Computing Small Search Numbers in Linear Time*

Hans L. Bodlaender Dimitrios M. Thilikos†
Department of Computer Science, Utrecht University,
P.O. Box 80.089, 3508 TB Utrecht, the Netherlands
e-mail: {hansb,sedthilk}@cs.ruu.nl

Abstract

Let $G = (V, E)$ be a graph. The *linear-width* of G is defined as the smallest integer k such that E can be arranged in a linear ordering $(e_1, \ldots, e_r)$ such that for every $i = 1, \ldots, r-1$, there are at most k vertices both incident to an edge that belongs to $\{e_1, \ldots, e_i\}$ as to an edge that belongs to $\{e_{i+1}, \ldots, e_r\}$. For each fixed constant k, a linear time algorithm is given, that decides for any graph $G = (V, E)$ whether the linear-width of G is at most k, and if so, finds the corresponding ordering of E. Linear-width has been proven to be related with the following graph searching parameters: mixed search number, node search number, and edge search number. A consequence of this is that we obtain for fixed k, linear time algorithms that check whether a given graph can be mixed, node, or edge searched with at most k searchers, and if so, output the corresponding search strategies.

1 Introduction

In this paper, we study algorithmic aspects of a relatively new graph parameter: *linear-width*. Apart from having interest on its own, this parameter has close relationships with some well known graph searching parameters: mixed search number, node search number, and edge search number, as well as well known parameters related or equal to these parameters, like pathwidth, vertex separation number, and proper pathwidth. The results for linear-width allow us to rederive some old results for some of these parameters, and obtain similar new results for the other parameters, basically all as consequences of one algorithm.

The *linear-width* of a graph G is defined to be the least integer k such that the edges of G can be arranged in a linear ordering $(e_1, \ldots, e_r)$ in such a way that for every $i = 1, \ldots, r \Leftrightarrow 1$, there are at most k vertices that incident to at least one edge that belongs to $\{e_1, \ldots, e_i\}$ and that are incident to at least one edge that belongs to $\{e_{i+1}, \ldots, e_r\}$. Linear-width was first mentioned by Thomas in [26] and is strongly connected with the notion of *crusades* introduced by Bienstock

*This research was partially supported by ESPRIT Long Term Research Project 20244 (project ALCOM IT: *Algorithms and Complexity in Information Technology*).

†The second author was supported by the Training and Mobility of Researchers (TMR) Program, (EU contract no ERBFMBICT950198).

and Seymour in [2]. Linear-width can be seen as "a linear variant of branch-width", in the same way as pathwidth can be seen as "a linear variant of treewidth". In [25], it is proved that several variants of problems appearing on graph searching can be reduced to the problem of computing linear-width.

In a graph searching game a graph represents a system of tunnels where an agile, fast, and invisible fugitive is resorting. We desire to capture this fugitive by applying a search strategy while using the fewest possible searchers. Briefly said, the search number of a graph is the minimum number of searchers a searching strategy requires in order to capture the fugitive. Several variations on the way the fugitive can be captured during a search, define the the parameters of the *edge, node,* and *mixed search number* of a graph (namely, $\mathrm{es}(G)$, $\mathrm{ns}(G)$, and $\mathrm{ms}(G)$). The first graph searching game was introduced by Breisch [6] and Parsons [18] and is the one of *edge searching*. *Node searching* appeared as a variant of edge searching and was introduced by Kirousis and Papadimitriou in [14]. Finally, *mixed searching* was introduced in [24] and [2] and is a natural generalisation of the two previous variants (for the formal definitions see Subsection 4.1 – for other results concerning search games on graphs see [1, 7, 9, 15, 16, 23].)

The problems of computing $\mathrm{es}(G), \mathrm{ns}(G), \mathrm{ms}(G)$, or linear-width$(G)$ is NP-complete (see [16, 14, 24, 25]). On the other hand, since all of these parameters are closed under taking of minors, we know (see e.g. [3, 19, 20, 22, 21]) that, for any k, there exists a linear algorithm that given a graph G checks whether $\mathrm{es}(G), \mathrm{ns}(G), \mathrm{ms}(G)$, or linear-width$(G)$ is at most k (in other words, all these parameters are "fixed parameter tractable"). Unfortunately, the above result is not constructive i.e. does not provide a way to *construct* the corresponding algorithm (see [10, 11]). Therefore, it is highly desired to have constructive "fixed parameter results" for the aforementioned parameters.

In this paper we carry out the above task by constructing such a linear algorithm for linear-width. This algorithm can be directly transfered to a linear algorithm for node, edge, and mixed search number thanks to their connection (see [25]) with linear-width.

So far, such a linear time algorithm has been constructed (see [3, 4]) only for the parameters of treewidth and pathwidth (actually, the result in [3, 4] can be directly transfered to the node search number which is known to be equal to the node search number minus one – see [12, 13, 17]). To be precise, [3, 4] state that for fixed k, one can determine in linear time whether a given graph has pathwidth at most k, and if so, find a path decomposition of minimum width. This algorithm first finds a tree decomposition of width at most k, if existing (and if not existing, then the pathwidth is also larger than k), and then uses this tree decomposition to solve the problem, using the result in [4] that states that for fixed k and l, one can test whether the pathwidth of a graph G is at most l, and if so, find a minimum width path decomposition, assuming that G is given together with a tree decomposition of width at most k. The technique in this paper uses a similar technique: we first determine a path decomposition of bounded width (if such a path decomposition does not exist, we know that the linear-width also is not bounded), and then apply an algorithm, presented in this paper, that, given such a path decomposition, solves our

problem. It can actually be avoided to work with tree decompositions by a modification of the algorithm in [3].

An other parameter related to linear-width is *branch-width*. In another paper [5], we give a similar algorithm for branch-width. That algorithm uses the techniques of this paper as a building block for a more complicated algorithm.

This paper is organised as follows. In Section 2, we give some preliminary results and definitions. The main algorithm is presented in Section 3. The consequences of the main algorithm can be found in Section 4.

2 Definitions and preliminary results

We first give a number of definitions and notations, dealing with sequences (i.e., ordered sets) $\omega = (\omega_1, \ldots, \omega_r)$ of objects from some set $\mathcal{O}$, where $\mathcal{O}$ can be a set of numbers, sequences of numbers, vertices, vertex sets, or of edges. We use the notation $(\omega)_{i,j}$, $1 \leq i \leq j \leq r$ to denote the subsequence $(\omega_i, \ldots, \omega_j)$ of ω. If "$\circ$" is an operation defined on $\mathcal{O}$, we use for any $\omega' \in \mathcal{O}$, the notation $(\omega)_{i,j} \circ \omega'$, $i \leq j$, to denote the sequence $(\omega_i \circ \omega', \ldots, \omega_j \circ \omega')$. Given two sequences $\omega^i, i = 1, 2$, defined on $\mathcal{O}$, where $\omega^i = (\omega^i_1, \ldots, \omega^i_{r_i}), i = 1, 2$ we set $\omega^1 \circ \omega^2 = (\omega^1_1 \circ \omega^2_1, \ldots, \omega^1_{r_1} \circ \omega^2_{r_2})$. We also set $\omega^1 \oplus \omega^2 = (\omega^1_1, \ldots, \omega^1_{r_1}, \omega^2_1, \ldots, \omega^2_{r_2})$. We finally denote the length of a sequence ω by $|\omega|$.

Unless mentioned otherwise, we will assume all graphs considered in this paper to be undirected and without parallel edges or self-loops. Given a graph $G = (V, e)$ we denote its vertex set and edge set with $V(G)$ and $E(G)$ respectively. If $V' \subseteq V(G)$, we call the graph $(V', \{\{v, u\} \in E(G) : v, u \in V')$ *subgraph of G induced by V'* and we denote it by $G[V']$. For any edge set $H \subseteq E(G)$ we denote by $V(H)$ the set of vertices that are incident to edges of H (i.e. $V(H) = \cup_{e \in H} e$). The degree of a vertex v in graph G is the number of edges containing it and is denoted by $d_G(v)$. We call a vertex *pendant* when it has degree 1. We call an edge of a graph *pendant* when it contains a pendant vertex.

2.1 Pathwidth

A *path decomposition* of a graph $G = (V, e)$ is a sequence $X = (X_1, \ldots, X_{|X|})$ of subsets of $V(G)$ such that for every $V = \bigcup_{1 \leq i \leq r} X_i$, for every $\{v, w\} \in e$, there is an i, $1 \leq i \leq r$, $\{v, w\} \subseteq X_i$, and for all i, i', i'', $1 \leq i \leq i' \leq i''$, $X_i \cap X_{i''} \subseteq X_{i'}$. (The sets X_i are called the *nodes* of the path decomposition.) The width of a path decomposition $X = (X_i, 1 \leq i \leq |X|)$ equals $\max_{1 \leq i \leq |X|} \{|X_i| \Leftrightarrow 1\}$. The *pathwidth* of a graph G is the minimum width over all path decompositions of G.

Let $X = (X_i, 1 \leq i \leq |X|)$ be a path decomposition of a graph G. We say that X is *nice* if $|X_1| = 1$ and $\forall_{2 \leq i \leq |X|} \ (X_i \Leftrightarrow X_{i-1}) \cup (X_{i-1} \Leftrightarrow X_i)| = 1$. The following lemma is easy.

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

Let X_i be a node of a nice path decomposition X such that $i \geq 1$. We say that X_i is an *introduce* (*forget*) node if $|X_i \Leftrightarrow X_{i-1}| = 1$ ($|X_{i-1} \Leftrightarrow X_i| = 1$). 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.

2.2 Linear-width

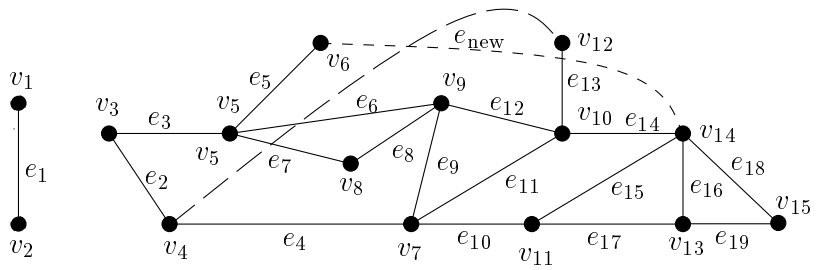
The linear-width of a graph is defined as follows. Let G be a graph and $l = (e_1, \ldots, e_{|E(G)|})$ an ordering of $E(G)$. Define $\delta_l(e_i) = V((l)_{1,i}) \cap V((l)_{i+1,|E(G)|})$ (i.e. $\delta_l(e_i)$ is the set of vertices in $V(G)$ that are endpoints of an edge in $(l)_{1,i}$ and also of an edge in $(l)_{i+1,|E(G)|}$). The linear-width of an ordering l is $\max_{1 \leq i \leq |E(G)|}\{|\delta_l(e_i)|\}$. The linear-width of a graph is the minimum linear-width over all the orderings of $E(G)$.

