Contents lists available at ScienceDirect

Theoretical Computer Science

Finding small-width connected path decompositions in polynomial time

Dariusz Dereniowski [a,*], Dorota Osula [a,*], Paweł Rzążewski [b,*]

[a] *Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland*
[b] *Faculty of Mathematics and Information Science, Warsaw University of Technology, Warsaw, Poland*

A R T I C L E   I N F O

A B S T R A C T

A connected path decomposition of a simple graph G is a path decomposition $(X_1, \ldots, X_l)$ such that the subgraph of G induced by $X_1 \cup \cdots \cup X_i$ is connected for each $i \in \{1, \ldots, l\}$. The connected pathwidth of G is then the minimum width over all connected path decompositions of G. We prove that for each fixed k, the connected pathwidth of any input graph can be computed in polynomial-time. This answers an open question raised by Fedor V. Fomin during the GRASTA 2017 workshop, since connected pathwidth is equivalent to the connected (monotone) node search game.

© 2019 Published by Elsevier B.V.

1. Introduction

Since the famous 'graph minor' project by Robertson and Seymour that started with [35], the notions of treewidth and pathwidth received growing interest and a vast amount of results has been obtained. The pathwidth, informally speaking, allows us to say how closely an arbitrary graph resembles a path. This concept proved to be useful in designing algorithms for various graph problems, especially in the case when the pathwidth of an input graph is small (e.g. fixed), in which case quite often a variant of the well-known dynamic programming approach that progresses along a path decomposition of the input graph turns out to be successful.

Several modifications to pathwidth have been proposed and in this work we are interested in the connected variant in which one requires that a path decomposition $(X_1, \ldots, X_l)$ of a graph G satisfies: the vertices $X_1 \cup \cdots \cup X_i$ induce a connected subgraph in G for each $i \in \{1, \ldots, l\}$. This version of the classical pathwidth problem is motivated by several pursuit-evasion games, including, but not limited to, *edge search*, *node search* or *mixed search* [28,33,38]. More precisely, computing the minimum number of searchers needed to clean a given graph G in the node search game (i.e., computing the *node search number* of G) is equivalent to computing the connected pathwidth of G. Moreover, a connected path decomposition can be easily translated into the corresponding node search strategy that cleans G and vice versa. For further references that provide more details on correspondences between connected path decomposition and different variants of the search games see e.g. [2,16,24,23].

---

* Corresponding authors.
E-mail addresses: deren@eti.pg.edu.pl (D. Dereniowski), dorurban@student.pg.edu.pl (D. Osula), p.rzazewski@mini.pw.edu.pl (P. Rzążewski).

1.1. Related work

A lot of research has been done in the direction of obtaining FPT algorithms for pathwidth, parametrized by the pathwidth k. One of the first polynomial-time algorithms was presented in 1983 by Ellis et al. and had running time $O(n^{2k^2+4k+2})$ [18]. Later, these results were improved (e.g., [7,10]) leading to the currently fastest FPT algorithm, working in time $2^{O(k^2)}n$ [26]. In these algorithms, in order to produce an optimal solution, an approximate path- or tree-decomposition is pre-computed. It is thus of interest to have good approximations for these problems. Numerous works have been published in this direction [1,30,34], leading to the currently fastest algorithm for constant-factor approximation for treewidth, working in time $2^{O(k)}n$ [8], that is, single exponential in the treewidth k. The best known approximation ratio of a polynomial time approximation algorithm for pathwidth is $O(\sqrt{\log(opt)}\log n)$ [19].

There exist exact algorithms for computing pathwidth, whose running times are exponential in the order of the input graph. Pathwidth can be computed in $O^*(2^n)$-time (in $O^*(2^n)$ space) or in $O^*(4^n)$-time with the use of polynomial space, using a simple algorithm from [9]. There is also a faster algorithm with running time $O^*(1.9657^n)$ [37], which has been further improved very recently to $O^*(1.89^n)$ [29]. See [5,13,14] for some experimental approaches to pathwidth computation.

