A constructive linear time algorithm for small cutwidth*†

Dimitrios M. Thilikos, Maria J. Serna

Departament de Llenguatges i Sistemes Informàtics, Universitat Politècnica de Catalunya, Campus Nord – Mòdul C5, c/Jordi Girona Salgado 1-3, 08034 Barcelona, Spain

E-mail: {sedthilk, mjserna}@lsi.upc.es

and

Hans L. Bodlaender

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

E-mail: hansb@cs.uu.nl

The *cutwidth* of a graph G is defined to be the smallest integer k such that the vertices of G can be arranged in a linear layout $[v_1, \ldots, v_n]$ in such a way that for every $i = 1, \ldots, n-1$, there are at most k edges with the one endpoint in $\{v_1, \ldots, v_i\}$ and the other in $\{v_{i+1}, \ldots, v_n\}$. In this paper we show how to construct, 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.

Key Words: cutwidth, graph immersion, pathwidth

1. INTRODUCTION

Layouts provide the framework for the definition of several graph theoretic parameters with a wide range of applications. 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, cutwidth$(G) \leq k$, is an NP-complete problem known in the literature as the Mini-

* The work of all the authors was supported by the EU project ALCOM-FT (IST-99-14186). The work of the first author was partially supported by the Ministry of Education and Culture of Spain, Grant number MEC-DGES SB98 0K148809.

† 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 the proceedings of ISAAC 2000.

mum Cut Linear Arrangement (see [12]). Cutwidth has been extensively examined (see [7, 10, 11, 13, 15, 16, 19]). It is closely related with other graph theoretic parameters like pathwidth, bandwidth, modified bandwidth (see [14, 7, 15, 13, 8]), and it is approximable within a factor of $O(\log n \log\log n)$ in polynomial time (see [9]). Finally, while it remains NP-complete even for planar graphs with maximum degree 3 (see [16]), it is polynomially computable for trees (see [19]).

Our results concern the fixed parameter tractability of cutwidth. 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 [17]). 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 in 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 in G (see [18, 10]), 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 a way to construct it. However, it provides 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 [10, 11]) 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$\leq k$ was given by Makedon and Sudborough in [15] where a $O(n^{k-1})$ dynamic programming algorithm is described. This time complexity has been considerably improved by Fellows and Langston in [11] where, among others, they prove that for any fixed k, a $O(n^3)$ algorithm can be constructed checking whether a graph has cutwidth at most k. Furthermore, a technique introduced in [10] (see also [2]) further reduced the bound 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 the case of a positive answer, the corresponding layout. In this paper, we give an explicit description, for any $k \geq 1$, of a linear time algorithm that checks whether an input graph G has cutwidth$\leq k$ and, if this is the case, it further *outputs* a vertex layout of G of minimum cutwidth.

The key tool in our algorithm, as for other parameters like pathwidth and treewidth in [4], linear-width in [6], and branchwidth in [5], is the notion of *characteristics*. In a few words, 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 a path decomposition. Moreover, as we will see, the information encoded by a characteristic depends on the width of the path decomposition and, therefore, it is constant for graphs with bounded pathwidth.

Our algorithm starts with an adequate bounded width path decomposition of the input graph G. The path decomposition allows the definition of an appropriate sequence of subgraphs. The algorithm computes, in a bottom-up fashion, a set of characteristics that "represent" the vertex orderings that have cutwidth $\leq k$ for any of the subgraphs.

A consequence of our result is that there exists 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 [4, 3], agile search parameters [6], linear-width [6], and branch-width [5]. 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 [10]).

2. DEFINITIONS AND PRELIMINARY RESULTS

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 $V(G)$ ($E(G)$). A linear (one-dimensional) layout of the vertices of G is a bijection, mapping $V(G)$ to the integers in $\{1, \ldots, n\}$. We denote such a layout by the sequence $[v_1, \ldots, v_n]$.

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 $\mathcal{O}$. 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.

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. 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 $\omega^1 \oplus \omega^2 = [\omega^1_1, \ldots, \omega^1_{r_1}, \omega^2_1, \ldots, \omega^2_{r_2}]$. Unless mentioned otherwise, we will always consider that the first element of a sequence ω is indexed by 1, i.e. $\omega = \omega[1, |\omega|]$.

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 $G \overset{u}{+} N$ is obtained by adding 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 $V(E) = \cup_{e \in E} e$. 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 sequences of all the vertices of G without repetitions, then we will call it *vertex ordering 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 $\mathrm{rank}(u)$.

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. $\cup_{1 \leq i \leq i} X_i = V(G)$.
2. $\forall_{e \in E(G)}\ \exists_{1 \leq i \leq r}\ e \subseteq X_i$.
3. $\forall_{v \in V(G)}\ \exists_{1 \leq i \leq j \leq r}\ \forall_{1 \leq h \leq r}\ v \in X_h \Leftrightarrow i \leq h \leq j$.

We call the sets $X_1, \ldots, X_r$, *nodes* of the path decomposition X. The *width* of X is equal to $\max_{1 \leq i \leq r}\{|X_i| - 1\}$ 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 $\forall_{2 \leq i \leq |X|}\ |(X_i - X_{i-1}) \cup (X_{i-1} - X_i)| = 1$. The following lemma follows directly from the definitions.

LEMMA 2.1. *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 at most $2|V(G)|$ nodes in $O(|V(G)|)$ time.*

Let X_i be a node of a nice path decomposition X. We say that X_i is an *introduce* (*forget*) node if $|X_i - X_{i-1}| = 1$ ($|X_{i-1} - X_i| = 1$). It is easy to observe that any node $X_i, i \geq 2$ of a nice path decomposition is either an *introduce* or a *forget* node. We call the first node X_1 of X, *start* node. Notice that if 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 $i = 1, \ldots, r$, we define $V_i = \cup_{1 \leq j \leq i} X_j$ 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 $\theta_{l,G}(i) = E_G(l[1,i]) \cap E_G(l[i+1,n])$ (i.e. $\theta_{l,G}(i)$ is the set of edges of G that have one endpoint in $l[1,i]$ and one in $l[i+1,n]$). The cutwidth of an ordering l of $V(G)$ is $\max_{1 \leq i \leq n-1}\{|\theta_{l,G}(i)|\}$. The cutwidth of a graph is the minimum cutwidth over all the orderings of $V(G)$. It is easy to see the following (see also [15]).