Given an edge ordering l we define $P(l) = (\delta_l(e_1), \ldots, \delta_l(e_r))$. Moreover, we define $\mathbf{Q}(l) = ((|\delta_l(e_1)|), \ldots, (|\delta_l(e_{r-1})|))$ Notice that $P(l)$ is a sequence of sets and $\mathbf{Q}(l)$ is a sequence of sequences of numbers each containing only one element. We use this, somewhat overloaded, definition for reasons of consistency with terminology that will be introduced later. We finally define the pendant sequence of l as $H(l) = (e_j \cap A(G), 1 \leq j \leq |E(G)|)$ where $A(G) = \{v \in V(G) : d_G(v) = 1\}$. It is not hard to prove that if l has linear-width at most k, then $P(l) \cup H(l)$ form a path decomposition of G with width at most k. Therefore, we we have the following.

Lemma 2 *For every graph G, pathwidth$(G) \leq$ linear-width(G).*

Given a vertex set V, a vertex $x \in V$, and a sequence of vertex sets $I = (I_1, \ldots, I_{|I|})$ where $I_i \subseteq V, 1 \leq i \leq r$ and $\cup_{i=1,\ldots,r} I_i = V$, we define $F_I(x) = \min\{m, 1 \leq m \leq r \mid x \in I_m\}$ and $L_I(x) = \max\{m, 1 \leq m \leq r \mid x \in I_m\}$.

As an example we consider the following graph G along with an ordering $l = (e_1, \ldots, e_{19})$ of its edges.



One can now observe that if $P(l) = (P_1, \ldots, P_{19})$ and $H(l) = (H_1, \ldots, H_{19})$, then

$$
\begin{array}{llll}
P_1 = \emptyset, & P_2 = \{v_3, v_4\}, & P_3 = \{v_4, v_5\}, & P_4 = \{v_5, v_7\}, \\
P_5 = \{v_5, v_7\}, & P_6 = \{v_5, v_7, v_9\}, & P_7 = \{v_7, v_8, v_9\}, & P_8 = \{v_7, v_9\}, \\
P_9 = \{v_7, v_9\}, & P_{10} = \{v_7, v_9, v_{11}\}, & P_{11} = \{v_9, v_{10}, v_{11}\}, & P_{12} = \{v_{10}, v_{11}\}, \\
P_{13} = \{v_{10}, v_{11}\}, & P_{14} = \{v_{11}, v_{14}\}, & P_{15} = \{v_{11}, v_{14}\}, & P_{16} = \{v_{11}, v_{14}, v_{13}\}, \\
P_{17} = \{v_{13}, v_{14}\}, & P_{18} = \{v_{13}, v_{15}\}, & P_{19} = \emptyset, &
\end{array}
$$

$H_1 = \{v_1, v_2\}, H_2 = \ldots = H_4 = \emptyset, H_5 = \{v_6\}, H_6 = \ldots = H_{12} = \emptyset, H_{13} = \{v_{12}\}, H_{14} = \ldots = H_{10} = \emptyset.$ and $\mathbf{Q}(l) = ((0), (2), (2), (2), (2), (3), (3), (2), (2), (2), (3), (2), (2), (2), (2), (3), (2), (2), (0))$.

Suppose now that we introduce in G a new edge $v_{\text{new}} = \{v_6, v_{14}\}$ and that l' is the edge ordering of $G' = (V(G), E(G) \cup \{\{v_6, v_{14}\}\})$ obtained from l after inserting v_{new} between $e_9 = \{v_7, v_9\}$ and $e_{10} = \{v_7, v_{11}\}$. Then, if $P(l') = (P'_1, \ldots, P_9, P_{\text{new}}, P_{10}, \ldots, P_{19})$ and $H(l') = (H'_1, \ldots, H'_9, H_{\text{new}}, H'_{10}, \ldots, H'_{19})$ we have that

$$
\begin{array}{llll}
P_1 = \emptyset, & P_2 = \{v_3, v_4\}, & P_3 = \{v_4, v_5\}, & P_4 = \{v_5, v_7\}, \\
P_5 = \{v_5, v_6, v_7\}, & P_6 = \{v_5, v_6, v_7, v_9\}, & P_7 = \{v_6, v_7, v_8, v_9\}, & P_8 = \{v_6, v_7, v_9\}, \\
P_9 = \{v_6, v_7, v_9\}, & P_{\text{new}} = \{v_7, v_9, v_{14}\}, & P_{10} = \{v_7, v_9, v_{11}, v_{14}\}, & P_{11} = \{v_9, v_{10}, v_{11}, v_{14}\}, \\
P_{12} = \{v_{10}, v_{11}, v_{14}\}, & P_{13} = \{v_{10}, v_{11}, v_{14}\}, & P_{14} = \{v_{11}, v_{14}\}, & P_{15} = \{v_{11}, v_{14}\}, \\
P_{16} = \{v_{11}, v_{13}, v_{14}\}, & P_{17} = \{v_{13}, v_{14}\}, & P_{18} = \{v_{13}, v_{15}\}, & P_{19} = \emptyset,
\end{array}
$$

$H'_1 = \{v_1, v_2\}, H'_2 = \ldots = H'_9 = H_{\text{new}} = H'_{10} = \ldots H'_{12} = \emptyset, H'_{13} = \{v_{12}\}, H'_{14} = \ldots = H'_{19} = \emptyset,$ and $\mathbf{Q}(l) = ((0), (2), (2), (2), (3), (4), (4), (3), (3), (3), (4), (4), (3), (3), (2), (2), (3), (2), (2), (0))$.

We can now describe the changes that are happening after introducing $e_{\text{new}} = \{v_2, v_7\}$ as follows.

(a) Set P_9 is duplicated and thus a new set $P_{\text{new}} = P_9$ is inserted between P_9 and P_{10}.

(b) v_6 is added to all the sets of $P(l)$ from the set having as index the one of the unique set containing it in $H(l)$ until P_9 i.e. from $P_{L_{H(l)}(v_6)} = P_5$ until P_8.

(c) v_{14} is added to all the sets of $P(l)$ from P_{new} until one set before the first set containing it in $P(l)$ i.e. from P_{new} until $P_{F_{P(l)}(v_{14})-1} = P_{13}$.

Suppose now that $e_{\text{new}} = \{v_4, v_{12}\}$ and that e_{new} is inserted between e_7 and e_8. In such a case the changes are the the following.

(a) Set P_7 is duplicated and a new set $P_{\text{new}} = P_7$ is inserted between P_7 and P_8.

(b) v_4 is added to all the sets of $P(l)$ from the set after the last set containing v_4 until P_7 i.e. from $P_{L_{P(l)}(v_4)+1} = P_4$ until P_7.

(c) v_{12} is added to all the sets of $P(l)$ from P_{new} until the set having as indice the one of the unique set containing it in $H(l)$ decreased by one i.e. from P_{new} until $P_{F_{H(l)}(v_7)-1} = P_{12}$.

The observations of the above examples are utilised in the following lemma (see also Lemma 14).

Lemma 3 *Let $l = (e_1, \ldots, e_r)$ be an edge ordering of a graph G and $e_{\text{new}} \subseteq V(G)$ be an edge not in $E(G)$. Let $P(l) = (P_1, \ldots, P_r)$. Suppose that $l' = (e_1, \ldots, e_i, e_{\text{new}}, e_{i+1}, \ldots, e_r)$ for some $i, 1 \leq i \leq r \Leftrightarrow 1$ (l' is an edge ordering of $G' = (V(G), E(G) \cup \{e_{\text{new}}\})$). Then the following hold.*

(i) If $H(l) = (H_1, \ldots, H_r)$ and $H(l') = (H'_1, \ldots, H'_{r+1})$ then

$$\forall_{1 \leq j \leq i}\ H'_j = H_j \Leftrightarrow e, H'_{i+1} = e \cap A(G), \forall_{i+2 \leq j \leq r+1}\ H'_j = H_{j-1} \Leftrightarrow e$$

(ii) If $P(l) = (P_1, \ldots, P_r)$ and $P(l') = (P'_1, \ldots, P'_{r+1})$ then

$$\forall_{1 \leq j \leq i}\ P'_j = P_j \cup \{x \in e_{\text{new}} \mid L_{P(l) \cup H(l)}(x) \leq j\}, \text{ and} \quad (1)$$

$$\forall_{i+1 \leq j \leq r+1}\ P'_j = P_{j-1} \cup \{x \in e_{\text{new}} \mid j \Leftrightarrow 1 \leq F_{P(l) \cup H(l)}(x) \Leftrightarrow 1\}, \quad (2)$$

Proof. (i) follows immediately from the definition of $H(l)$. In order to prove (ii) we notice first that $\forall_{1 \leq j \leq i}\ P'_j \subseteq P_j$ and $\forall_{i+1 \leq j \leq r+1}\ P'_j \subseteq P_{j-1}$. As only vertices from e can be introduced we have that $\forall_{1 \leq j \leq i}\ P'_j \Leftrightarrow P_j \subseteq e_{\text{new}}$ and $\forall_{i+1 \leq j \leq r+1}\ P'_j \Leftrightarrow P_{j-1} \subseteq e_{\text{new}}$. Assume that $x \in \cup_{P \in P(l)} P$. Using the definition of linear-width we can observe that for any $x \in e_{\text{new}}$ the following hold.

(i) If $1 \leq j \leq i$ then $x \in P'_j \Leftrightarrow P_j$ iff $L_{P(l)}(x) + 1 \leq j$.

(ii) If $i + 1 \leq j \leq r + 1$ then $x \in P'_j \Leftrightarrow P_{j-1}$ iff $j \Leftrightarrow 1 \leq F_{P(l)}(x) \Leftrightarrow 1$.

Let now $x \notin \cup_{P \in P(l)} P$. Clearly x is a pendant vertex. We observe the following.

(iii) If $1 \leq j \leq i$ then $x \in P'_j \Leftrightarrow P_j$ iff $L_{H(l)}(x) \leq j$.

(iv) If $i + 1 \leq j \leq r + 1$ then $x \in P'_j \Leftrightarrow P_{j-1}$ iff $j \Leftrightarrow 1 \leq F_{H(l)}(x) \Leftrightarrow 1$.