For pathwidth, it is known due to [36] that the set of minimal forbidden minors (i.e., the obstruction set) is finite for each fixed k. However, a significant difference between pathwidth and connected pathwidth is that the latter one is not closed under taking minors and hence it is not known if the set of minimal forbidden minors for connected pathwidth is finite [4].

We also point out that a number of results have been obtained for connected pathwidth or the closely related connected graph (edge) search, including algorithmic and computational ones [2,15,16,21,24,31], monotonicity [25,39], structural properties [3] or distributed algorithms [11,20,27].

1.2. Motivation

The connectivity constraint for pathwidth is natural and useful in graph searching games [2,20,24]. The connectivity is in some cases implied by potential applications (e.g., security constraints may enforce the clean, or safe, area to be connected) or it is a necessity, like in distributed or online versions of the problem [6,11,27,32].

Our second motivation comes from connections between pathwidth and connected pathwidth. More specifically, [16] implies that for any graph G, these parameters differ multiplicatively only by a small constant. This implies that an approximation algorithm for connected pathwidth immediately provides an approximation algorithm for pathwidth with asymptotically the same approximation ratio. This may potentially lead to obtaining better approximations for pathwidth since, informally speaking, the algorithmic search space for connected pathwidth is for some graphs much smaller than that for pathwidth. On the other hand, we do not know any algorithm computing the connected pathwidth in time $O^*((2-\epsilon)^n)$, for any $\epsilon > 0$. Thus, despite this smaller algorithmic search space, it is not clear how these two problems algorithmically differ in the context of designing exact algorithms.

During the GRASTA 2017 workshop, Fedor V. Fomin [22] raised an open question, whether we can verify in polynomial time, if the connected pathwidth of a given graph is at most k, for a fixed constant k. In this paper we answer this question in the affirmative.

1.3. Outline

In the next section we recall the definition of connected pathwidth and related terms used in this work. Section 3 provides a polynomial-time algorithm for determining whether the connected pathwidth of an arbitrary input graph G is at most k, where k is a fixed integer. The algorithm is inspired by the algorithms for computing minimum-length path decompositions by Dereniowski, Kubiak, and Zwols [17]. We also use the notation from this paper. Then, Section 4 contains the analysis of the algorithm (its correctness and running time). We finish with some open problems in Section 5.

2. Definitions

For a simple graph $G = (V(G), E(G))$ and a set $Y \subseteq V(G)$, the subgraph with vertex set Y and edge set $\{\{u, v\} \in E(G) \mid u, v \in Y\}$ is denoted by $G[Y]$ and is called the subgraph *induced* by Y. For $Y \subseteq V(G)$, we write $N_G(Y)$ to denote the *neighborhood* of Y in G, defined as $N_G(Y) = \{v \in V(G) \setminus Y \mid \exists u \in Y \text{ s.t. } \{u, v\} \in E(G)\}$.

Definition 1. A *path decomposition* of a simple graph $G = (V(G), E(G))$ is a sequence $\mathcal{P} = (X_1, \ldots, X_l)$, where $X_i \subseteq V(G)$ for each $i \in \{1, \ldots, l\}$, and

(i) $\bigcup_{i=1}^{l} X_i = V(G)$,
(ii) for each $\{u, v\} \in E(G)$ there exists $i \in \{1, \ldots, l\}$ such that $u, v \in X_i$,
(iii) for each i, j, k with $1 \le i \le j \le k \le l$ it holds that $X_i \cap X_k \subseteq X_j$.
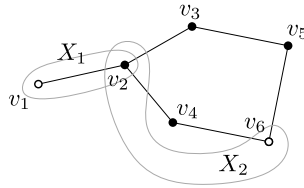
Fig. 1. An illustration of $\mathcal{I}$-connectivity; *empty circles* denote vertices from the set $\mathcal{I}$. A partial path decomposition $\mathcal{P} = (X_1, X_2)$ is $\mathcal{I}$-connected, as every connected subgraph of $G[X_1]$ and $G[X_1 \cup X_2]$ contains a vertex from $\mathcal{I}$ as required in Definition 2.