LEMMA 2.2. *For any graph G, cutwidth(G) $\geq$ pathwidth(G).*

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

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

We also assume that the indices of the elements of $\mathbf{Q}_{G,l}$ start from 0 and finish on n, i.e. $\mathbf{Q}_{G,l} = \mathbf{Q}_{G,l}[0,n]$. 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 $\mathbf{Q}_{G,l}$ for reasons of consistency with terminology that will be introduced later (for an example of $\mathbf{Q}_{G,l}$, see Figure 1).

2.3. Sequences of integers

We denote as $\mathcal{S}$ the set of all the sequences of non-negative integers. For any sequence $A = [a_1, \dots, a_{|A|}] \in \mathcal{S}$ and any integer $t \geq 0$ we set $A + t = [a_1 + t, \dots, a_{|A|} + t]$. If $A = [a_1, \dots, a_{|A|}] \in \mathcal{S}$, we define $\tau(A)$ as the subsequence obtained after iterating the following operations, until none is possible any more.

(i) If for some i, $1 \leq i \leq |A| - 1$ $a_i = a_{i+1}$, then set $A \leftarrow A[1, i] \oplus A[i+2, |A|]$.

(ii) 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 set $A \leftarrow A(1, i) \oplus A(j, |A|)$.

For example, $\tau([\mathbf{5},5,6,7,7,\mathbf{7},4,4,\mathbf{3},5,4,6,\mathbf{8},\mathbf{2},\mathbf{9},3,4,6,7,\mathbf{2},\mathbf{7},5,4,4,6,\mathbf{4}]) = [5,7,3,8,2,9,2,7,4]$. We call a sequence A *typical* if $A \in \mathcal{S}$ and $\tau(A) = A$.

The following result has been proved in [4] (Lemma 3.5).

LEMMA 2.3. *The number of different typical sequences consisting of integers in* $\{0, 1, \dots, n\}$ *is at most* $\frac{8}{3}2^{2n}$.

Notice that $B = \tau(A)$ is a subsequence $[a_{i_1}, \dots, a_{i_{|B|}}]$ of $A = [a_1, \dots, a_{|A|}]$ such that for any j, $1 \leq j \leq |B| - 1$ either $a_{i_j} \leq a_{i_j+1} \leq \cdots \leq a_{i_{j+1}-1} \leq a_{i_{j+1}}$ or $a_{i_j} \geq a_{i_j+1} \leq \cdots \geq a_{i_{j+1}-1} \geq a_{i_{j+1}}$. We can now define a function $\beta_A : \{1, \dots, |\tau(A)|\} \rightarrow \{1, \dots, |A|\}$ where $\beta_A(j) = i_j$ is one of the possible original positions in A of the j-th 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

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

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

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

As an example we have that if $A = [1, 3, 2]$ and $B = [8, 5, 9]$, we have that $\tau(A \oplus B) = [1, 9]$, $\delta(A, B, 1) = (0, 1)$, and $\delta(A, B, 2) = (1, 3)$

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

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

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 $A \leq B$ if $r_1 = r_2$ and $\forall_{1 \leq i \leq r_1}\ a_i \leq b_i$. In general, we say that $A \prec B$ if there exist extensions $\tilde{A} \in \mathcal{E}(A)$, and $\tilde{B} \in \mathcal{E}(B)$ such that $\tilde{A} \leq \tilde{B}$. For example if $A = [1, 7, 2, 6, 4]$ and $B = [5, 7, 3, 8]$ then $A \prec B$ because $\tilde{B} =$ [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 $\tilde{A} \leq \tilde{B}$.

The following three lemmata are easy consequences of the definitions.

LEMMA 2.4. *Given $R \in \mathcal{S}$, if we set $A = \tau(R)$ then, for any $m, 1 \leq m \leq |A|$, there exists a i, $1 \leq i \leq |R|$ such that $A[1, m] = \tau(R[1, i])$ and $A[m, |R|] = \tau(R[i, |A|])$.*

LEMMA 2.5. *Let A_1, A_2 be two typical sequences where $A_2 \prec A_1$. Then, for any $m_1, 1 \leq m_1 \leq |A_1|$, there exists a $m_2, 1 \leq m_2 \leq |A_2|$ 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 2.6. *Let $A_i, B_i, i = 1, 2$ be four typical sequences where $A_i \prec B_i, i = 1, 2$. Then $\tau(A_1 \oplus B_1) \prec \tau(A_2 \oplus B_2)$.*

LEMMA 2.7. *Given $R \in \mathcal{S}$, if we set $A = \tau(R)$ then, for any $r, 1 \leq r \leq |R|$, there exists an integer $i, 1 \leq i \leq |A|$ such that $A[1, i] \prec \tau(R[1, r])$ and $A[i, |R|] \prec \tau(R[r, |A|])$.*

Proof. In the case where there exists an integer i, $1 \leq i \leq |A|$ 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, there exist an integer j, $1 \leq j < |A|$ and two integers k ,l, where $1 \leq k < r < l \leq |R|$ and such that $A[1, j] = \tau(R[1, k])$ and $A[j + 1, |A|] = \tau(R[l, |R|])$. 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 $A[j+1, j+1] \prec \tau(R[r, l])$. Using the fact that $A[1, j] = \tau(R[1, k])$ and Lemma 2.6, we get

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

Now using the fact that $A[j+1, |A|] = \tau(R[l, |R|])$ and Lemma 2.6, we get

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

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 of Lemma 2.7 we consider the sequences

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

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

$$\begin{aligned} [2, 8] &= \tau([1, 6, 7, 8]), \\ [8, 2] &\prec \tau([8, 5, 4, 3]), \\ [2] &\prec \tau([3, 5, 2]), \\ [2, 6, 4] &= \tau[2, 4, 6, 4, 4], \\ [2, 8, 2] &\prec \tau([1, 6, 7, 8, 5, 4, 3]) \text{ and} \\ [2, 6, 4] &\prec \tau([3, 5, 2, 4, 6, 4, 4]). \end{aligned}$$

Suppose now that $\mathbf{A} = [A_1, \ldots, A_r]$ and $\mathbf{B} = [B_1, \ldots, B_r]$ are two sequences of typical sequences. We say that $\mathbf{A} \prec \mathbf{B}$ if $\forall_{1 \le i \le r}\ A_i \prec B_i$. For any integer t we set $\mathbf{A} + t = [A_1 + t, \ldots, A_{|\mathbf{A}|} + t]$ and $\max(\mathbf{A}) = \max_{1 \le i \le |\mathbf{A}|}\{\max A_i\}$. Finally, for any sequence of typical sequences $\mathbf{A}$ we set $\tau(\mathbf{A}) = \tau(\mathbf{A}(1) \oplus \cdots \oplus \mathbf{A}(|\mathbf{A}|))$. As an example,

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

2.4. Characteristic pairs

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

The following procedure defines the *compression* of a characteristic pair relative to a subset of $\mathcal{O}$.