It is now sufficient to observe that (i),(iii) and (ii),(iv) certify the correctness of (1) and (2) respectively. □

3 A decision algorithm for linear-width

In this section, we give for every pair of integer constants k, l, an algorithm that, given a graph G with a path decomposition of width at most l, decides whether G has linear-width at most k.

3.1 Sequences of integers

If $A = (a_1, \ldots, a_{|A|})$ is a sequence of integers, we define $\max(A) = \max_{1 \leq i \leq |A|}\{\max(a_i)\}$ and for any integer t we set $A + t = (a_1 + t, \ldots, a_{|A|} + t)$. The *typical sequence* $\tau(A)$ of a sequence of integers A is the sequence obtained after iterating the following operations, until none is possible any more.

(i) If for some $i, 1 \leq i \leq |A| \Leftrightarrow 1$ $a_i = a_{i+1}$, then set $A = (a_1, \ldots, a_{|A|}) \leftarrow (a_1, \ldots, a_i, a_{i+2}, \ldots, a_{|A|})$.
(ii) If the sequence contains two elements a_i and a_j such that $j \Leftrightarrow i \geq 2$ 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 $A = (a_1, \ldots, a_{|A|}) \leftarrow (a_1, a_i, a_j, \ldots, a_{|A|})$.

As an example we mention that if $A = (5,5,6,7,7,7,4,4,3,5,4,6,8,2,9,3,4,6,7,2,7,5,4,4,6,4)$, then $\tau(A) = (5,7,3,8,2,9,2,7,4)$.

The following has been proved in [4].

Lemma 4 ([4]) *The number of different typical sequences of integers in* $\{0,1,\ldots,L\}$ *is at most* $\frac{8}{3}2^{2L}$.

Let A, B be two typical sequences where $A = (a_1,\ldots,a_{|A|})$ and $B = (b_1,\ldots,b_{|B|})$. If $|A| = |B|$ then we say that $A \leq B$ if $\forall_{1\leq i\leq |A|}\ a_i \leq b_i$. We define the set of extensions of A as

$$e(A) = \{A^* = (a^*_1,\ldots,a^*_{A^*}) \mid \exists_{1=t_1<\ldots<t_{|A|+1}}\ \forall_{1\leq i\leq |A|}\ \forall_{t_i\leq k<t_{i+1}}\ a^*_k = a_i\}.$$

We say that $A \prec B$ if there exist extensions $A^* \in e(A), B^* \in e(B)$ such that $|A^*| = |B^*|$ and $A^* \leq B^*$. For example if $A = (5,7,3,8)$ and $B = (1,7,2,6,4)$ then $B \prec A$ because $A^* = (5,7,7,7,3,8,8,8,8)$ is an extension of A, $B^* = (1,7,2,6,4,4,4,4,4)$ is an extension of A, and $B^* \leq A^*$. The proof of the following Lemma can be found in [4] (Lemma 3.19).

Lemma 5 *Let* A, A', B, B' *be typical sequences such that* $A \prec A'$ *and* $B \prec B'$. *Then* $\tau(A \oplus B) \prec \tau(A' \oplus B')$.

Suppose now that $\mathbf{A} = (A_1,\ldots,A_{|\mathbf{A}|})$ and $\mathbf{B} = (B_1,\ldots,B_{|\mathbf{B}|})$ are two sequences of sequences such that $|\mathbf{A}| = |\mathbf{B}| = r$. We say that $\mathbf{A} \prec \mathbf{B}$ if $\forall_{1\leq i\leq r}\ A_i \prec B_i$. Finally, for any integer t we set $\mathbf{A} + t = (A_1 + t,\ldots,A_{|\mathbf{A}|} + t)$ and $\max(\mathbf{A}) = \max_{1\leq i\leq |\mathbf{A}|}\{\max(A_i)\}$.

3.2 Characteristic of a typical triple

Let X be a path decomposition of a graph G and let X_i be a node in X where $1 \leq i \leq |X|$. Let $S = (S_1,\ldots,S_{|S|})$ be a sequence of vertex sets where $S_j \subseteq X_i, 1 \leq j \leq |S|$, $D = (D_1,\ldots,D_{|D|})$ a sequence of subsets of X_i where $|D_j| \leq 2, 1 \leq j \leq |F|$ and $\forall_{1\leq j<i\leq |D|} D_i \cap D_j = \emptyset$, and $\mathbf{T} = (T_1,\ldots,T_{|\mathbf{T}|})$ be a sequence of typical sequences such that $|\mathbf{T}| = |S|$. We call such a triple $(S, D, \mathbf{T})$ a *typical triple* of X_i.

Given a typical triple $(S, D, \mathbf{T})$, of X_i we define its *characteristic* $C(S, D, \mathbf{T})$ of it as a typical triple $(I, K, \mathbf{A})$ defined as follows.

Let $(r_1,\ldots,r_{q+1})$ be the sequence with $1 = r_1 < \cdots < r_{q+1} = |S| + 1$ and such that

(a) $\forall_{1\leq j\leq q}\ \forall_{r_j\leq h<r_{j+1}}\ (S_h = S_{r_j}$ and $D_h = D_{r_j})$,

(b) $\forall_{1\leq j<q}\ (S_{r_{j+1}} \neq S_{r_j}$ or $D_{r_{j+1}} \neq D_{r_j})$.

We call the pair (I, K) where $I = (S_{r_1},\ldots,S_{r_q})$ and $K = (D_1,\ldots,D_{r_q})$ the *interval model* of S and D. Notice that, as $\forall_{1\leq j<i\leq |D|} D_i \cap D_j = \emptyset$, we can observe that if $D_{r_j} \neq \emptyset$ then $r_{j+1} = r_j + 1$ and $r_{j-1} = r_j \Leftrightarrow 1$.

For any $j, 1 \leq j \leq q$, let $A_j = (T_{r_j} \oplus \cdots \oplus T_{r_{j+1}-1})$. We also set $\mathbf{A} = (A_1,\ldots,A_{|A|}) \leftarrow (\tau(A_1),\ldots,\tau(A_q))$ and we call the typical triple $(I, K, \mathbf{A})$ the *characteristic* of the typical triple $(S, D, \mathbf{T})$.

Given two typical triples $(S^i, D^i, \mathbf{T}^i), j = 1, 2$ we say that $(S^1, D^1, \mathbf{T}^1) \equiv (S^2, , D^2, \mathbf{T}^2)$ if they have the same characteristic i.e. $C(S^1, D^1, \mathbf{T}^1) = C(S^2, D^2, \mathbf{T}^2)$. We also say that $(S^1, D^1, \mathbf{T}^1) \prec (S^2, D^2, \mathbf{T}^2)$ if $S^1 = S^2$, $D^1 = D^2$ (i.e. they have the same interval model), and $\mathbf{T}^1 \prec \mathbf{T}^2$. It is easy to see that relation "$\prec$" is transitive i.e. if $\mathbf{T}^1 \prec \mathbf{T}^2$ and $\mathbf{T}^2 \prec \mathbf{T}^3$ then $\mathbf{T}^1 \prec \mathbf{T}^3$. We also extending the definition of "$\oplus$" so that whenever $(S^i, D^i, \mathbf{T}^i), i = 1, 2$ are two typical triples, the typical triple $(S^1 \oplus S^2, D^1 \oplus D^2, \mathbf{T}^1 \oplus \mathbf{T}^2)$ is denoted by $(S^1, D^1, \mathbf{T}^1) \oplus (S^2, D^2, \mathbf{T}^2)$.

3.3 Characteristic of an ordering

Let X be a path decomposition of a graph G and X_i some node of X where $1 \leq i \leq |X|$. We define $V_i = \cup_{1 \leq j \leq i} X_j$ and $G_i = G[V_i]$.

Let l be an edge ordering of G_i with linear-width at most k. Let also $P(l) = (P_1, \ldots, P_{P(l)})$. We define the *restriction* of l on X_i as $R(l) = (P_j \cap X_i, 1 \leq j \leq |P(l)|)$.

We define the *characteristic* $C(l)$ of l as $C(R(l), H(l), \mathbf{Q}(l))$. If $C(R(l), H(l), \mathbf{Q}(l)) = (I, K, \mathbf{A})$, we say that the pair (I, K) is an interval model at X_i and $\mathbf{A}$ is the corresponding sequence of typical sequences. From now on, whenever we consider an edge ordering we will associate it with a node X_i of a path decomposition.

As an example of a characteristic, we mention that if l is the ordering of the graph G of the example of Section 2.2, and $X_i = \{v_5, v_7, v_{10}, v_{12}\}$ then $C(R(l), H(l), \mathbf{Q}(l)) = (I, K, \mathbf{A})$ where

$$
\begin{array}{rcccccccccc}
I = (& \emptyset, & \{v_5\}, & \{v_5, v_7\}, & \{v_7\}, & \emptyset, & \{v_{10}\}, & \{v_{10}\}, & \emptyset, & \{v_{13}\}, & \emptyset &), \\
K = (& \emptyset, & \emptyset, & \emptyset, & \emptyset, & \emptyset, & \emptyset, & \{v_{12}\}, & \emptyset, & \emptyset, & \emptyset &), \\
\mathbf{A} = (& (0,2), & (2), & (2,3), & (3,2), & (2), & (3,2), & (2), & (2), & (3,2), & (0) &).
\end{array}
$$

Using now Lemma 3.1 of [4] we can prove the following.

Lemma 6 *For each node X_i where $|X_i| \leq l$ the number of different interval models at X_i is bounded by $(2l+3)^{2l+3}$. The number of elements of any interval model is bounded by $2l+3$.*

Using Lemmata 6 and 4 we have the following.

Lemma 7 *Let $X = (X_1, \ldots, X_{|X|})$ be a path decomposition of G with width at most l. Let $X_i, 1 \leq i \leq |X|$ be some node in X. The number of different characteristics of all possible edge orderings of G_i with linear-width at most k, is at most $(2l+3)^{2l+3} \cdot (\frac{8}{3} \cdot 2^{2k})^{2l+3}$.*

The following procedure defines function Com, that maps a typical triple $(I, K, \mathbf{A})$ to another typical triple that is a 'compression' of the input pair (see Lemma 8.)

PROCEDURE $\mathsf{Com}(I, K, \mathbf{A})$
Input: A typical triple $(I, K, \mathbf{A})$
Output: A typical triple $(I, K, \mathbf{A})$.

1: Set $r = |I| = |K| = |\mathbf{A}|$.

2: Apply the following operation until it is no longer possible.

