl(|\mathbf{A}_h| + 1, 1\bigr), \dots, \bigl(|\mathbf{A}_h| + 1, \bigl|A_{|\mathbf{A}_h|}^h\bigr|\bigr)\bigr\} \end{aligned}$$

and, we can define ϕ_{h+1} as

$$\phi_{h+1}(\nu,\xi)=\begin{cases}\phi_h(\nu,\xi) & \text{if } \nu<j+1,\\ \gamma+1 & \text{if } \nu=j+1 \text{ and } \xi=1,\\ \phi_h(j,m+\xi-1)+1 & \text{if } \nu=j+1 \text{ and } \xi>1,\\ \phi_h(\nu-1,\xi)+1 & \text{if } \nu>j+1\end{cases}$$

and, therefore the required condition holds.

Suppose now that $(\lambda_{h+1},\mathbf{A}_{h+1})$ was computed after a call of Forget-Node. Assume

$$\lambda_h=[v_1,\ldots,v_j,\ldots,v_{|\lambda_h|}]$$

where v_j is the forgotten vertex. Clearly, the new vertex ordering l_{h+1} is the same as l_h. Taking now in mind the outputs of $\mathsf{Del}(\lambda_h,\mathbf{A}_h,v_j)$ and $\mathsf{Del}(l_h,\mathbf{Q}_{G_h,l_h},v_j)$, the new index set is

$$\begin{aligned}\mathcal{P}_{h+1}=&\{(0,1),\ldots,(0,|A_0^h|)\}\cup\cdots\cup\{(j-2,1),\ldots,(j-2,|A_{j-2}^h|)\}\\ &\cup\{(j-1,1),\ldots,(j-1,|\tau(A_h(j-1)\cdot A_h(j))|)\}\\ &\cup\{(j,1),\ldots,(j,|A_{j+1}^h|)\}\cup\cdots\cup\{(|\mathbf{A}_h|-1,1),\ldots,(|\mathbf{A}_h|-1,|A_{|\mathbf{A}_h|}^h|)\}\end{aligned}$$

and the function ϕ_{h+1} is obtained by setting

$$\phi_{h+1}(\nu,\xi)=\begin{cases}\phi_h(\nu,\xi) & \text{if } \nu<j-1,\\ \phi_h(j-1+\sigma,\psi) & \text{if } \nu=j-1,\\ \phi_h(\nu+1,\xi)-1 & \text{if } \nu>j-1\end{cases}$$

where $(\sigma,\psi)=\delta((A_h(j-1),A_h(j)),\xi))$. Clearly, the function ϕ_{h+1} verifies the required conditions.

If at each time a new characteristic is computed, we set up a pointer to the characteristic it was constructed from, we obviously have a suitable structure for constructing also a witness path in linear time. We will also maintain a data structure associating the position (determined by the pair (j,m)) of each element $a_m^{h,j}$ of a typical sequence A_j^h of $\mathbf{A}_h$ with the value $\gamma=\phi_h(a_m^{h,j})$, $0\leqslant\gamma|l_h|$. Furthermore, from the definitions, l_{h+1} and ϕ_{h+1} can be computed in $O(1)$ time from l_h and ϕ_h. Therefore, as $l_1=[x_{\text{start}}]$ and $\phi_1(0,1)=0$ and $\phi_1(1,1)=1$, we are able to construct in time linear in $|X|$, a vertex ordering $l=l_r$ such that $C_{X_r}(G_r,l_r)\in FS(r)$.

Notice that, because of Lemma 15, the algorithms Introduce-Node and Forget-Node run in $O(1)$ time when k and w are fixed. We summarize the results of the previous subsections in the following.

Theorem 24. *For all $k,w\geqslant 1$, the* Layout-Cutwidth *algorithm, with input a graph G and a path decomposition $X=[X_1,\ldots,X_r]$ of G with width at most w, computes whether the cutwidth of G is at most k and, if so, constructs a vertex ordering of G with cutwidth at most k, in $O(n+r)$ time.*

According to the results in [7] and [3], one can construct, for any k, a linear time algorithm that decides whether the pathwidth of a graph is at most k and, in case of a positive answer, outputs the corresponding path decomposition. Combining this fact with Lemma 2 and Theorem 24, we derive the following:

Theorem 25. *For all $k \geqslant 0$, it is possible to construct an algorithm, that given a graph G, computes whether the cutwidth of G is at most k and, if so, constructs a vertex ordering of G with minimum cutwidth in $O(n)$ time.*

5. Final remarks

It is known (e.g., see [9]) that graphs with maximum degree bounded by Δ and pathwidth bounded by w have cutwidth bounded by $w\Delta$. Therefore, our result implies a linear time algorithm for computing the cutwidth for graphs where both maximum degree and pathwidth are bounded by a constant.

Moreover, one can easily observe that, in the more general case where pathwidth(G) is at most w and the maximum degree of G is at most $\beta \cdot \log n$, for some $\beta \geqslant 0$, Lemma 15 bounds the number of different characteristics by $O((w+1)!(\frac{8}{3})^{w+1}n^{2\beta w(w+1)})$. The complexity of the algorithm Introduce-Node is $O(w^3 w!(\frac{8}{3})^{w+1}n^{2\beta w(w+1)+1})$ as step **2** requires $O((w+1)!(\frac{8}{3})^{w+1}n^{2\beta w(w+1)})$ repetitions, step **3** requires $O(w)$ repetitions, step **4** requires $O(n)$ repetitions, and the call of the procedure Ins in step **5** requires $O(w)$ steps. Similarly, the complexity of the algorithm Forget-Node is $O(w^2 w!(\frac{8}{3})^{w+1}n^{2\beta w(w+1)})$. Therefore, we can conclude that for any constants w and β there exists a polynomial time algorithm (i.e., $O(w^3 w!(\frac{8}{3})^{w+1}n^{2\beta w(w+1)+2}\log n))$ that outputs a minimum cutwidth linear layout of any graph G with maximum degree $\beta \cdot \log n$, $\beta \geqslant 0$ and pathwidth $\leqslant w$, $w \geqslant 0$.