Procedure $\mathsf{Com}(l, \mathbf{R}, S)$.

Input: A characteristic pair $(l, \mathbf{R})$ and a set S.

Output: A characteristic pair $(\lambda, \mathbf{A})$.

We assume the notations $l = [v_1, \ldots, v_{|l|}]$ and $\lambda = [v_{i_1}, v_{i_2}, \ldots, v_{i_\rho}]$.

1: $\lambda \leftarrow l[S]$.

2: $\mathbf{A} \leftarrow [\tau(\mathbf{R}[0, i_1 - 1]), \tau(\mathbf{R}[i_1, i_2 - 1]), \ldots, \tau(\mathbf{R}[i_{\rho-1}, i_\rho - 1]), \tau(\mathbf{R}[i_\rho, |l|])]$.

3: Output $(\lambda, \mathbf{A})$.

4: End.

The pair $([a, b, c, d, e], [[0, 3], [4], [3, 7, 2], [3], [8, 1, 3], [3, 8, 4, 6]])$ is an example of a characteristic pair. We also use a different notation for characteristic pairs, 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 compression of this characteristic pair to the set $S = \{a, c\}$ is the characteristic pair

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

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 $X = [X_1, \ldots, X_r]$ of width at most w, decides whether G has cutwidth at most k.

Let us start by defining a *characteristic* of 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 that the $V(G)$-characteristic of l is equal to $(l, \mathbf{Q}_{G,l})$, i.e. $C_{V(G)}(G, l) = (l, \mathbf{Q}_{G,l})$ (clearly, $\mathsf{Com}(l, \mathbf{Q}_{G,l}, V(G)) = (l, \mathbf{Q}_{G,l})$).

As an example we mention that, for the graph and ordering l given in Figure 1, the characteristic pairs $C_N(G, l)$ and $C_{V(G)-N}(G, l)$ are:

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

Given the S-characteristics $(\lambda^i, \mathbf{A}^i)$, $i = 1, 2$, of two different vertex orderings 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 3.1. *Given a graph G, a vertex ordering l of G, and a vertex subset S. Assume that $C_S(l, G) = (\lambda, \boldsymbol{A})$. For any two vertices a, b of S, with ranks i, j (i', j') in l and λ respectively, we have*

$$(\lambda, \mathbf{A}[1, i-1] \oplus (\mathbf{A}[i, j-1] + 1) \oplus \mathbf{A}[j, |\mathbf{A}|]) \\ = \mathsf{Com}(l, \mathbf{Q}_{G,l}[1, i'-1] \oplus (\mathbf{Q}_{G,l}[i', j'-1] + 1) \oplus \mathbf{Q}_{G,l}[j', |\mathbf{Q}_{G,l}|], S).$$

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

Using now Lemma 2.3 and working in a similar way as in the proof of Lemma 3.1 in [4] we can prove that for any i, the number of X_i characteristics depends only on k and w.

LEMMA 3.2. *Let G be a graph and let $X = [X_1, \ldots, X_r]$ be a nice path decomposition of G with width at most w. Let X_i, $1 \leq i \leq r$ be some node in X. The number of different X_i-characteristics of all possible vertex orderings of $G_i = G[\cup_{j=1}^{i} X_j]$ with cutwidth at most k, is bounded by $w!\,(\frac{8}{3}\,2^{2k})^{w+1}$.*

Proof. Let $(\lambda, \mathbf{A})$ be a X_i-characteristic of some vertex ordering of G_i. Clearly, $V(\lambda) \subseteq X_i$ and, as $|X_i| \leq w$, there are at most $w!$ ways to choose λ. For each one of them, there are $\leq w+1$ typical sequences in $\mathbf{A}$ to be chosen. From lemma 2.3, there are at most $\frac{8}{3}2^{2k}$ different ways to choose each of these sequences and the lemma follows. ∎

Assume from now on that we have a graph G and that $X = [X_1, \ldots, X_r]$ is a nice path decomposition of G, with width at most w. A set $FS(i)$ of X_i-characteristics of vertex orderings of the graph $G_i = G[\cup_{j=1}^{i} X_i]$ with cutwidth at most k is called a *full set of characteristics* for G_i if 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 3.3. *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 3.3 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. In what follows, we will show 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 ($i \geq 2$).

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 $\{x_{\text{start}}\} = X_1$ 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 following procedure is the basis to compute a characteristic after the insertion of the introduced vertex and the additional edges that appear in G_i.

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

Input: A graph G, a vertex $u \notin V(G)$, two sets S, N where $N \subseteq S \subseteq V(G)$,

a S-characteristic $(\lambda, \mathbf{A})$ of some vertex ordering l of G,

an integer $j, 0 \leq j \leq |\lambda|$, and an integer $m, 1 \leq m \leq |\mathbf{A}(j)|$.

Output: An $(S \cup \{u\})$-characteristic $(\lambda', \mathbf{A}')$ of some vertex ordering

$l' = l[1, \ldots, \gamma] \oplus [u] \oplus l[\gamma + 1, \ldots, |l|]$ of $G' = G \overset{u}{+} N$ where $0 \leq \gamma \leq |l|$.

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

1: (Insertion of u)

Set $\lambda' = \lambda[1, j] \oplus [u] \oplus \lambda[j+1, \rho]$

and $\mathbf{A}' = \mathbf{A}[0, j-1] \oplus [\mathbf{A}(j)[1, m]] \oplus [\mathbf{A}(j)[m, |\mathbf{A}(j)|]] \oplus \mathbf{A}[j+1, \rho]$.

2: (Insertion of the edges from u)

for $h = 1$ **to** σ **do**

(i) If $j_h \leq j$ then set $\mathbf{A}' \leftarrow \mathbf{A}'[0, j_h - 1] \oplus (\mathbf{A}'[j_h, j] + 1) \oplus \mathbf{A}'[j+1, \rho+1]$.