The *width* of a path decomposition $\mathcal{P}$ is width$(\mathcal{P}) = \max_{i \in \{1,\ldots,l\}} |X_i| - 1$. The *pathwidth* of G, denoted by pw(G), is the minimum width over all path decompositions of G.

We say that a path decomposition $\mathcal{P} = (X_1, \ldots, X_l)$ is *connected* if the subgraph $G[X_1 \cup \cdots \cup X_i]$ is connected for each $i \in \{1, \ldots, l\}$. The *connected pathwidth* of a graph G, denoted by cpw(G), is the minimum width taken over all connected path decompositions of G.

Finally, a *connected partial path decomposition* of a graph G is a connected path decomposition $(X_1, \ldots, X_i)$ of some subgraph H of G, where $N_G(V(G) \setminus V(H)) \subseteq X_i$. In other words, in the latter condition we require that each vertex of H that has a neighbor outside H belongs to the last bag X_i. Intuitively, a connected partial path decomposition of G can be potentially a prefix of some connected path decomposition of G. Also note that if $(X_1, \ldots, X_l)$ is a connected path decomposition, then for each $i \in \{1, \ldots, l\}$, the sequence $(X_1, \ldots, X_i)$ is a connected partial path decomposition of G, where $H = G[X_1 \cup \cdots \cup X_i]$.

In our analysis we use the following intermediate notion between arbitrary and connected path decompositions.

Definition 2. Given $\mathcal{I} \subseteq V(G)$, a (partial) path decomposition $\mathcal{P} = (X_1, \ldots, X_l)$ of G is $\mathcal{I}$-*connected* if for each $i \in \{1, \ldots, l\}$ each connected component H of $G[X_1 \cup \cdots \cup X_i]$ contains a vertex from $\mathcal{I}$ and this vertex belongs to the first bag in which H appears in $\mathcal{P}$.

In other words, if $\mathcal{P} = (X_1, \ldots, X_l)$ is a partial path decomposition or a prefix of one, then the subgraph of G induced by the union of bags of $\mathcal{P}$ may have several connected components and each of them must have a vertex in $\mathcal{I}$. Moreover, if one looks at the path decomposition of such a connected component H derived from $\mathcal{P}$, i.e. at the path decomposition obtained from $(X_1 \cap V(H), \ldots, X_l \cap V(H))$ by removing the empty bags, then such a decomposition starts with a bag containing a vertex in $\mathcal{I}$. Note that if $\mathcal{P}$ is connected then there is only one such connected component. See Fig. 1 for an illustration.

In our analysis, we will assume that the path decompositions we consider have the property that each bag introduces at most one new vertex, i.e., for $\mathcal{P} = (X_1, \ldots, X_l)$ it holds $|X_{i+1} \setminus X_1| \leq 1$ for each $i \in \{1, \ldots, l-1\}$. This property can be easily reestablished whenever needed (a folklore).

3. The algorithm

We will present an algorithm that decides whether, for a connected input graph G and a set $\mathcal{I}$, there exists an $\mathcal{I}$-connected path decomposition of width at most $k - 1$ (thus k is the maximum bag size of the connected path decomposition to be computed). In the following, G, $\mathcal{I}$ and k are hence fixed. Note that we may without loss of generality assume that the first bag in a connected path decomposition to be computed has only one vertex.

We start by informally sketching the high-level idea of the algorithm. Then, Sections 3.1 and 3.2 give the algorithm and Section 3.3 gives a summary of our method. We use a dynamic programming approach. To that end we introduce a set of states, so that each state encodes some partial path decomposition of G. If a transition from one state to another is possible, then this certifies that the partial path decomposition corresponding to one state can be extended to (i.e., is a prefix of) a partial path decomposition corresponding to the other state. We will verify whether the transition is possible using a recursive procedure. The need to ensure consistency of solutions found in recursive calls is the reason why we consider a more general problem of finding $\mathcal{I}$-connected path decompositions.