- If $\exists_{1\leq h\leq r-1} : (I_h = I_{h+1}$ and $K_{h+1} = \emptyset)$ then set
  $I = (I_1, \ldots, I_{r-1}) \leftarrow (I_1, \ldots, I_h, I_{h+2}, \ldots, I_r)$,
  $K = (K_1, \ldots, K_{r-1}) \leftarrow (K_1, \ldots, K_h, K_{h+2}, \ldots, K_r)$,
  $\mathbf{A} = (A_1, \ldots, A_{r-1}) \leftarrow (A_1, \ldots, A_{h-1}, \tau(A_h \oplus A_{h+1}), A_{h+2}, \ldots, A_r)$, and
  $r \leftarrow r \Leftrightarrow 1$.

3: end.

The following lemmata follow from the definition of the characteristic of an ordering.

Lemma 8 *Let l be an edge ordering of some G_i. Then $C(l) = \mathsf{Com}(R(l), H(l), \mathbf{Q}(l))$.*

Lemma 9 *Let $(I^i, K^i, \mathbf{A}^i), i = 1, 2$ be two typical triples such that $(I^1, K^1, \mathbf{A}^1) \equiv (I^2, K^1, \mathbf{A}^2)$. Then $\mathsf{Com}(I^1, K^1, \mathbf{A}^1) = \mathsf{Com}(I^2, K^2, \mathbf{A}^2)$.*

The following lemmata follow easily from the definitions and Lemma 5.

Lemma 10 *Let $(I^i, K^i, \mathbf{A}^i), i = 1, 2$ be two typical triples such that $(I^1, K^1, \mathbf{A}^1) \prec (I^2, K^2, \mathbf{A}^2)$. Then $\mathsf{Com}(I^1, K^1, \mathbf{A}^1) \prec \mathsf{Com}(I^2, K^2, \mathbf{A}^2)$.*

Lemma 11 *Let $(I^i, K^i, \mathbf{A}^i), i = 1, \ldots, 4$ be two typical triples where $(I^1, K^1, \mathbf{A}^1) \equiv (I^2, K^2, \mathbf{A}^2)$ and $(I^3, K^3, \mathbf{A}^3) \equiv (I^4, K^4, \mathbf{A}^4)$. Then $(I^1 \oplus I^3, K^1 \oplus K^3, \mathbf{A}^1 \oplus \mathbf{A}^3) \equiv (I^2 \oplus I^4, K^4 \oplus K^4, \mathbf{A}^2 \oplus \mathbf{A}^4)$.*

Lemma 12 *Assume that $(I, K, \mathbf{A}) = (I^1, K^1, \mathbf{A}^1) \oplus (I^2, K^2, \mathbf{A}^2)$. Then $\mathsf{Com}(I, K, \mathbf{A}) = \mathsf{Com}($ $\mathsf{Com}(I^1, K^1, \mathbf{A}^1) \oplus \mathsf{Com}(I^2, K^2, \mathbf{A}^2))$.*

A set $FS(i)$ of characteristics of edge orderings of a graph G_i (i is a node of the path decomposition) with width at most k is called a *full set of characteristics* at i if for each linear ordering l of G_i with linear-width at most k, there is a edge ordering l' such that $C(l') \prec C(l)$ and $C(l') \in FS(i)$, i.e. the characteristic of l' is in $FS(i)$. The following lemma can be derived directly from the definitions.

Lemma 13 *A full set of characteristics at i is non-empty if and only if the linear-width of G_i is at most k. If some full set of characteristics at i is non-empty, then every full set of characteristics at this node is non-empty.*

An important consequence of Lemma 13 is that the linear-width of G is at most k, if and only if any full set of characteristics of $G_{|X|} = G$ 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 of X_{i-1} is given.

3.4 Introducing an edge

The following procedure is an important ingredient of our algorithm. Given a typical triple $(I, K, \mathbf{A})$, it "inserts an edge": after the mth position in the jth sequence in $\mathbf{A}$, an edge is inserted; S is the set of the vertices that are endpoints of the edge. Both I and $\mathbf{A}$ are modified accordingly.

PROCEDURE $\mathsf{Int}(I, K, \mathbf{A}, j, m, S)$
Input A typical triple $(I, K, \mathbf{A})$ where $I = (I_i, 1 \leq i \leq |I|)$, $K = (K_i, 1 \leq i \leq |K|)$, $\mathbf{A} = (A_i, 1 \leq i \leq |I| = |K|)$, and $A_i = (a^i_j, 1 \leq j \leq |A_i|), 1 \leq i \leq |I|$, two integers j, m where $1 \leq j \leq |I|, m, 1 \leq m \leq |I_j|$, and a vertex set S.
Output A typical triple $(I, K, \mathbf{A})$.

1: (*Splitting step*) Apply the following three steps.
- Set $I' = (I'_1, \ldots, I'_{|I|+1}) \leftarrow (I_1, \ldots, I_{j-1}, I_j, I_j, I_{j+1}, \ldots, I_{|I|})$
- Set $K' = (K'_1, \ldots, K'_{|I|+1}) \leftarrow (K_1 \Leftrightarrow S, \ldots, K_{j-1} \Leftrightarrow S, K_j \Leftrightarrow S, S \cap A(G), K_{j+1} \Leftrightarrow S, \ldots, K_{|I|} \Leftrightarrow S)$
- Set $\mathbf{A}' = (A'_1, \ldots, A'_{|I|+1}) \leftarrow (A_1, \ldots, A_{j-1}, (a^j_1, \ldots, a^j_m), (a^j_m, \ldots, a^j_{|A_j|}), A_{j+1}, \ldots, A_{|I|})$.

2: (*Insertion step*) For any $x \in S$, apply the following.
- For any $h, 1 \leq h \leq |I| + 1, \min\{L_{I \cup K}(x), j\} \leq h \leq \max\{j, F_{I \cup K}(x) \Leftrightarrow 1\}$: set $I_h \leftarrow I_h \cup \{x\}$ and $A_h \leftarrow A_h + 1$

3: Output $I', K', \mathbf{A}'$.

4: end.

The above procedure has two main steps: Step **1** duplicates the jth position of sequence I, redefine K so that a new edge, having as endpoint the vertices in S, is considered, and "splits" $\mathbf{A}$ on the mth position of its jth sequence. Step **2** inserts vertices from S in some of the elements of I and increases the numbers of some of the sequences in $\mathbf{A}$. Notice that using the above procedure we can add elements in any position of a typical triple except from the beginning. This drawback can be overcome if we enhance the input graph with an isolated edge which will always be on the beginning of an edge ordering. The next lemma proves that procedure Int is useful in computing the characteristic of the layout occurring after the insertion of a new edge.

Lemma 14 *Let G be a graph rooted on X_i and G' be the graph obtained by G after introducing an edge e_{new} with both endpoints in X_i. Let also $l = (e_1, \ldots, e_{|l|})$ be an edge ordering of G and $l' = (e_1, \ldots, e_\gamma, e_{\text{new}}, e_{\gamma+1}, \ldots, e_{|l|})$ for some $\gamma, 1 \leq \gamma \leq |l|$. Then $C(l') = \mathsf{Com}(\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, e_{\text{new}}))$.*

Proof. From Lemma 8 we have that $C(l') = \mathsf{Com}(R(l'), H(l'), \mathbf{Q}(l'))$ and thus it is sufficient to prove that $(R(l'), H(l'), \mathbf{Q}(l')) = \mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, e_{\text{new}})$. Let $H(l) = (H_1, \ldots, H_{|l|})$ and $P(l) = (P_1, \ldots, P_{|l|})$. Observe that $\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, e_{\text{new}})$ enters step **2** with the typical triple $(I', K', \mathbf{A}')$ where

$$I' = (P_1 \cap X_i, \ldots, P_\gamma \cap X_i, P_\gamma \cap X_i, \ldots, P_{|l|}) \cap X_i),$$

$$K' = (H_1 \Leftrightarrow e_{\text{new}}, \ldots, H_\gamma \Leftrightarrow e_{\text{new}}, e_{\text{new}} \cap A(G), H_{\gamma+1} \Leftrightarrow e_{\text{new}}, \ldots, H_{|l|} \Leftrightarrow e_{\text{new}}),$$

$$\mathbf{A}' = ((|P_1|), \ldots, (|P_\gamma|), (|P_\gamma|), (|P_{|l|}|)).$$

Using Lemma 3.i, we have that $K' = H(l')$. It remains to prove that ($R(l')$ and $\mathbf{Q}(l')$ are correctly computed after step **2**. This follows easily from relations (1) and (2) of Lemma 3.ii. □

Lemma 15 *Let l be an ordering and assume that $(I, K, \mathbf{A}) = C(R(l), H(l), \mathbf{Q}(l))$. Let also $R(l) = (R_1, \ldots, R_{|l|})$, $F(l) = (F_1, \ldots, F_{|l|})$, $\mathbf{Q}(l) = ((q_1), \ldots, (q_{|l|}))$, $I = (I_1, \ldots, I_{|I|})$, and $\mathbf{A} = (A_1, \ldots, A_{|I|})$ where $A_j = (a_1^j, \ldots, a_{|A_j|}^j), 1 \leq j \leq |I|$. Then for any vertex set S, the following hold.*

(i) For any $j, m, 1 \leq j \leq |I|, 1 \leq m \leq |A_j|$, there exists an integer $\gamma, 1 \leq \gamma \leq |l|$ such that $\mathsf{Com}(\mathsf{Int}(I, K, \mathbf{A}, j, m, S)) = \mathsf{Com}(\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, S))$.

(ii) For any $\gamma, 1 \leq \gamma \leq |l|$ and any $S \subseteq X_i$, there exist two integers $j, m, 1 \leq j \leq |I|, 1 \leq m \leq |A_j|$ such that $\mathsf{Com}(\mathsf{Int}(I, K, \mathbf{A}, j, m, S)) \prec \mathsf{Com}(\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, S))$.

Proof (i) Notice first that there exist $\sigma, \rho, 1 \leq \sigma \leq \rho \leq |l|$ such that

$$(I, K, \mathbf{A})_{1,j-1} \equiv (R(l), H(l), \mathbf{Q})_{1,\sigma-1}, \quad (3)$$

$$(I, K, \mathbf{A})_{j,j} \equiv (R(l), H(l), \mathbf{Q})_{\sigma,\rho}, \quad (4)$$

$$(I, K, \mathbf{A})_{j+1,|I|} \equiv (R(l), H(l), \mathbf{Q})_{\rho+1,|l|}, \quad (5)$$

From (4) we have that $\tau(q_\sigma, \ldots, q_\rho) = A_j$ and therefore there exist a $\gamma, \sigma \leq \gamma \leq \rho$ such that

$$\tau(q_\sigma, \ldots, q_\gamma) = (a_1^j, \ldots, a_m^j), \quad (6)$$

$$\tau(q_\gamma, \ldots, q_\rho) = (a_m^j, \ldots, a_{|A_j|}^j). \quad (7)$$

We claim that $\mathsf{Com}(\mathsf{Int}(I, K, \mathbf{A}, j, m, S)) = \mathsf{Com}(\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, S))$. Clearly, from Lemma 9, it is sufficient to prove that $\mathsf{Int}(I, K, \mathbf{A}, j, m_1, S) \equiv \mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, S)$. In order to prove this claim we will proceed applying in parallel the steps of procedures $\mathsf{Int}(I, K, \mathbf{A}, j, m, S)$ and $\mathsf{Int}(R(l), H(l), \mathbf{Q}, \gamma, 1, S)$. We we will show that the corresponding typical triples constructed after each step are equivalent.