(ii) If $j_h \geq j+1$ then set $\mathbf{A}' \leftarrow \mathbf{A}'[0, j] \oplus (\mathbf{A}'[j+1, j_h] + 1) \oplus \mathbf{A}'[j_h + 1, \rho + 1]$.

3: Output $(\lambda', \mathbf{A}')$.

4: End.

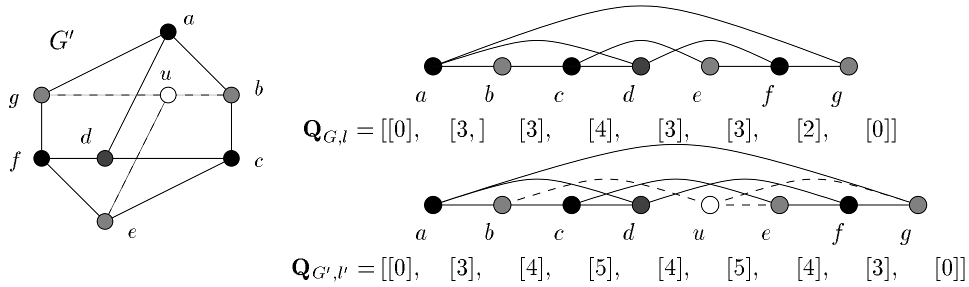

FIG. 1. An example of Lemma 3.4 where $G = G'[V(G') - \{u\}], l = [a, b, c, d, e, f, g], N = \{b, e, g\}$, and $\gamma = 4$.

The following lemma is a direct consequence of the definitions of $\mathbf{Q}_{G,l}$, $\mathbf{Q}_{G',l'}$ and the insertion procedure. Notice 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 (for an example, see Figure 1).

LEMMA 3.4. *Let G be a graph, l be a vertex ordering of G and γ be an integer where $0 \leq \gamma \leq |l|$. If $G' = G \overset{u}{+} N$ where $N \subseteq V(G)$ 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 $l' = l(1, \gamma) \oplus [u] \oplus l(\gamma+1, |l|)$, i.e. $\mathsf{Ins}(G, u, V(G), N, l, \mathbf{Q}_{G,l}, \gamma, 1) = (l', \mathbf{Q}_{G',l'})$.*

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 3.5. *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 $G' = G \overset{u}{+} N$ where $u \notin V(G)$. Then the following hold.*

(i)For any $j, 0 \leq j \leq |\lambda|, m, 1 \leq m \leq |\mathbf{A}(j)|$ there exists an integer $0 \leq \gamma \leq |l|$ such that $\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}, j, m) = \mathsf{Com}(\mathsf{Ins}(G, u, V(G), N, l, \mathbf{Q}_{G,l}, \gamma, 1), S \cup \{u\})$.

(ii)For any $\gamma, 0 \leq \gamma \leq |l|$, there exist two integers $j, 0 \leq j \leq |\lambda|, m, 1 \leq m \leq |\mathbf{A}(j)|$ such that $\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}, j, m) \prec \mathsf{Com}(\mathsf{Ins}(G, u, V(G), N, l, \mathbf{Q}_{G,l}, \gamma, 1), S \cup \{u\})$.

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 Procedure Com we have that $\forall_{h, 0 \leq h \leq \rho} \mathbf{A}(h) =$

$\tau(\mathrm{Q}_{G,l}[i_h, i_{h+1}-1])$ (for convenience, we set $i_0 = 0$). It is now easy to verify that

$$\begin{aligned}
\mathbf{A}[0, j-1] &= [\tau(\mathbf{Q}_{G,l}[0, i_1 - 1]), \tau(\mathbf{Q}_{G,l}[i_1, i_2 - 1]), \ldots, \tau(\mathbf{Q}_{G,l}[i_{j-1}, i_j - 1])] && (1)\\
\mathbf{A}(j) &= \tau(\mathbf{Q}_{G,l}[i_j, i_{j+1} - 1]) && (2)\\
\mathbf{A}[j+1, \rho] &= [\tau(\mathbf{Q}_{G,l}[i_{j+1}, i_{j+2} - 1]), \ldots, \tau(\mathbf{Q}_{G,l}[i_{\rho-1}, i_\rho - 1]), \tau(\mathbf{Q}_{G,l}[i_\rho, |l|])] && (3)
\end{aligned}$$

Applying now Lemma 2.4 on (2), we have that there exists $\gamma, i_j \leq \gamma < i_{j+1}$ such that

$$\begin{aligned}
\mathbf{A}(j)[1, m] &= \tau(\mathbf{Q}[i_j, \gamma]) && (4)\\
\mathbf{A}(j)[m, |\mathbf{A}(j)|] &= \tau(\mathbf{Q}[\gamma, i_{j+1} - 1]) && (5)
\end{aligned}$$

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

$$\begin{aligned}
l[1, \gamma] \cap S &= \lambda[1, j] && (6)\\
l[\gamma + 1, |l|] \cap S &= \lambda[j + 1, \rho] && (7)
\end{aligned}$$

Let $(\lambda', \mathbf{A}')$ and $(l', \mathbf{R})$ be the characteristic pairs constructed by step **1** of 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 $\lambda' = \lambda[1, j] \oplus [u] \oplus \lambda[j+1, |\rho|]$ and $l' = l[1, \gamma] \oplus [u] \oplus l[\gamma+1, |l|]$. 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

$$\begin{aligned}
\mathbf{A}' &= \mathbf{A}[0, j-1] \oplus [\mathbf{A}(j)[1, m]] \oplus [\mathbf{A}(j)[m, |\mathbf{A}(j)|]] \oplus \mathbf{A}[j+1, \rho] && (8)\\
\mathbf{R} &= \mathbf{Q}_{G,l}[0, i_j - 1] \oplus \mathbf{Q}_{G,l}[i_j, \gamma] \oplus \mathbf{Q}_{G,l}[\gamma, i_{j+1} - 1] \oplus \mathbf{Q}_{G,l}[i_{j+1}, |l|] && (9)
\end{aligned}$$

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

$$\begin{aligned}
&[\tau(\mathbf{Q}_{G,l}[0, i_1 - 1]), \tau(\mathbf{Q}_{G,l}[i_1, i_2 - 1]), \ldots, \tau(\mathbf{Q}_{G,l}[i_{j-1}, i_j - 1]), \tau(\mathbf{Q}_{G,l}[i_j, \gamma]),\\
&\tau(\mathbf{Q}_{G,l}[\gamma, i_{j+1} - 1]), \tau(\mathbf{Q}_{G,l}[i_{j+1}, i_{j+2} - 1]), \ldots, \tau(\mathbf{Q}_{G,l}[i_{\rho-1}, i_\rho - 1]), \tau(\mathbf{Q}_{G,l}[i_\rho, |l|])] \quad (10)
\end{aligned}$$

Using now (1), (4), (5), and (3) 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 on $(\lambda', \mathbf{A}')$ and $(l', \mathbf{R})$ during the loops of step **2** of the computation 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.