Finally, we will show that each transition can be checked in polynomial time and that it is enough to consider only polynomially many states. This eventually leads to an algorithm with a desired complexity.

For any $S \subseteq V(G)$, we say that a subgraph H of G is an S-*branch* if H is a connected component of $G - S$ and $N_G(V(H)) = S$. For any $S \subseteq V(G)$, define $\mathcal{B}(S)$ to be the set of all S-branches. A set S is called a *bottleneck* if the number of S-branches is at least $2k + 1$, as it guarantees us the existence of at least one special branch called an *in-branch*, which will be defined formally in the next section. Observe that each connected component of $G - X$, for any $X \subseteq V(G)$, is an S-branch for exactly one non-empty subset S of X.

Let us mention that S-branches are also known as *full components associated with S* (see e.g. Bouchitté and Todinca [12]).

3.1. States

By a *potential state* we mean a triple $\left(X, \{B_S\}_{S \subseteq X}, \{f^{B_S}\}_{S \subseteq X}\right)$, consisting of:

- a non-empty set $X \subseteq V(G)$ with $|X| \leq k$,
- a subset $B_S \subseteq \mathcal{B}(S)$ of cardinality at most $2k$, chosen for every non-empty $S \subseteq X$,
- a function $f^{B_S} : \mathcal{B}(S) \to \{0, 1\}$, chosen for every non-empty $S \subseteq X$. We additionally require that if $H, H' \in \mathcal{B}(S) \setminus B_S$, then $f^{B_S}(H) = f^{B_S}(H')$.

The exact meaning of B_S will be explained later on, but let us present some intuition. In Lemma 1 we show that for every path decomposition $\mathcal{P}$, the vertices of all but at most $2k$ S-branches appear in bags of $\mathcal{P}$ in a certain, well-structured way. The set B_S and the function f^{B_S} will be used to describe the structure of the remaining, badly behaving S-branches.

Observe that the set X may be chosen in at most n^k ways and the number of choices of S is at most 2^k. For every S, the number of S-branches is at most n, so B_S can be chosen in at most n^{2k} ways. The function f^{B_S} can be chosen in at most $2^{|B_S|} \cdot 2 \leq 2^{2k+1}$ ways. Therefore, the number of potential states is at most $n^k \cdot 2^k \cdot n^{2k} \cdot 2^{2k+1} = O(n^{3k})$, i.e., polynomial in n, where the asymptotic notation hides the factor that depends on k.

With a potential state $s = (X, \{B_S\}_{S \subseteq X}, \{f^{B_S}\}_{S \subseteq X})$ we associate the following notions. By bag(s) we denote the set X. By cover(s) we denote the set of vertices

$$ X \cup \bigcup_{S \subseteq X, S \neq \emptyset} \left(\bigcup_{H \in \mathcal{B}(S) : \, f^{B_S}(H) = 1} V(H) \right). $$

By G_s we denote the subgraph of G induced by cover(s). We say that two states w, s are *indistinguishable* if cover$(w) =$ cover(s) and bag$(w) =$ bag(s). Otherwise the states are *distinguishable*. In particular, indistinguishable states can only differ by the choice of B_S. We note that for such two distinguishable states it may hold that cover$(w) =$ cover(s) but bag$(w) \neq$ bag(s).