In the second part of this work [21] we provide a polynomial time algorithm that outputs a minimum cutwidth linear layout of any graph G where the maximum degree and the treewidth are fixed constants. The algorithm uses as subroutines the algorithms Introduce-Node and Forget-Node and its analysis is based on the definitions and the results of this paper. For an analogous result concerning the computation of pathwidth for graphs with bounded degree, see Section 7 of [5].

Acknowledgment

We wish to thank one of the anonymous referees for his detailed comments and remarks that helped us improving the overall presentation of this work.

References

[1] K. Abrahamson, M. Fellows, Finite automata, bounded treewidth and well-quasiordering, in: Graph Structure Theory, Seattle, WA, 1991, in: Contemp. Math., vol. 147, Amer. Math. Soc., Providence, RI, 1993, pp. 539–563.

[2] H.L. Bodlaender, Improved self-reduction algorithms for graphs with bounded treewidth, Discrete Appl. Math. 54 (2–3) (1994) 101–115.

[3] H.L. Bodlaender, A linear-time algorithm for finding tree-decompositions of small treewidth, SIAM J. Comput. 25 (6) (1996) 1305–1317.

[4] H.L. Bodlaender, M.R. Fellows, D.M. Thilikos, Starting with nondeterminism: the systematic derivation of linear-time graph layout algorithms, in: Proc. 26th International Symposium on Mathematical Foundations

of Computer Science, MFCS, 2003, in: Lecture Notes in Comput. Sci., vol. 2747, Springer-Verlag, 2003, pp. 239–248.

[5] H.L. Bodlaender, T. Kloks, Efficient and constructive algorithms for the pathwidth and treewidth of graphs, J. Algorithms 21 (1996) 358–402.

[6] H.L. Bodlaender, D.M. Thilikos, Constructive linear time algorithms for branchwidth, in: Automata Languages and Programming, Bologna, 1997, in: Lecture Notes in Comput. Sci., vol. 1256, Springer-Verlag, Berlin, 1997, pp. 627–637.

[7] H.L. Bodlaender, D.M. Thilikos, Computing small search numbers in linear time, Technical Report UU-CS-1998-05, Department of Computer Science, Utrecht University, 1998.

[8] F.R.K. Chung, On the cutwidth and the topological bandwidth of a tree, SIAM J. Algebraic Discrete Methods 6 (2) (1985) 268–277.

[9] F.R.K. Chung, P.D. Seymour, Graphs with small bandwidth and cutwidth, Discrete Math. 75 (1–3) (1989) 113–119.

[10] M.J. Chung, F. Makedon, I.H. Sudborough, J. Turner, Polynomial time algorithms for the MIN CUT problem on degree restricted trees, SIAM J. Comput. 14 (1) (1985) 158–177.

[11] J. Díaz, J. Petit, M. Serna, A survey on graph layout problems, ACM Comput. Surveys 34 (3) (2002) 313–356.

[12] M.R. Fellows, M.A. Langston, On well-partial-order theory and its application to combinatorial problems of VLSI design, SIAM J. Discrete Math. 5 (1) (1992) 117–126.

[13] M.R. Fellows, M.A. Langston, On search, decision, and the efficiency of polynomial-time algorithms, J. Comput. System Sci. 49 (3) (1994) 769–779.

[14] M.R. Garey, D.S. Johnson, Computers and Intractability. A Guide to the Theory of NP-Completeness, Freeman, San Francisco, CA, 1979.

[15] E. Korach, N. Solel, Tree-width, path-width, and cutwidth, Discrete Appl. Math. 43 (1) (1993) 97–101.

[16] F.S. Makedon, C.H. Papadimitriou, I.H. Sudborough, Topological bandwidth, SIAM J. Algebraic Discrete Methods 6 (3) (1985) 418–444.

[17] B. Monien, I.H. Sudborough, Min cut is NP-complete for edge weighted trees, Theoret. Comput. Sci. 58 (1–3) (1988) 209–229.

[18] N. Robertson, P.D. Seymour, Graph minors. XXII. The Nash–Wiliams immersion conjecture. J. Combin. Theory Ser. B, in press.

[19] N. Robertson, P.D. Seymour, Graph minors. XIII. The disjoint paths problem, J. Combin. Theory Ser. B 63 (1) (1995) 65–110.

[20] D.M. Thilikos, H.L. Bodlaender, Constructive linear time algorithms for branchwidth, Technical Report UU-CS-2000-38, Department of Computer Science, Utrecht University, 2000.

[21] D.M. Thilikos, M. Serna, H.L. Bodlaender, Cutwidth II: Algorithms for partial w-trees of bounded degree, J. Algorithms 56 (1) (2005) 25–49.

[22] M. Yannakakis, A polynomial algorithm for the min-cut linear arrangement of trees, J. ACM 32 (4) (1985) 950–988.