Notice that during the execution of step **2**, no vertex is introduced, only new edges are taken into account, therefore the respective vertex orderings do not change. We will use the notation $(\lambda', \mathbf{A}^{(h)})$ and $(l', \mathbf{R}^{(h)})$ for the results of the h-th execution of the loop in step **2** of $\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}, j, m)$ and $\mathsf{Ins}(G, u, V(G), N, l, \mathbf{Q}_{G,l}, \gamma, 1), S \cup \{u\})$ respectively. And for convenience we denote $(\lambda', \mathbf{A}')$ as $(\lambda', \mathbf{A}^{(0)})$, and $(l', \mathbf{R})$ as $(l', \mathbf{R}^{(0)})$. Our proof is 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

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

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 $j_\xi \leq j$ we have that

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

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

As j_ξ and $\gamma + 1$ (ξ and $j + 1$) are the ranks of u_{j_ξ} and u in l' (λ'), Lemma 3.1 implies that $(\lambda^{(\xi)}, \mathbf{A}^{(\xi)}) = \mathsf{Com}(l^{(\xi-1)}, \mathbf{R}^{(\xi)}, S \cup \{u\})$.

In the case $j_\xi \geq j + 1$ we have that

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

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

As $\gamma + 1$ and $j_\xi + 1$ ($j + 1$ and $\xi + 1$) are the ranks of u and u_{j_ξ} in l' (λ'), Lemma 3.1 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 flow of the proof for (ii) is 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 Procedure Com we have that there exists a $j, 0 \leq j \leq |\rho|$ such that $i_j \leq \gamma < i_{j+1}$ (for convenience, we set $i_0 = 0$ and $i_{\rho+1} = |l| + 1$). Notice now that for this choice of j, (1)-(3) hold as well and (1) and (3) can be rewritten in the following weaker form.

$$\mathbf{A}[0, j-1] \prec [\tau(\mathbf{Q}_{G,l}[0, i_1 - 1]), \tau(\mathbf{Q}_{G,l}[i_1, i_2 - 1]), \ldots, \tau(\mathbf{Q}_{G,l}[i_{j-1}, i_j - 1])] \quad (15)$$

$$\mathbf{A}[j+1, \rho] \prec [\tau(\mathbf{Q}_{G,l}[i_{j+1}, i_{j+2} - 1]), \ldots, \mathbf{Q}_{G,l}[i_{\rho-1}, i_\rho - 1]), \tau(\mathbf{Q}_{G,l}[i_\rho, |l|])] \quad (16)$$

From Lemma 2.7 there exists an integer $m, 1 \leq m \leq \mathbf{A}(j)$ where

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

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

Notice that (15), (16), (17), and (18) are the same as (1), (3), (4), and (5) with the difference that "=" has been replaced by "$\prec$".

It is now easy to check that repeating the steps of the proof of case (i) it follows that

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

As an example for Lemma 3.5 we consider the graphs G and G' and their vertex orderings l and l' respectively as they are depicted in Figure 1. If we set $S = N \cup \{d\} = \{b, d, e, g\}$ we have that $C_S(G, l) = (\lambda, \mathbf{A})$ is the characteristic pair

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

For $\gamma = 4$ we have that, for $j = 2$ and $m = 1$,

$$\begin{aligned}\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}, 2, 1) &= ([u, b, d, e, g], [[0,3], [3,5], [4], [5], [4,3], [0]]) \\ &= \mathsf{Com}(\mathsf{Ins}(G, u, V(G), N, l, \mathbf{Q}_{G,l}, 4, 1), S \cup \{u\}).\end{aligned}$$

Moreover, for $j = 0$, and $m = 2$ we have that, for $\gamma = 1$ (see Figure 2), $\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}, 0, 2) = \mathsf{Com}(\mathsf{Ins}(G, u, V(G), N, l, \mathbf{Q}_{G,l}, 1, 1), S \cup \{u\})$ is the pair

$$[0,3]\ \boldsymbol{u}\ [6]\ \boldsymbol{b}\ [5,6]\ \boldsymbol{d}\ [5]\ \boldsymbol{e}\ [4,3]\ \boldsymbol{g}\ [0]$$

(in this example, the stronger version holds where "$\prec$" can be replaced by "=").

Let us prove now some monotonicity properties of the insertion procedure.

Lemma 3.6. *Let $(\lambda, \mathbf{A}_i), i = 1, 2$ be two characteristic pairs of a graph G where $(\lambda, \mathbf{A}_2) \prec (\lambda, \mathbf{A}_1)$. Let also N, S be subsets of $V(G)$, and $G' = G \overset{u}{+} N$ where $u \notin V(G)$.*

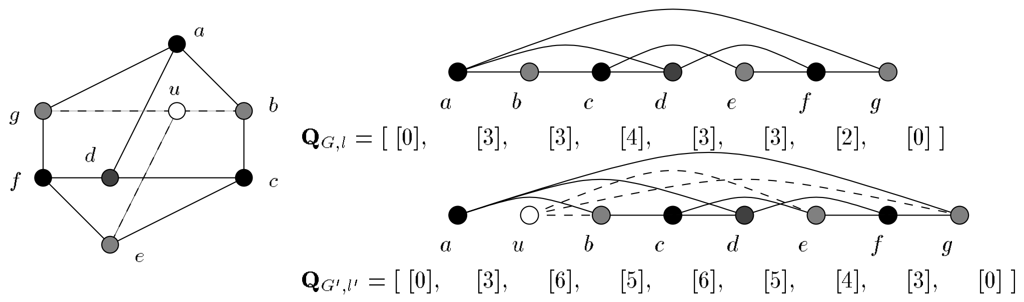

FIG. 2. An example of Lemma 3.5 where $G = G'[V(G') - \{u\}], l = [a, b, c, d, e, f, g], N = \{b, e, g\}$, and $\gamma = 1$.