Clearly $(I, K, \mathbf{A}) \equiv (R(l), H(l), \mathbf{Q}(l))$. Let $(I'^i, K'^i, \mathbf{A}'^i), i = 1, 2$ be the triples created after step **1** of $\mathsf{Int}(I, K, \mathbf{A}, K, j, m, S)$ and $\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, S)$ respectively. For notational simplicity we assume that $j_1 = j$ and $j_2 = \gamma$. We observe the following.

$$(I'^1, K'^1, \mathbf{A}'^1)_{1,j_1} \equiv (I'^2, K'^2, \mathbf{A}'^2)_{1,j_2}, \quad (8)$$

$$(I'^1, K'^1, \mathbf{A}'^1)_{j_1+1,|I'^1|} \equiv (I'^2, K'^2, \mathbf{A}'^2)_{j_2+1,|I'^2|}. \quad (9)$$

(8) follows from (3) and (6). (9) follows from (5) and (7).

Suppose now that the main command of step **2** is executed for some vertex $x \in S$. Observe that it is sufficient to prove that conditions (8) and (9) hold for the resulting typical triple as well. In order to prove this, we first observe that $L_{I'^1 \cup K'^1}(x) \leq j_1 \Leftrightarrow L_{I'^2 \cup K'^2}(x) \leq j_2$ and $F_{I'^1 \cup K'^1}(x) > j_1 + 1 \Leftrightarrow F_{I'^2 \cup K'^2}(x) > j_2 + 1$. We also assume that either $L_{I'^1 \cup K'^1}(x) \leq j_1$ or $F_{I'^1 \cup K'^1}(x) > j_1 + 1$ as, in any other case, no change in $(I'^i, K'^i, \mathbf{A}'^i), i = 1, 2$ happens. We examine the following cases.

Case (i) $L_{I'^1 \cup K'^1}(x) \leq j_1$. We notice first the following.

$$(I'^1, K'^1, \mathbf{A}'^1)_{1, L_{I'^1 \cup K'^1}(x)-1} \equiv (I'^2, K'^2, \mathbf{A}'^2)_{1, L_{I'^2 \cup K'^2}(x)-1}, \quad (10)$$

$$(I'^1, K'^1, \mathbf{A}'^1)_{L_{I'^1 \cup K'^1}(x), j_1} \equiv (I'^2, K'^2, \mathbf{A}'^2)_{L_{I'^2 \cup K'^2}(x), j_2}, \quad (11)$$

$$(I'^1 \cup \{x\}, K'^1, \mathbf{A}'^1 + 1)_{L_{I'^1 \cup K'^1}(x), j_1} \equiv (I'^2 \cup \{x\}, K'^2, \mathbf{A}'^2 + 1)_{L_{I'^2 \cup K'^2}(x), j_2}. \quad (12)$$

(10) and (11) follow easily from (8) and the definition of a characteristic triple. (12) follows directly from (11). Suppose now that $(I'^i_{\text{new}}, K'^i_{\text{new}}, \mathbf{A}'^i_{\text{new}}), i = 1, 2$ are the produced typical pairs. Clearly,

$$I'^i_{\text{new}} = (I'^i)_{1, L_{I'^i \cup K'^i}(x)-1} \oplus (I'^i \cup \{x\})_{L_{I'^i \cup K'^i}(x), j_i} \oplus (I'^i)_{j_i+1, |I'^i|}, \; K'^i_{\text{new}} = K'^i, \text{ and}$$
$$\mathbf{A}'^i_{\text{new}} = (\mathbf{A}'^i)_{1, L_{I'^i \cup K'^i}(x)-1} \oplus (\mathbf{A}'^i \cup \{x\})_{L_{I'^i \cup K'^i}(x), j_i} \oplus (\mathbf{A}'^i)_{j_i+1, |I'^i|}, \; i = 1, 2.$$

Using now (10) and (12), and Lemma 11, one can see that (8) and (9) also hold if we replace $(I'^i, K'^i, \mathbf{A}'^i), i = 1, 2$ with $(I'^i_{\text{new}}, K'^i_{\text{new}}, \mathbf{A}'^i_{\text{new}}), i = 1, 2$.

Case 2. $F_{I'^1 \cup K'^1}(x) > j_1 + 1$. Similar to (i).

(ii) Notice that there exists some $j, 1 \leq j \leq |I|$ such that $I_j = R_\gamma$. Moreover, there exist $\sigma, \rho, 1 \leq \sigma \leq \rho \leq |l|$ such that relations (3)–(5) of (i) hold. We first claim that we can exclude the case where there exists a $m, 1 \leq m \leq |A_j|$ such that relations (6) and (7) hold as well. Indeed, this case is trivial as it is sufficient to follow the steps of proof of (i) and finally conclude that $\mathsf{Com}(\mathsf{Int}(I, K, \mathbf{A}, j, m, S)) = \mathsf{Com}(\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, S))$ which is a stronger version of the required relation.

We now observe that there exist some $m', 1 \leq m' \leq |A_j| \Leftrightarrow 1$ and two integers $\sigma', \rho', \sigma \leq \sigma' < \gamma < \rho' \leq \rho$ such that

$$\tau(q_\sigma, \ldots, q_{\sigma'}) = (a^j_1, \ldots, a^j_{m'}), \quad (13)$$

$$\tau(q_{\rho'}, \ldots, q_\rho) = (a^j_{m'+1}, \ldots, a^j_{|A_j|}). \quad (14)$$

Clearly, $q_{\sigma'} \neq q_{\rho'}$. If $q_{\sigma'} > q_{\rho'}$ then we set $m \leftarrow m' + 1$, otherwise we set $m \leftarrow m'$. We claim that, in any case, $\mathsf{Com}(\mathsf{Int}(I, K, \mathbf{A}, j, m, S)) \prec \mathsf{Com}(\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, S))$. We will examine the case where $q_{\sigma'} > q_{\rho'}$ (the other case is similar).

Notice that $\forall_{\sigma'+1 \leq h \leq \rho'-1} \; q_h > a^j_m$. Using this fact and relations (13) and (14) we have the following.

$$\tau(q_\sigma, \ldots, q_\gamma) \succ (a^j_1, \ldots, a^j_m), \quad (15)$$

$$\tau(q_\gamma, \ldots, q_\rho) \succ (a^j_m, \ldots, a^j_{|A_j|}). \quad (16)$$

As in the proof of (i), we will apply in parallel the steps of procedures $\mathsf{Int}(I, K, \mathbf{A}, j, m, S)$ and $\mathsf{Int}(R(l), H(l), \mathbf{Q}, \gamma, 1, S)$.

Clearly $(I, K, \mathbf{A}) \equiv (R(l), H(l), \mathbf{Q}(l))$. Let $(I'^i, K'^i, \mathbf{A}'^i), i = 1, 2$ be the typical triples created after step **1** of $\mathsf{Int}(I, \mathbf{A}, K, j, m, S)$ and $\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, S)$ respectively. For notational simplicity we assume that $j_1 = j$ and $j_2 = \gamma$. We observe the following

$$(I'^1, K'^1, \mathbf{A}'^1)_{1,j_1-1} \equiv (I'^2, K'^2, \mathbf{A}'^2)_{1,j_2-1}, \tag{17}$$

$$(I'^1, K'^1, \mathbf{A}'^1)_{j_1,j_1} \prec (I'^2, K'^2, \mathbf{A}'^2)_{j_2,j_2}, \tag{18}$$

$$(I'^1, K'^1, \mathbf{A}'^1)_{j_1+1,j_1+1} \prec (I'^2, K'^2, \mathbf{A}'^2)_{j_2+1,j_2+1}, \tag{19}$$

$$(I'^1, K'^1, \mathbf{A}'^1)_{j_1+2,|I'^1|} \equiv (I'^2, K'^2, \mathbf{A}'^2)_{j_2+2,|I^2|}. \tag{20}$$

Relations (17), (18), (19), and (20) follow from (3), (15), (5), and (16) respectively.

We claim now that the typical triples constructed after the application of step **2** are satisfying relations (17)–(20) as well. We omit the proof as it is similar to the one used for the corresponding claim in the proof of (i). In what follows we will use the notation $(I'^i, K'^i, \mathbf{A}'^i), i = 1, 2$ in order to denote the outputs of $\mathsf{Int}(I, K, \mathbf{A}, K, j, m, S)$ and $\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, S)$. From Lemma 12 we have that

$$\begin{aligned}\mathsf{Com}(I'^i, K'^i, \mathbf{A}'^i) = \; & \mathsf{Com}(\mathsf{Com}((I'^i, K'^i, \mathbf{A}'^i)_{1,j_1-1}) \oplus \\ & \mathsf{Com}((I'^i, K'^i, \mathbf{A}'^i)_{j_1,j_1}) \oplus \\ & \mathsf{Com}((I'^i, K'^i, \mathbf{A}'^i)_{j_1+1,j_1+1}) \oplus \\ & \mathsf{Com}((I'^i, K'^i, \mathbf{A}'^i)_{j_1+1,|I'^i|})), i = 1, 2.\end{aligned}$$

The result follows now directly using (17), (18), (19), (20), and Lemmata 9 and 10. □

As an example we assume that R_a, R_b, R_c, R_e, R_f are pairwise different vertex sets and we set

$R(l) = (\ R_a, R_a, R_a, R_b, R_b, R_b, R_b, R_c, R_c, R_d, R_d, R_d, R_d, R_d, R_d, R_d, R_d, R_e, R_e, R_e, R_e, R_f, R_f, R_f\ )$

$H(l) = (\ \emptyset,\ \emptyset\ )$

$\mathbf{Q}(l) = (\ (5),(5),(2),(4),(8),(9),(3),(5),(5),(2),(7),(8),(4),(3),(5),(2),(6),(4),(5),(2),(1),(2),(2),(9)\ (9)\ ).$

$q_\sigma \quad q_{\sigma'} \quad q_\gamma \quad q_{\rho'} \quad q_\rho$

If $(I, \mathbf{A}) = C(R(l), \mathbf{Q}(l))$, then we clearly have

$I_j = I_4$

$a_m = a_3^4$

$I = (\ R_a,\ R_b,\ R_c,\ R_d,\ R_e\ R_f\ )$

$K = (\ \emptyset,\ \emptyset\ \emptyset\ \emptyset\ \emptyset\ \emptyset\ )$

$\mathbf{A} = (\ (5,2),\ (4,9,3)\ (5)\ (2,8,2,6,4)\ (5,1,2)\ (2,9)\ )$

Assume that $j = 2$ and $m = 2$. Using the proof of Lemma 15.i we can verify that if $\gamma = 6$ then $\mathsf{Com}(\mathsf{Int}(I, K, \mathbf{A}, j, m, S)) = \mathsf{Com}(\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, S))$. Let now $\gamma = 14$. Following the proof of Lemma 15.ii we have that, if $j = 4$ and $m = 3$, then $\mathsf{Com}(\mathsf{Int}(I, K, \mathbf{A}, j, m, S)) \prec \mathsf{Com}(\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, S))$.