Example. In the example in Fig. 1 we can consider two states: s and w with bags bag$(s) = X_1$ and bag$(w) = X_2$. There exists one $\{v_1\}$-branch $H_1 = G - \{v_1\}$, two $\{v_2\}$-branches: $H_2 = G[\{v_1\}]$ and $H_3 = G[\{v_3, v_4, v_5, v_6\}]$, one $\{v_6\}$-branch $H_4 = G - \{v_6\}$ and two $\{v_2, v_6\}$-branches: $H_5 = G[\{v_4\}]$ and $H_6 = G[\{v_3, v_5\}]$. We notice that the set of $\{v_1, v_2\}$-branches is empty. The state s is equal to $(X_1, \{\{H_1\}, \{H_2, H_3\}, \emptyset\}, \{f_1, f_2, f_\emptyset\})$ and the state w is $(X_2, \{\{H_2, H_3\}, \{H_4\}, \{H_5, H_6\}\}, \{f_2, f_3, f_4\})$, such that $f_\emptyset$ is a function with empty domain, $f_1(H_1) = f_2(H_3) = f_3(H_4) = f_4(H_5) = f_4(H_6) = 0$ and $f_2(H_2) = 1$. We notice that cover$(s) = \{v_1, v_2\}$ and cover$(w) = \{v_1, v_2, v_6\}$.

Let v be a vertex from cover(s) which has a neighbor $u \notin$ cover(s). We argue that $v \in$ bag(s). Otherwise, if $v \notin$ bag(s), then both v and u belong to the same S-branch for some $S \subseteq$ bag(s). Thus, they are either both in cover(s), or outside it. From this it follows that every vertex $v \in$ cover(s), which has a neighbor $u \notin$ cover(s), must belong to bag(s). Let us denote the set of such vertices v by border(s). We have proved:

Observation 1. For each potential state s it holds that border$(s) \subseteq$ bag(s).

We say that a potential state s is a *state* if each connected component of G_s contains a vertex in $\mathcal{I}$.

We introduce a Boolean table Tab, indexed by all states. For a state s, the value of $Tab[s]$ will be set to *true* by our algorithm if and only if there exists some $\mathcal{I}$-connected partial path decomposition $\mathcal{P} = (X_1, X_2, \ldots, X_l)$ of G with $H = G_s$, such that width$(\mathcal{P}) \leq k - 1$ and $X_l =$ bag(s). We will use a dynamic programming to fill the table Tab. Then, we will conclude that G has an $\mathcal{I}$-connected path decomposition of width at most $k - 1$ if and only if $Tab[s] = true$ for some state s with cover$(s) = V(G)$.

Observe that such a final state exists, since for $s = (X, \{B_S\}_S, \{f^{B_S}\}_S)$, we have cover$(s) = V(G)$ if and only if $f^{B_S}(H) = 1$ for every S and H. However, the astute reader may notice that in our representation we might have not included some $\mathcal{I}$-connected partial path decompositions and it could be possible that we do not find a solution, even though it exists. We will show that if cpw$(G) \leq k - 1$, then there exists a special type of an $\mathcal{I}$-connected path decomposition of width at most $k - 1$, called a *structured path decomposition* (defined later), which can be found using our algorithm because, as we will argue, our table Tab does not 'omit' any structured path decompositions.

3.2. Extension rules

Let us introduce a total ordering $\prec$ on the set of states. We say that $w \prec s$ if $|$cover$(w)| < |$cover$(s)|$, or $|$cover$(w)| = |$cover$(s)|$ and $|$bag$(w)| > |$bag$(s)|$. If $|$cover$(w)| = |$cover$(s)|$ and $|$bag$(w)| = |$bag$(s)|$, then we resolve such a tie arbitrarily.

We initialize Tab by setting $Tab[s] = true$ for every state s, such that cover$(s) =$ bag$(s) = \{v\}$, for some $v \in \mathcal{I}$, while for the remaining states s we initialize $Tab[s]$ to be *false*. In particular for each s with $|$cover$(s)| = 1$ and cover$(s) =$ bag$(s) \in \mathcal{I}$,

we have $Tab[s] = true$. In our dynamic programming algorithm, we process states according to the ordering $\prec$ and fill the table Tab using two extension rules: *step extension* and *jump extension*.