Then for any j, $0 \leq j \leq |\mathbf{A}_1|$ and any m_1, $1 \leq m_1 \leq |\mathbf{A}_1(j)|$, there exists a m_2, $1 \leq m_2 \leq |\mathbf{A}_2(j)|$ 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. Since, $\mathbf{A}_2(j) \prec \mathbf{A}_1(j)$, Lemma 2.5 implies that there exists an m_2, $1 \leq m_2 \leq |\mathbf{A}_2(j)|$ such that

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

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

$$\mathbf{A}'_i = \mathbf{A}_i[0, j-1] \oplus [\mathbf{A}_i(j)[1, m_i]] \oplus [\mathbf{A}_i(j)[m_1, |\mathbf{A}_i(j)|]] \oplus \mathbf{A}_i[j+1, \rho], i = 1, 2,$$

then $\mathbf{A}'_2 \prec \mathbf{A}'_1$.

Clearly, $\mathbf{A}'_1$ 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 $\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}_i, j, m_i), i = 1, 2$, is the same ordering λ' for both. It now remains to prove that $\mathbf{A}'_2 \prec \mathbf{A}'_1$ holds after step **2** as well.

In what remains we will denote $\mathbf{A}'_i$ as $\mathbf{A}_i^{(0)}, i = 1, 2$ and we will use the notation $\mathbf{A}_i^{(h)}$ for the results of the h-th execution of the loop in step **2** during the computation of $\mathsf{Ins}(G, u, S, N, \lambda, \mathbf{A}_i, j, m_i)$, $i = 1, 2$. Suppose that $\mathbf{A}_2^{(h)} \prec \mathbf{A}_2^{(h)}$ 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 the case where $j_\xi \leq j$ (the case where $j_\xi \geq j+1$ is analogous). As $\mathbf{A}_2^{(\xi-1)} \prec \mathbf{A}_1^{(\xi-1)}$ we have that

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

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

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

Clearly, (20) is equivalent to the following.

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

Taking into account that

$$\mathbf{A}_i^\xi = \mathbf{A}_i^{(\xi-1)}[0, j_\xi - 1] \oplus (\mathbf{A}_i^{(\xi-1)}[j_\xi, j] + 1) \oplus \mathbf{A}_i^{(\xi-1)}[j, \rho+1], i = 1, 2,$$

we have $\mathcal{A}_2^{(\xi)} \prec \mathcal{A}_1^{(\xi)}$ as a consequence of (19), (22), and (21) and the lemma is proved. ▮

We now give an algorithm that, given a path decomposition $X = [X_1, \dots, X_r]$ of the graph G, for any introduce node X_i, $2 \leq i \leq r$, computes a full set of characteristics $FS(i)$ for the graph G_i, given a full set of characteristics $FS(i-1)$ for the graph G_{i-1}.

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** ρ **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 $\max(\mathbf{A}') \leq k$, then set $FS(i) \leftarrow FS(i) \cup \{(\lambda', \mathbf{A}')\}$.

5: Output $FS(i)$.

6: end.

Now we can prove the correctness of the algorithm for an introduce node in the path decomposition.

LEMMA 3.7. *Given a 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 $G_{i-1} =$*

$G[\cup_{1\leq j\leq i-1}X_j]$, *then the set* $FS(i)$ *constructed by the* Introduce-Node *algorithm is a full set of characteristics for* $G_i = G[\cup_{1\leq j\leq i}X_j]$.

Proof. We will prove first that the set $FS(i)$ computed by the algorithm is a set of X_i-characteristics for the graph G_i. We will 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 Algorithm Introduce-Node, there must be a characteristic $(\lambda, \mathbf{A}) \in FS(i-1)$ and two integers $j, 0 \leq j \leq |\lambda|$ and $m, 1 \leq m \leq |\mathbf{A}(j)|$ such that

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

As $(\lambda, \mathbf{A}) \in FS(i-1)$, it will be a X_{i-1}-characteristic for G_{i-1} and therefore, 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 3.5 we have that there exists an integer γ, $0 \leq \gamma \leq |l|$ such that

$$\mathsf{Ins}(G_{i-1}, u, X_{i-1}, N, \lambda, \mathbf{A}, j, m) = \mathsf{Com}(\mathsf{Ins}(G_{i-1}, u, V(G_{i-1}), N, l, \mathbf{Q}_{G_{i-1},l}, \gamma, 1), X_{i-1} \cup \{u\}) \tag{24}$$

From Lemma 3.4, 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 $l = l(1,\gamma) \oplus [u] \oplus l(\gamma+1, |l|)$ and therefore,

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

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 now 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. We will show that there exists a vertex ordering l'_* of G_i such that

$$C_{X_i}(G_i, l'_*) \prec C_{X_i}(G_i, l') \text{ and } C_{X_i}(G_i, l'_*) \in FS(i).$$

Set now $\gamma = \mathrm{rank}_{l'}(u) - 1$ and let $l = l'[1,\gamma] \oplus l'[\gamma+2, |l|]$. From Lemma 3.4, we have that

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

and therefore,

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

Set now $(\lambda, \mathbf{A}) = C_{X_i}(G_i, l)$. From part (ii) of Lemma 3.5 we have that there are values j and m, $0 \leq j \leq |\lambda|$ and $1 \leq m \leq |\mathbf{A}(j)|$ such that

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

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

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

Let $C_{X_{i-1}}(G_{i-1}, l_*) = (\lambda_*, \mathbf{A}_*)$. From Lemma 3.6 we have that there exists a m_* such that

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

From part (i) of Lemma 3.5, we have that there exists a γ_*, $0 \leq \gamma_* \leq |l_*|$ such that

$$\mathsf{Com}(\mathsf{Ins}(G_{i-1}, u, V(G_{i-1}), N, l_*, \mathbf{Q}_{G_{i-1},l_*}, \gamma_*, 1), X_i) = \mathsf{Ins}(G_{i-1}, u, X_{i-1}, N, \lambda_x*, \mathbf{A}_*, j, m_*) \quad (29)$$

Defining $l'_* = l_*[1, \gamma_*] \oplus [u] \oplus l_*[\gamma_* + 1, |l_*|]$ and applying Lemma 3.4 we have that