Lemma 16 *Let* $(I^i, K^i, \mathbf{A}^i), i = 1, 2$ *be two typical triples such that* $(I^1, K^1, \mathbf{A}^1) \succ (I^2, K^2, \mathbf{A}^2)$. *Let also* $I^i = (I^i_j, 1 \leq j \leq |I^i|), i = 1, 2$, $K^i = (K^i_j, 1 \leq j \leq |K^i|), i = 1, 2$, $\mathbf{A}^i = (A^i_j, 1 \leq i \leq |A^i|), i = 1, 2$ $A^i_j = (a^{j,i}_h, 1 \leq h \leq |A^i_j|), 1 \leq j \leq |A^i|, i = 1, 2$. *Then, for any* $S \subseteq X_i, j, 1 \leq j \leq |I^1| = |I^2|$ *and* $m_1, 1 \leq m_1 \leq |A^1_j|$ *there exist a* $m_2, 1 \leq m_2 \leq |A^2_j|$, *such that* $\mathsf{Com}(\mathsf{Int}(I^1, K^1, \mathbf{A}^1, j, m_1, S)) \succ \mathsf{Com}(\mathsf{Int}(I^2, K^2, \mathbf{A}^2, j, m_2, S))$.

Proof Since $A^1_j \succ A^2_j$, we can easily observe that there exist $m_2, 1 \leq m_2 \leq |A^i_j|$ such that

$$(a^1_{j,1}, \ldots, a^i_{j,m_1}) \succ (a^2_{j,1}, \ldots, a^2_{j,m_2}) \quad (21)$$

$$(a^1_{j,m_1}, \ldots, a^i_{j,|A^1_j|}) \succ (a^2_{j,m_2}, \ldots, a^2_{j,|A^2_j|}). \quad (22)$$

From Lemma 10, it is sufficient to prove that $\mathsf{Int}(I^1, K^1, \mathbf{A}^1, j, m_1, S) \prec \mathsf{Int}(I^2, K^2, \mathbf{A}^2, \gamma, a, S)$. Let $(I'^i, K'^i \mathbf{A}'^i), i = 1, 2$ be the typical triples created after step **1** (as in the proof of Lemma 15 we follow "in parallel" the steps of $\mathsf{Int}(I^i, K^i, \mathbf{A}^i, j, m_i, S) i = 1, 2$). We notice the following.

$$(I'^1, K'^1, \mathbf{A}'^1)_{1,j_1} \succ (I'^2, K'^2, \mathbf{A}'^2)_{1,j_2}, \quad (23)$$

$$(I'^1, K'^1, \mathbf{A}'^1)_{j_1+1,|I'^1|} \succ (I'^2, K'^2, \mathbf{A}'^2)_{j_2+1,|I'^2|}. \quad (24)$$

(23) follows from (21) and (24) follows from (22).

We claim now that the typical triples constructed after the application of step **2** are satisfying relations (23) and (24) as well. We omit the proof as it is similar to those used for the corresponding claims in the proofs of Lemma 15. □

3.5 A full set for an introduce node

We will now consider the case where X_i is an *introduce* node.

Clearly $V_i = V_{i-1} \cup \{x\}$ where $x \notin V_{i-1}$. Suppose that $E_x = \{e_1, \ldots, e_r\}, 0 \leq r \leq |X_{i-1}| \leq l$ is the set of edges incident to x in G_i (notice that, $\cup_{e \in E_x} e \subseteq X_i$). If $E_x = \emptyset$, then, we simply set $FS(i) = FS(i \Leftrightarrow 1)$. What remains is to examine the case where $|E_x| \geq 1$.

We define $G^p_i = (V(G_i), E(G_{i-1}) \cup \{e_1, \ldots, e_p\}), 0 \leq p \leq r$. Clearly, $FS(i \Leftrightarrow 1)$ is a full set of characteristics for $G^0_i = G_{i-1}$. Notice also that $G_i = G^p_i$. Suppose that we have a full set of characteristics $FS(i, p \Leftrightarrow 1)$ for $G^{p-1}_i, 1 \leq p \leq r$ (which is the case when $p = 1$). It is sufficient to give a $O(1)$ time algorithm constructing a full set of characteristics $FS(i, p)$ for G^p_i.

ALGORITHM Introduce-edge

Input: A full set of characteristics $FS(i, p \Leftrightarrow 1)$ for G^{i-1}_p.
Output: A full set of characteristics $FS(i, p)$ for G^i_p.

1: Initialise $FS(i, p) = \emptyset$.
2: For each characteristic $(I, K, \mathbf{A}) \in FS(i, p \Leftrightarrow 1)$ where $I = (I_j : 1 \leq j \leq |I|)$ and $\mathbf{A} = (A_j : 1 \leq j \leq |I|)$ apply step **3**.

3: For any set $I_j \in I$ let $A_j = (a^j_1, \ldots, a^j_{|A_j|})$ and apply step **4**.
4: For any $a^j_m, 1 \leq m \leq |A_j|$ in A_j, set $(I', K', \mathbf{A}') = \mathsf{Com}(\mathsf{Int}(I', K', \mathbf{A}', j, m, e_p))$ and if $\max(\mathbf{A}') \leq k$, then set $FS(i,p) \leftarrow FS(i,p) \cup \{(I', K', \mathbf{A}')\}$.
5: end.

Lemma 17 *The set $FS(i,p)$ constructed by the above algorithm is a full set of characteristics.*

Proof. We will prove first that $FS(i,p)$ is a set of characteristics. Let $(I', K', \mathbf{A}') \in FS(i,p)$. We will show that there exists an edge ordering of G^i_p with this characteristic. Clearly, as $(I', K', \mathbf{A}')$ is constructed by the algorithm above, there must be a characteristic $(I, K, \mathbf{A}) \in FS(i, p \Leftrightarrow 1)$, and two integers m and j such that

$$\mathsf{Com}(\mathsf{Int}(I, K, \mathbf{A}, j, m, e_p)) \quad = \quad (I', K'\mathbf{A}'). \tag{25}$$

Let $l = (e_1, \ldots, e_{|l|})$ be an edge ordering of G^{p-1}_i that has $(I, K, \mathbf{A})$ as characteristic. From Lemma 15.(i) we have that there exist a $\gamma, 1 \leq \gamma \leq |l|$ such that

$$\mathsf{Com}(\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, e_p)) \quad = \quad \mathsf{Com}(\mathsf{Int}(I, K, \mathbf{A}, j, m, e_p)). \tag{26}$$

We now claim that $l' = (e'_1, \ldots, e'_{|l'|}) = (e_1, \ldots, e_\gamma, e_p, e_{\gamma+1}, \ldots, e_r)$ is an edge ordering of G^p_i such that $C(l') = (I', K', \mathbf{A}')$. Indeed, from Lemma 14, we have that

$$C(l') \quad = \quad \mathsf{Com}(\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, e_p)). \tag{27}$$

and, now, $C(l') = (I', K'\mathbf{A}')$ follows directly from (25), (26) and (27).

In what follows, we prove that $FS(i,p)$ is a full set of characteristics. Let l' be an edge ordering of G^p_i of width at most k. We will show that there exists an edge ordering l'_* of G^p_i such that $C(l'_*) \prec C(l')$ and $C(l'_*) \in FS(i,p)$. Suppose that $l' = (g_1, \ldots, g_\gamma, e_p, g_{\gamma+1}, \ldots, g_r)$. We set $l = (g_1, \ldots, g_\gamma, g_{\gamma+1}, \ldots, g_r)$. Let $C(l) = (I, K, \mathbf{A})$. From Lemma 14 we have that

$$\mathsf{Com}(\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, e_p)) \quad = \quad C(l'). \tag{28}$$

From Lemma 15.(ii) we have there exist j, m, $1 \leq j \leq |I|$, $1 \leq m \leq |I_j|$ such that

$$\mathsf{Com}(\mathsf{Int}(I, K, A, j, m, e_p)) \prec \mathsf{Com}(\mathsf{Int}(R(l), H(l), \mathbf{Q}(l), \gamma, 1, e_p)). \tag{29}$$

As $FS(i, p \Leftrightarrow 1)$ is a full set of characteristics, we have that there exists an edge ordering l_* of G^{p-1}_i such that $C(l_*) \prec C(l)$ and $C(l_*) \in FS(i, p \Leftrightarrow 1)$. Let $C(l_*) = (I_*, K_*, \mathbf{A}_*)$. From Lemma 16 we have that there exist a m' such that

$$\mathsf{Com}(\mathsf{Int}(I_*, K_*, \mathbf{A}_*, j, m', e_p)) \quad \prec \quad \mathsf{Com}(\mathsf{Int}(I, K, \mathbf{A}, j, m, e_p)). \tag{30}$$

Let $l_* = (g^*_1, \ldots, g^*_{|l_*|})$ (notice that $|l| = |l_*| = |E(G^{p-1}_i)|$) and we set $l'_* = (g^*_1, \ldots, g^*_{\gamma^*}, e_p, g^*_{\gamma^*+1}, \ldots, g^*_{|l_*|})$. From Lemma 15.(i) there exist a $\gamma^*, 1 \leq \gamma^* \leq |l_*|$ such that

$$\mathsf{Com}(\mathsf{Int}(R(l_*), H(l_*), \mathbf{Q}(l_*), \gamma^*, 1, e_p) \quad = \quad \mathsf{Com}(\mathsf{Int}(I_*, K_*, \mathbf{A}_*, j, m', e_p)). \tag{31}$$

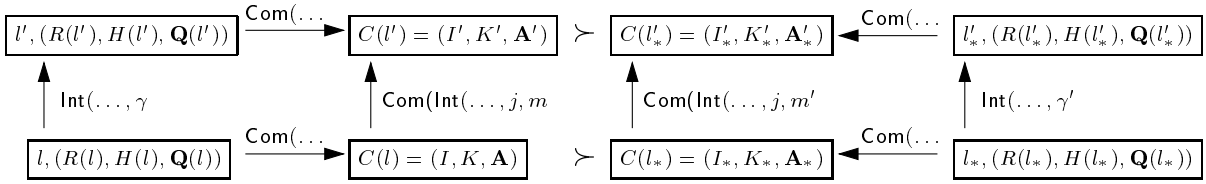


Figure 1: A scheme for the proof of Lemma 17.