Step extension   for distinguishable states w, s with $w \prec s$ and $|\text{cover}(s)| > 1$: if $Tab[w] = true$ and
        (S1) each connected component of $G[\text{bag}(s)]$ contains a vertex from $\text{bag}(w) \cup \mathcal{I}$,
        (S2) $\text{border}(w) \subseteq \text{bag}(s)$,
        (S3) $\text{cover}(s) = \text{cover}(w) \cup \text{bag}(s)$,
        (S4) $\text{bag}(s) \cap \text{cover}(w) \subseteq \text{bag}(w)$,
        then set $Tab[s]$ to $true$.

Jump extension   for distinguishable states $w = (X, \{B_S\}_{S \subseteq X}, \{f^{B_S}\}_S)$ and $s = (X, \{B_S\}_{S \subseteq X}, \{g^{B_S}\}_S)$ with $w \prec s$ and
        $|\text{cover}(s)| > 1$: if $Tab[w] = true$ and there exists a bottleneck set $S' \subseteq X$, such that
        (J1) $f^{B_{S'}}(H) = 0$ and $g^{B_{S'}}(H) = 1$ for every $H \in \mathcal{B}(S') \setminus B_{S'}$,
        (J2) $f^{B_{S'}}(H) = g^{B_{S'}}(H)$ for every $H \in B_{S'}$,
        (J3) $f^{B_S}(H) = g^{B_S}(H)$ for every non-empty $S \neq S'$, $S \subseteq X$, and $H \in \mathcal{B}(S)$,
        (J4) for each $H \in \mathcal{B}(S') \setminus B_{S'}$ there exists an $\big((N_G(S') \cap V(H)) \cup \mathcal{I} \big)$-connected path decomposition $\mathcal{P}_H$ of H of
            width at most $k - |X| - 1$,
        then set $Tab[s]$ to $true$.

Let us present some intuitions behind these extension rules. In step extension, if w corresponds to some $\mathcal{I}$-connected partial path decomposition $\mathcal{P} = (X_1, X_2, \ldots, X_l = \text{bag}(w))$, then s corresponds to an $\mathcal{I}$-connected partial path decomposition $\mathcal{P}' = (X_1, X_2, \ldots, X_l, X_{l+1} = \text{bag}(s))$ (we extend $\mathcal{P}$ by adding a single bag, namely $\text{bag}(s)$). Also note, that each new vertex in X_{l+1}, i.e., one that is not in X_l, is (due to (S1)) connected by a path consisting of vertices from X_{l+1} to a vertex in X_l or $\mathcal{I}$, as required in $\mathcal{I}$-connected path decompositions.

In jump extension, if the state w corresponds to some $\mathcal{I}$-connected partial path decomposition $\mathcal{P} = (X_1, X_2, \ldots, X_l)$, then the state s corresponds to an $\mathcal{I}$-connected partial path decomposition $\mathcal{P}' = (X_1, X_2, \ldots, X_l, X_{l+1}, X_{l+2}, \ldots, X_{l+l'})$, where

- $\left(\bigcap_{i=l}^{l+l'} X_i \right) = X_l = X_{l+l'}$,
- $\mathcal{P}'' := (X_{l+1} \setminus X_l, X_{l+2} \setminus X_l, \ldots, X_{l+l'} \setminus X_l)$ is a (not necessarily connected) path decomposition of the graph induced by some S'-branches, for some $S' \subseteq X_l$. These are the S'-branches H in (J1) and $\mathcal{P}''$ is obtained by 'concatenating' the path decompositions from (J4).

Note that although a path decomposition $\mathcal{P}_H$ in (J4) may not be connected, we ensure (by definition of $\mathcal{I}$-connectivity) that each connected component of the subgraph induced by each prefix of $\mathcal{P}_H$ has a vertex from $N_G(S') \cap V(H)$ or from $\mathcal{I}$. Hence it contains a neighbor of S' or a vertex from $\mathcal{I}$, ensuring the required $\mathcal{I}$-connectivity of the resulting path decomposition. Fig. 1 presents an example of step extension from state s to state w, i.e., these states fulfill the conditions of the step extension.