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

and therefore,

$$C_{X_i}(G_i, l'_*) = \mathsf{Com}(l'_*, \mathbf{Q}_{G_i,l'_*}, X_i) = \mathsf{Com}(\mathsf{Ins}(G_{i-1}, u, V(G_{i-1}), N, l_*, \mathbf{Q}_{G_{i-1},l_*}, \gamma_*, 1), X_i) \quad (30)$$

From (29) and (30) we have that $C_{X_i}(G_i, l'_*) = \mathsf{Ins}(G_{i-1}, u, X_{i-1}, N, \lambda_x*, \mathbf{A}_*, j, m_*)$. Since $(\mathbf{Q}_{G_{i-1}}, l_*) \in FS(i-1)$ algorithm Introduce-Node makes that $C_{X_i}(G_i, l'_*) \in FS(i)$. Moreover, 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 will now consider the case where X_i is a *forget* node. We will 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 a deletion procedure that operates inversely to procedure Ins.

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

Input: A characteristic $(\lambda, \mathbf{A})$ and a vertex $u \in V(\lambda)$.

Output: A characteristic pair $(\lambda', \mathbf{A}')$.

Assume the notations $\lambda = [u_1, \ldots, u_\rho]$ and $j = \text{rank}_\lambda(u) = j$.

1: $\lambda' \leftarrow \lambda(1, j-1) \oplus \lambda(j+1, \rho)$.

2: $\mathbf{A}' \leftarrow \mathbf{A}[0, j-2] \oplus [\tau(\mathbf{A}(j-1) \oplus \mathbf{A}(j))] \oplus \mathbf{A}[j+1, \rho]$.

3: Output $(\lambda', \mathbf{A}')$.

4: End.

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

LEMMA 3.8. *Let $(l, \mathbf{R})$ be a characteristic pair of a given graph G and let $V \subseteq V(l)$. Then, for any $v \in V$ the following holds.*

$$\mathsf{Com}(l, \mathbf{R}, V - \{v\}) \;=\; \mathsf{Del}(\mathsf{Com}(l, \mathbf{R}, V), v)$$

Observe that Lemma 3.8 can provide an alternative, recursive definition of procedure Com, based on procedure Del.

The following monotonicity result is a direct consequence of Lemma 2.6

LEMMA 3.9. *Let $(l_i, \mathbf{R}_i), i = 1, 2$ be two characteristic pairs 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 that, given a path decomposition $X = [X_1, \ldots, X_r]$ of the graph G, for any forget node X_i, $2 \leq i \leq r$, computes a full set of characteristics $FS(i)$ for the graph G_i, given a full set of characteristics $FS(i-1)$ for the graph G_{i-1}.

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: $\quad FS(i) \leftarrow FS(i) \cup \{\mathsf{Del}(\lambda, \mathbf{A}, u)\}$.

4: Output $FS(i)$.

5: end.

LEMMA 3.10. *If $FS(i-1)$ is a full set of X_{i-1}-characteristics then the set $FS(i)$ constructed by the above algorithm is a full set of X_i-characteristics for G_i.*

Proof. As $G_i = G_{i-1}$ we will use the notation G for both of them. We will also denote as u the forget vertex of G_i. We will prove first that $FS(i)$ is a set of X_i-characteristics for G. We need to prove that there exists a vertex ordering l of G where

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

for any $(\lambda', \mathbf{A}') \in FS(i)$. As $(\lambda', \mathbf{A}')$ has been constructed by 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 G such that

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

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

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

and using (33) and Lemma 3.8 we have that $C_{X_i}(l, G) = \mathsf{Com}(l, \mathbf{Q}_{G,l}, X_i) = (\lambda', \mathbf{A}')$.

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

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

From Lemma 3.8 we have that

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

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

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

Using now Lemma 3.9 we can rewrite (35) as follows.

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

Applying again Lemma 3.8 we have that

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

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

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

the output of $\mathsf{Del}(\mathsf{Com}(l_*, \mathbf{Q}_{G,l_*}, X_{i-1}), u)$ will be one of the characteristics included in $FS(i)$. Therefore, $C_{X_i}(G,l_*) \in FS(i)$ and this completes the proof of the lemma. ∎

4. MAKING THE DECISION ALGORITHM CONSTRUCTIVE

Suppose now that, given a path decomposition $X = [X_1, \ldots, X_r]$ of G with bounded width, after running the algorithm described in the previous subsections we know that a graph G has cutwidth at most k, i.e., the computed set $FS(r)$ is not empty. We will now describe a way to construct a vertex ordering of G with cutwidth at most k. By observing the working of the algorithm, it follows that there exist a sequence of characteristics, $(\lambda_1, \mathbf{A}_1), (\lambda_2, \mathbf{A}_2) \ldots, (\lambda_r, \mathbf{A}_r)$, called a *witness path*, such that

1. $(\lambda_1, \mathbf{A}_1) = ([x_{\mathbf{start}}], [[0],[0]])$ is the unique characteristic of the vertex ordering consisting of the unique vertex in the *starting node* $X_1 = \{x_{\mathbf{start}}\}$,

2. $(\lambda_r, \mathbf{A}_r)$ is some characteristic in $FS(r)$, and

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

Let us show now how to compute a layout with cutwidth $\leq k$, in linear time, given a witness path

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

Let h, $1 < h < r$, and l_h be a vertex ordering such that $C_{X_h}(G_i, l_h) = (\lambda_h, \mathbf{A}_h)$. Assume that

$$l_h = [v_1^h, \ldots, v_{|l_h|}^h] \text{ and}$$
$$\mathbf{A}_h = [A_0^h, \ldots, A_{|\mathbf{A}_h|}^h] \text{ where } A_j^h = [a_1^{h,j}, \ldots, a_{|A_h|}^{h,j}].$$

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

Let κ be the minimum number such that $h < \kappa \leq r$ 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 $\phi_h : \mathcal{P}_h \rightarrow \{0, 1, \ldots, |l_h|\}$ such that $\phi_h(j, m) = \gamma$ implies that

$$\mathsf{Ins}(G_h, u^h, X_h, N_h, \lambda_h, \mathbf{A}_h, j, m) = \mathsf{Com}(\mathsf{Ins}(G_h, u^h, V(G_h), N_h, l_h, \mathbf{Q}_{G_h, l_h}, \gamma, 1), X_h) \quad (38)$$