Moreover, from Lemma 14 we have that

$$C(l'_*) \quad = \quad \mathsf{Com}(\mathsf{Int}(R(l_*), H(l_*), \mathbf{Q}(l_*), \gamma^*, 1, e_p)). \tag{32}$$

From (32) and algorithm Introduce-edge we have that $C(l'_*) \in FS(i,p)$. Finally, from (28), (29), (30), (31), and (32), we have that $C(l'_*) \prec C(l')$. □

3.6 A full set for a forget node

We will now consider the case where X_i is a *forget* node. Clearly, $G_i = G_{i-1}$ and there exists a unique vertex $v \in X_{i-1}$ with $v \notin X_i$. We call this vertex v *forgotten*. Given a full set of characteristics $F(i \Leftrightarrow 1)$ for X_{i-1}, the following algorithm computes a full set of characteristics $F(i)$ for X_i.

ALGORITHM Forget-Vertex

Input: A full set of characteristics $FS(i \Leftrightarrow 1)$ for G_{i-1} and a forgotten vertex x.
Output: A full set of characteristics $FS(i)$ for G_i.

1: Initialise $FS(i) = \emptyset$.
2: For any $(I, K, \mathbf{A}) \in FS(i \Leftrightarrow 1)$ set $FS(i) \leftarrow FS(i) \cup \{\mathsf{Com}(I \Leftrightarrow \{x\}, K \Leftrightarrow \{x\}, \mathbf{A})\}$.
3: end.

Lemma 18 *The set $FS(i)$ constructed by the above algorithm is a full set of characteristics.*

Proof. We will prove first that $FS(i)$ is a set of characteristics. Let $(I', K', \mathbf{A}') \in FS(i)$. We will prove that there is an edge ordering of G_i with this characteristic. From algorithm Forget-Vertex, there exist some $(I, K, \mathbf{A}) \in FS(i \Leftrightarrow 1)$ such that

$$\mathsf{Com}(I \Leftrightarrow \{x\}, K \Leftrightarrow \{x\}, \mathbf{A}) \quad = \quad (I', K', \mathbf{A}'). \tag{33}$$

Let l be an ordering of G_{i-1} where $C(l) = (I, K, \mathbf{A})$. From the obvious fact that $(R(l), H(l), \mathbf{Q}(l)) \equiv (I, K, \mathbf{A})$ we can easily derive the following.

$$(R(l) \Leftrightarrow \{x\}, H(l) \Leftrightarrow \{x\}, \mathbf{Q}(l)) \quad \equiv \quad (I \Leftrightarrow \{x\}, K \Leftrightarrow \{x\}, \mathbf{A}). \tag{34}$$

Notice that $l' = l$ is an ordering of G_i as well. We claim that $C(l') = (I', K', \mathbf{A}')$. From the definitions we have that $R(l') = R(l) \Leftrightarrow \{x\}, H(l') = H(l) \Leftrightarrow \{x\}$, and $\mathbf{Q}(l') = \mathbf{Q}(l)$. Therefore, we easily have that

$$\mathsf{Com}(R(l'), H(l'), \mathbf{Q}(l')) \quad = \quad \mathsf{Com}(R(l) \Leftrightarrow \{x\}, H(l) \Leftrightarrow \{x\}, \mathbf{Q}(l)). \tag{35}$$

Using now (33), (34), and (35), and Lemma 9 we conclude that $C(l') = \mathsf{Com}(R(l'), H(l'), \mathbf{Q}(l')) = (I', K', \mathbf{A}')$.

Next we prove that $FS(i, p)$ is a full set of characteristics. Let l' be an edge ordering of G_i of width at most k. We will show that there exists an edge ordering l'_* of G_i such that $C(l'_*) \prec C(l')$ and $C(l'_*) \in FS(i)$. Observe that $l = l'$ is an edge ordering of G_{i-1} as well and set $C(l) = (I, K, \mathbf{A})$. Notice that relations (34) and (35) hold as well and finally using Lemma 9 we obtain

$$C(l') \quad = \quad \mathsf{Com}(I \Leftrightarrow \{x\}, K \Leftrightarrow \{x\}, \mathbf{A}). \tag{36}$$

As $FS(i \Leftrightarrow 1)$ is a full set of characteristics, there exist an ordering l_* of G_{i-1} such that if $C(l_*) = (I_*, K_*, \mathbf{A}_*)$ then $(I_*, K_*, \mathbf{A}_*) \prec (I, K, \mathbf{A})$. It is now easy to verify that

$$(I_* \Leftrightarrow \{x\}, K_* \Leftrightarrow \{x\}, \mathbf{A}_*) \quad \prec \quad (I \Leftrightarrow \{x\}, K \Leftrightarrow \{x\}, \mathbf{A}). \tag{37}$$

Notice now that $l'_* = l_*$ is an ordering of G_i as well. Notice also that relations (34) and (35) hold also for the case of l_* and $(I_*, K_*, \mathbf{A}_*)$ and therefore we obtain the following.

$$C(l'_*) \quad = \quad \mathsf{Com}(I_* \Leftrightarrow \{x\}, K_* \Leftrightarrow \{x\}, \mathbf{A}_*) \tag{38}$$

From Algorithm Forget-Vertex and (38) we have that $C(l'_*) \in FS(i)$. Finally, from Lemma 8 and relations (36), (37), and (38) we have that $C(l'_*) \prec C(l')$.

3.7 The decision algorithm

Using the algorithms of the previous sections we can compute a full set of characteristics for $G_1, G_2, G_3, \ldots, G_{|X|} = G$ (obviously $E(G_1) = \emptyset$). Notice that if a graph consists of a single edge $e_{\text{start}} = \{v^1_{\text{start}}, v^2_{\text{start}}\}$, its full set of characteristics is $((\{v^1_{\text{start}}, v^2_{\text{start}}\}), (\{v^1_{\text{start}}, v^2_{\text{start}}\}), ((0)))$. Using this full set of characteristics as a base we can use the procedures of the previous sections to compute the full sets for G_2, G_3, etc., in order. Note that the computation needs $O(1)$ time per node of the path decomposition, and thus in total, time linear on the number of vertices of G. After the full set for the last node has been computed, in $O(1)$ time one can decide whether the linear-width of G is at most k, as this holds if and only if this last full set $FS(|X|)$ is not empty.

3.8 Turning the decision algorithm to a constructive one

Suppose that, after running the algorithm described in the previous subsections we know that a graph G has linear-width at most k, i.e., the computed set $FS(|X|)$ is not empty. We will now describe a way to construct an edge ordering of $E(G)$ with linear-width at most k. By observing the working of the algorithm, it follows that there exist a sequence of characteristics, $(I^1, K^1, \mathbf{A}^1), (I^2, K^2, \mathbf{A}^2) \ldots, (I^n, K^n, \mathbf{A}^n)$ such that

- $(I^1, K^1, \mathbf{A}^1) = ((\{v^1_{\text{start}}, v^2_{\text{start}}\}), (\{v^1_{\text{start}}, v^2_{\text{start}}\}), ((0)))$ (the unique characteristic of the ordering consisting of the first edge considered by the algorithm),
- $(I^n, K^n, \mathbf{A}^n)$ is some characteristic in $FS(|X|)$, and
- for any i, $1 \leq i \leq n \Leftrightarrow 1$ $(I^{i+1}, K^{i+1}, \mathbf{A}^{i+1})$ was constructed after a call of either Introduce-edge or Forget-Vertex with input $(I^i, K^i, \mathbf{A}^i)$. We call such a sequence *witness path*.

If for each time a new characteristic is computed we set up a pointer to the the characteristic it was constructed from, we obviously have a suitable structure for constructing such a *witness path* in linear time.

Let $(I^i, K^i, \mathbf{A}^i)$ be a characteristic of the witness path where $\mathbf{A}^i = (A^i_1, \ldots, A^i_{|\mathbf{A}_i|})$ and $A^i_j = (a^{i,j}_1, \ldots, a^{i,j}_{|A_j|}), 1 \leq j \leq |A_i|$. Let also $l_i = (e^i_1, \ldots, e^i_{r_i})$ be an ordering such that $C(l_i) = (I^i, K^i, \mathbf{A}^i)$. We define $\phi_i : A^i_1 \oplus \cdots \oplus A^i_{|\mathbf{A}|} \to l_i$ such that

$$\phi_i(a^{i,j}_m) = e_\gamma \Leftrightarrow \mathsf{Com}(\mathsf{Int}(I^i, K^i, \mathbf{A}^i, j, m, e_\gamma)) = \mathsf{Com}(\mathsf{Int}(R(l_i), H(l_i), \mathbf{Q}(l_i), \gamma, 1, e_\gamma).$$

We maintain a data structure associating the position (determined by j and m) of each number $a^{i,j}_m$ of a typical sequence A^i_j of $\mathbf{A}^i$ with an edge $\phi_i(a^{i,j}_m)$ in l_i.

We assume that for some i, $1 \leq i \leq n \Leftrightarrow 1$ l_i and ϕ_i are known. We will show that l_{i+1}, ϕ_{i+1} can be computed in $O(1)$ time.