3.3. Summing up

Let us recall how the algorithm works. We introduce a Boolean table Tab, indexed by all states. We initialize Tab by setting $Tab[s] = true$ for every state s, such that $\text{cover}(s) = \text{bag}(s) = \{v\}$, for some $v \in \mathcal{I}$, while for the remaining states s we initialize $Tab[s]$ to be $false$.

Then we process the states with respect to the ordering $\prec$, checking whether a step extension or a jump extension can be applied to set $Tab[s] = true$. We terminate when we find a state corresponding to a feasible solution (i.e., when we set $Tab[s] = true$ for some state s with $\text{cover}(s) = V(G)$), or when we have processed all states. In the latter case we report that a solution does not exist.

We note that the algorithm will be subsequently switching some entries of Tab from $false$ to $true$, and hence until the completion of the algorithm it is understood that the value $false$ of a particular entry of Tab does not provide any information as to whether a corresponding path decomposition exists.

4. The analysis

Let us start by introducing some more definitions and additional notation. Let $\mathcal{P} = (X_1, X_2, \ldots, X_l)$ be a path decomposition of G. We say that a connected subgraph H of G is *contained in* an interval $[i, j]$ of $\mathcal{P}$ for some $1 \leq i \leq j \leq l$, if $V(H) \cap X_t \neq \emptyset$ if and only if $t \in \{i, \ldots, j\}$. Note that this definition is valid since it follows that for a connected subgraph H, the subset of indices t such that $V(H) \cap X_t \neq \emptyset$ is indeed an interval. If H is contained in $[i, j]$, then we denote these endpoints of the interval as $i = \alpha(H, \mathcal{P})$ and $j = \beta(H, \mathcal{P})$. If $\mathcal{P}$ is clear from the context, we will often write shortly $\alpha(H) := \alpha(H, \mathcal{P})$ and $\beta(H) := \beta(H, \mathcal{P})$.

For a set S and an $\mathcal{I}$-connected path decomposition $\mathcal{P} = (X_1, X_2, \ldots, X_l)$, we say that an S-branch H is an *in-branch* if $S \subseteq X_{\alpha(H)}$ and $S \subseteq X_{\beta(H)}$. The lemma below gives us a lower bound on the number of in-branches of S.
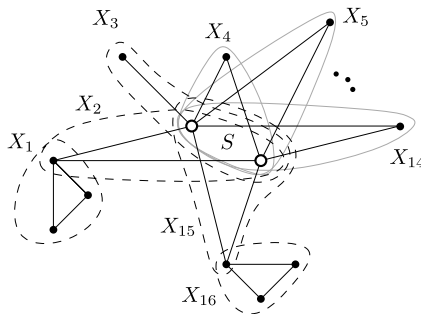
Fig. 2. An illustration of a two-vertex bottleneck set S and the corresponding S-branches.

Lemma 1. *For every set S and a path decomposition $\mathcal{P} = (X_1, X_2, \ldots, X_l)$, at most $2k$ S-branches are not in-branches.*

Proof. Consider an S-branch H, which is not an in-branch. This means that $S \nsubseteq X_{\alpha(H)}$ or $S \nsubseteq X_{\beta(H)}$.

First, consider H, such that $S \nsubseteq X_{\alpha(H)}$. Let t be the minimum index such that $S \subseteq X_1 \cup X_2 \cup \cdots \cup X_t$. Let $v \in S$ be a vertex in $X_t \setminus (X_1 \cup X_2 \cup \cdots \cup X_{t-1})$, it exists by the definition of t. Recall that v is a neighbor of some vertex w of H, so, since $\mathcal{P}$ is a path decomposition, there must be a bag X_i containing both v and w. Since X_t is the first bag, where v appears, we observe that $i \geq t$ and thus $\beta(H) \geq t$.

Suppose now that $\alpha(H) > t$. Note that since $S \nsubseteq X_{\alpha(H)}$, there is some $u \in S \setminus X_{\alpha(H)}$. Recall that u is a neighbor of some vertex from H. However, since $u \in X_1 \cup X_2 \cup \cdots \cup X_t$ and $u \notin X_{\alpha(H)}$, the vertex u does not appear in any bag containing a vertex of H, so $\mathcal{P}$ cannot be a path decomposition of G. Thus $\alpha(H) \leq t$.