Our definition is recursive. We assume that for some h, $1 \leq h \leq r-1$, l_h and ϕ_h are known. We will show that l_{h+1}, ϕ_{h+1} can be defined (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 $0 \leq j \leq |\lambda_h|$ and $1 \leq m \leq |\mathbf{A}(j)|$. From 38 and the proof of Lemma 3.7, we have that, if $\gamma = \phi_h(j, m)$, then constructing l_{h+1} so that

$$l_{h+1} = l_h[1, \gamma] \oplus [u^h] \oplus l_h[\gamma + 1, |l_h|]$$

we have that

$$C_{X_{h+1}}(G_{h+1}, l_{h+1}) = (\lambda_{h+1}, \mathbf{A}_{h+1}).$$

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

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

and, 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

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

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 number $a_m^{h,j}$ of a typical sequence A_j^h of $\mathbf{A}_h$ with the value $\gamma = \phi_h(a_m^{h,j}), 0 \leq \gamma |l_h|$. 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 $l_1 = [x_{\mathbf{start}}]$ 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 $C_{X_r}(G_r, l_r) \in FS(r)$.

4.1. Conclusions on cutwidth

Notice that, because of Lemma 3.2, the algorithms Introduce-Node and Forget-Node run in $O(1)$ time when k and w are fixed. We resume the results of the previous subsections in the following.

THEOREM 4.1. *For all k, $w \geq 1$ there exists an algorithm that, given 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 $O(|V(G)| + |X|)$ time.*

Lemma 2.2 and Theorem 4.1, along with the results in [6] and [3] give the following:

THEOREM 4.2. *For all $k \geq 0$, there exists 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 $O(|V(G)|)$ time.*

5. FINAL REMARKS

It is known (e.g. see [8]) 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 assumed to be constants.

Moreover, one can easily observe that, in the more general case where pathwidth$(G) \leq w$ and the maximum degree of G is at most $\leq \beta \cdot \log n$ for $\beta \geq 0, n = |V(G)|$, Lemma 3.2 bounds the number of different characteristics by $O(w!(\frac{8}{3})^{w+1}n^{2\beta w(w+1)})$. The complexity of Algorithm Introduce-Node is $O(w^2 w!(\frac{8}{3})^{w+1}n^{2\beta w(w+1)+1})$ as step **2** requires $O(w!(\frac{8}{3})^{w+1}n^{2\beta w(w+1)})$ repetitions, step **3** requires $O(w)$ repetitions, step **4** requires $O(n)$ repetitions, and the call of Procedure Ins in step **5** requires $O(w)$ steps. Similarly, the complexity of Algorithm Forget-Node is $O(ww!(\frac{8}{3})^{w+1}n^{2\beta w(w+1)})$. Therefore, we can conclude that for any constant w 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 $\beta \cdot \log n, \beta \geq 0$ and pathwidth $\leq w, w \geq 0$.

We believe that further results can be achieved. In particular, we conjecture that there exits 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. For an analogous result concerning the computation of pathwidth for graphs with bounded degree, see Section 7 of [4].

REFERENCES

1. K. R. Abrahamson and M. R. Fellows. Finite automata, bounded treewidth and well-quasiordering. In N. Robertson and P. Seymour, editors, *Proceedings of the AMS Summer Workshop on Graph Minors, Graph Structure Theory, Contemporary Mathematics vol. 147*, pages 539–564. American Mathematical Society, 1993.

2. H. L. Bodlaender. Improved self-reduction algorithms for graphs with bounded treewidth. *Disc. Appl. Math.*, 54:101–115, 1994.

3. H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *Siam Journal on Computing*, 25:1305–1317, 1996.

4. H. L. Bodlaender and T. Kloks. Efficient and constructive algorithms for the pathwidth and treewidth of graphs. *J. Algorithms*, 21:358–402, 1996.

5. H. L. Bodlaender and D. M. Thilikos. Constructive linear time algorithms for branchwidth. In P. Degano, R. Gorrieri, and A. Marchetti-Spaccamela, editors, *Proceedings 24th International Colloquium on Automata, Languages, and Programming, ICALP'97*, pages 627–637. Springer-Verlag, Lecture Notes in Computer Science, Vol. 1256, 1997.

6. H. L. Bodlaender and D. M. Thilikos. Computing small search numbers in linear time. Technical Report Technical Report No. UU-CS-1998-05, Dept. of Computer Science, Utrecht University, 1998.

7. F. R. K. Chung. On the cutwidth and topological bandwidth of a tree. *SIAM J. Alg. Disc. Meth.*, 6:268–277, 1985.

8. F. R. K. Chung and P. D. Seymour. Graphs with small bandwidth and cutwidth. *Disc. Math.*, 75:113–119, 1989.

9. G. Even, J. Naor, S. Rao, and B. Schieber. Divide-and-conquer approximation algorithms via spreading metrics. In *Proc. 36th Symp. on Foundations of Computer Science (FOCS)*, pages 62–71, 1995.

10. M. R. Fellows and M. A. Langston. On well-partial-order theory and its application to combinatorial problems of VLSI design. *SIAM J. Disc. Meth.*, 5:117–126, 1992.

11. M. R. Fellows and M. A. Langston. On search, decision and the efficiency of polynomial-time algorithms. *J. Comp. Syst. Sc.*, 49:769–779, 1994.

12. M. R. Garey and D. S. Johnson. *Computers and Intractability, A Guide to the Theory of NP-Completeness.* W.H. Freeman and Company, New York, 1979.

13. E. Korach and N. Solel. Tree-width, path-width and cutwidth. *Disc. Appl. Math.*, 43:97–101, 1993.

14. F. S. Makedon, C. H. Papadimitriou, and I. H. Sudborough. Topological bandwidth. *SIAM J. Alg. Disc. Meth.*, 6:418–444, 1985.

15. F. S. Makedon and I. H. Sudborough. On minimizing width in linear layouts. *Disc. Appl. Math.*, 23:243–265, 1989.

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

17. N. Robertson and P. D. Seymour. Graph minors. XXIII. The nash-williams immersion conjecture. To appear.

18. N. Robertson and P. D. Seymour. Graph minors. XIII. The disjoint paths problem. *J. Comb. Theory Series B*, 63:65–110, 1995.

19. M. Yannakakis. A polynomial algorithm for the min-cut linear arrangement of trees. *J. ACM*, 32:950–988, 1985.