We first examine the case where $(I^{i+1}, K^{i+1}, \mathbf{A}^{i+1})$ was computed after a call of Introduce-edge. Let e_{new} be the new edge introduced. Clearly, $(I^i, K^i, \mathbf{A}^{i+1}) = \mathsf{Com}(\mathsf{Int}(I^i, K^i, \mathbf{A}^i, j, m, e_{\text{new}}))$ for some j, m, and $e_{new} \notin l_i$ (notice that we can consider j, m, and e_{new} as known if we maintain a back up of them during the execution of the decision algorithm). From the proof of Lemma 17, we have that, if $e_\gamma = \phi(a^{i,j}_m)$, then $l_{i+1} = (e^i_1, \ldots, e^i_\gamma, e_{\text{new}}, e^i_{\gamma+1}, \ldots, e^i_{r_i})$ is an ordering where $C(l_{i+1}) = (I^{i+1}, K^{i+1}, \mathbf{A}^{i+1})$. Finally, in order to compute the function ϕ_{i+1} we first run again $\mathsf{Int}(I^i, K^i, \mathbf{A}^i, j, m, e_{\text{new}})$ initialising function ϕ'_{i+1} so that after step **5** the unique newly introduced number $|I^*_j|$ is mapped to e_{new} and all the other numbers of the typical sequences of $\mathbf{A}'$ are mapped to the images they had "originally" in ϕ_i. Now ϕ_{i+1} is just a "projection" of ϕ'_{i+1} to the numbers of typical sequence that "survive" after the application of Com on the output of $\mathsf{Int}(I^i, K^i, \mathbf{A}^i, j, m, e_{\text{new}})$. It is easy to see that l^{i+1} and ϕ_{i+1} can be computed in $O(1)$ time.

Let now now $(I^{i+1}, K^{i+1}, \mathbf{A}^{i+1})$ was computed after a call of Forget-Vertex and let v_{old} was the forgotten vertex. Clearly, the new ordering l_{i+1} is the same as l_i and the new function ϕ_{i+1} is obtained if we initialise $\phi'_{i+1} \leftarrow \phi_i$ as a function corresponding to $(I^i \Leftrightarrow x_{\text{old}}, K^i \Leftrightarrow x, \mathbf{A}^i)$ and obtain ϕ'_{i+1} as a "projection" of ϕ'_{i+1} to the numbers of typical sequence that "survive" after the application of $\mathsf{Com}(I^i \Leftrightarrow \{x_{\text{old}}\}, K^i \Leftrightarrow \{x_{\text{old}}\}, \mathbf{A}^i)$. Clearly l_{i+1} and ϕ_{i+1} can be computed in $O(1)$ time.

Now, as $l_1 = (e_{\text{start}})$ and $\phi_1(1) = e_{\text{start}}$, we are able to construct an ordering $l = l_n$ such that $C(l) \in FS(|X|)$. Therefore we conclude to the following.

Theorem 1 *For all $k, l \geq 1$ there exists an algorithm that, given a graph G and a path decomposition $X = (X_i, 1 \leq i \leq |X|)$ of G with width at most l, computes whether the linear-width of G is at most k and, if so, constructs an edge ordering of G with linear-width at most k and that uses $O(V(G) + |X|)$ time.*

The results presented so far can be trivially extended to graphs with multiple edges. In such a case, we should consider the complexity of the algorithm in theorem 1 to be $O(|E(G)| + |X|)$.

4 The consequences of our algorithm

In this section we define several search game parameters and we present their relations with linear-width. Using these relations we conclude that there exist for any fixed k, linear time algorithms that check whether given graphs can be mixed, node, or edge searched with at most k searchers, and if so, output the corresponding search strategies.

4.1 Search games for an agile fugitive

In this section we give the definitions of three versions of search games on graphs and we present their connection with linear-width.

A *mixed searching game* is defined in terms of a graph representing a system of tunnels where an omniscient and agile fugitive with unbounded speed is hidden (alternatively, we can formulate the same problem considering that the tunnels are contaminated by some poisonous gas). The object of the game is to *clear* all edges, using one or more *searchers*. An edge of the graph is cleared if one of the following cases occur.

A: *both of its endpoints are occupied by a searcher,*

B: *a searcher slides along it*, i.e., a searcher is moved from one endpoint of the edge to the other endpoint.

A search is a sequence containing some of the following moves. **a**: place a new searcher on a vertex, **b**: remove a searcher from a vertex, **c**: slide a searcher, residing on some of the endpoints of an edge e, along e and place it on the other endpoint of e.

The object of a mixed search is to clear all edges using a search. The search number of a search is the maximum number of searchers on the graph during any move. The mixed search number, $\mathrm{ms}(G)$, of a graph G is the minimum search number over all the possible searches of it. A move causes *recontamination* of an edge if it causes the appearance of a path from an uncleared edge to this edge not containing any searchers on its vertices or its edges. (Recontaminated edges must be cleared again.) A search without recontamination is called *monotone*.

The *node (edge) search number*, $\mathrm{ns}(G)$ ($\mathrm{es}(G)$) is defined similarly to the mixed search number with the difference that an edge can be cleared only if **A** (**B**) happens.

The following results were proved by Bienstock and Seymour in [2] (see also [24]).

Theorem 2 *For any graph G the following hold:*
(a) *If* $\mathrm{ms}(G) \leq k$ *then there exist a monotone mixed search in G using at most k searchers.*
(b) $\text{linear-width}(G) \leq \mathrm{ms}(G)$.
(c) *If G does not contain vertices of degree 1, then* $\text{linear-width}(G) = \mathrm{ms}(G)$.
(d) *If G^e is the graph occurring from G after subdividing each of it edges, then* $\mathrm{es}(G) = \mathrm{ms}(G^e)$.
(e) *If G^n is the graph occurring if we replace every edge in G with two edges in parallel, then* $\mathrm{ns}(G) = \mathrm{ms}(G^n)$.

We mention that the mixed search number is equivalent with the parameter of proper-pathwidth defined by Takahashi, Ueno, and Kajitani in [24]. It is also known that the node search number is equal to the pathwidth, the interval thickness, and the vertex separation number (see [13, 14, 17, 12, 8]).

The following is a generalisation of Theorem 2.(c) and has been proved in [25].

Theorem 3 *If G^h is the graph occurring from G after subdividing each of its pendant edges, then* $\mathrm{ms}(G) = \text{linear-width}(G^h)$.

4.2 Final Comments

The result of Theorem 1 has several consequences. First, as one can find a path decomposition of a graph G with width at most l, if existing, in linear time ([3, 4], but see also below) and using Lemma 2 we can conclude to the following result.

Theorem 4 *For all $k \geq 1$, there exists an algorithm that, given a graph G, computes whether the linear-width of G is at most k and, if so, constructs an edge ordering of G with linear-width at most k and that uses $O(V(G))$ time.*

Using now Theorems 2 and 3, we can obtain the following result for the search parameters.

Theorem 5 *For all k, l, there exists an algorithm, that given a graph G and a path decomposition $X = (X_i, 1 \leq i \leq |X|)$ of G with width at most l, computes whether the mixed search*

number (edge search number; node search number) of G is at most k, and if so, constructs a mixed search (edge search; node search) that clears G with most k searchers, and that uses at most $O(|V(G)| + |X|)$ time.

Using small modifications of techniques from [3], the result above can be used to obtain an alternative (but strongly related) proof for the result from [3, 4] that for each fixed k, the problem to determine whether a given graph has pathwidth at most k, and if so, to find a path decomposition of width at most k, has a linear time algorithm. Using this fact, we obtain from Theorem 5 the following.

Theorem 6 *For all k, there exists an algorithm, that given a graph G, computes whether the mixed search number (edge search number; node search number) of G is at most k, and if so, constructs a monotone mixed search (edge search; node search) that clears G with most k searchers, and that uses at most $O(|V(G)|)$ time.*

References

[1] D. Bienstock. Graph searching, path-width, tree-width and related problems (a survey). *DIMACS Ser. in Discrete Mathematics and Theoretical Computer Science*, 5:33–49, 1991.

[2] D. Bienstock and P. Seymour. Monotonicity in graph searching. *J. Algorithms*, 12:239 – 245, 1991.

[3] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. *SIAM J. Comput.*, 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*, pages 627–637. Springer Verlag, Lecture Notes in Computer Science, vol. 1256, 1997.

[6] R. Breisch. An intuitive approach to speleotopology. *A publication of the Southwestern Region of the National Speleological Society*, VI:72–78, 1967.

[7] N. D. Dendris, L. M. Kirousis, and D. M. Thilikos. Fugitive-search games on graphs and related parameters. *Theor. Comp. Sc.*, 172:233–254, 1997.

[8] J. A. Ellis, I. H. Sudborough, and J. Turner. Graph separation and search number. Report DCS-66-IR, University of Victoria, 1987.

[9] J. A. Ellis, I. H. Sudborough, and J. Turner. The vertex separation and search number of a graph. *Information and Computation*, 113:50–79, 1994.

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

[11] H. Friedman, N. Robertson, and P. D. Seymour. The metamathematics of the graph minor theorem. *Contemporary Mathematics*, 65:229–261, 1987.

[12] N. G. Kinnersley. The vertex separation number of a graph equals its path width. *Inform. Proc. Letters*, 42:345–350, 1992.

[13] L. M. Kirousis and C. H. Papadimitriou. Interval graphs and searching. *Disc. Math.*, 55:181–184, 1985.

[14] L. M. Kirousis and C. H. Papadimitriou. Searching and pebbling. *Theor. Comp. Sc.*, 47:205–218, 1986.

[15] A. S. LaPaugh. Recontamination does not help to search a graph. *J. ACM*, 40:224–245, 1993.

[16] N. Megiddo, S. L. Hakimi, M. R. Garey, D. S. Johnson, and C. H. Papadimitriou. The complexity of searching a graph. *J. ACM*, 35:18–44, 1988.

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

[18] T. D. Parsons. Pursuit evasion in a graph. In Y. Alavi and D. R. Lick, editors, *Theory and Application of Graphs*, pages 426–441, Berlin, 1976. Springer Verlag.

[19] N. Robertson and P. D. Seymour. Graph width and well-quasi ordering: a survey. In J. A. Bondy and U. S. R. Murty, editors, *Progress in Graph Theory*, pages 399–406, Toronto, 1984. Academic Press.

[20] N. Robertson and P. D. Seymour. Disjoint paths – a survey. *SIAM J. Alg. Disc. Meth.*, 6:300–305, 1985.

[21] N. Robertson and P. D. Seymour. An outline of a disjoint paths algorithm. *Paths, Flows and VLSI Design, Algorithms and Combinatorics*, 9:267–292, 1990.

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

[23] P. D. Seymour and R. Thomas. Graph searching and a minimax theorem for tree-width. *J. Comb. Theory Series B*, 58:239–257, 1993.

[24] A. Takahashi, S. Ueno, and Y. Kajitani. Mixed-searching and proper-path-width. *Theor. Comp. Sc.*, 137:253–268, 1995.

[25] D. M. Thilikos. Algorithms and obstructions for linear-width and related search parameters. Technical Report UU-CS-97-35, Department of Computer Science, Utrecht University, Utrecht, 1997.

[26] R. Thomas. Tree-decompositions of graphs. Lecture notes, School of Mathematics. Georgia Institute of Technology, Atlanta, Georgia 30332, USA, 1996.