Therefore, the bag X_t contains at least one vertex from H. Since S-branches are vertex-disjoint and $|X_t| \leq k$, we observe that there are at most k S-branches H, such that $S \nsubseteq X_{\alpha(H)}$.

Now consider an S-branch H, such that $S \subseteq X_{\alpha(H)}$ and $S \nsubseteq X_{\beta(H)}$. Let t' be the maximum index such that $S \subseteq X_{t'} \cup \cdots \cup X_l$.

Analogously to the previous case, we observe that $\alpha(H) \leq t'$ (by the maximality of t') and $\beta(H) \geq t'$, because $S \nsubseteq X_{\beta(H)}$. Thus the bag $X_{t'}$ contains at least one vertex from H, which shows that the number of S-branches H, such that $S \subseteq X_{\alpha(H)}$ and $S \nsubseteq X_{\beta(H)}$, is at most k. Therefore the total number of S-branches, that are not in-branches is at most $2k$, which completes the proof. □

Recall that a non-empty set S is a bottleneck if $|\mathcal{B}(S)| > 2k$. Thus Lemma 1 implies the following:

Corollary 1. *If $\mathcal{P} = (X_1, \ldots, X_l)$ is a path decomposition and S is a bottleneck, then the following properties hold:*

1. *S has at least one in-branch,*
2. *there is i, such that $S \subseteq X_i$,*
3. *$|S| \leq k$.* □

These properties justify the following definition.

Definition 3. For a bottleneck $S \subseteq V(G)$, let $t_1(S, \mathcal{P})$ (respectively $t_2(S, \mathcal{P})$) be the minimum (respectively maximum) index i such that $i = \alpha(H, \mathcal{P})$ ($i = \beta(H, \mathcal{P})$, respectively) for some S-branch H (respectively H), which is an in-branch.

Note that the definition of an in-branch implies that $S \subseteq X_{t_1(S, \mathcal{P})} \cap X_{t_2(S, \mathcal{P})}$. The interval $I(S, \mathcal{P}) = [t_1(S, \mathcal{P}), t_2(S, \mathcal{P})]$ is called the *interval of S*. Again, we will often write shortly $t_1(S)$, $t_2(S)$, and $I(S)$, if $\mathcal{P}$ is clear from the context.

For a bottleneck S we can refine the classification of S-branches, which are not in-branches. We say that an S-branch H, which is not an in-branch, is

- a *pre-branch* if $\alpha(H) < t_1(S)$,
- a *post-branch* if $\alpha(H) \geq t_1(S)$ and $\beta(H) > t_2(S)$.

By $\mathcal{B}^x(S, \mathcal{P})$ we denote the set of all x-branches for S, where $x \in \{\text{pre, in, post}\}$. Again, if $\mathcal{P}$ is clear from the context, we will write $\mathcal{B}^x(S)$ instead of $\mathcal{B}^x(S, \mathcal{P})$. By $\mathcal{C}(S)$ we denote the set of all connected components of $G - S$, that are not S-branches.

Example. The graph G in Fig. 2 illustrates the above concepts. The sequence $X_1, X_2, \ldots, X_{16}$ forms a connected path decomposition of G. The only bottleneck set S consists of two vertices denoted by circles. (In this example we take $k = 3$.) There are 13 S-branches: one pre-branch ($G[X_1 \cup X_2 \setminus S]$), eleven in-branches ($G[X_i \setminus S]$ for $i \in \{4, 5, \ldots, 14\}$), and one post-branch

$(G[(X_{15} \cup X_{16}) \setminus S])$. The interval of S is equal to $I(S, \mathcal{P}) = [4, 14]$ and the component $G[X_3 \setminus S]$ is not an S-branch, because some vertices in S do not have a neighbor in this component.

For a subgraph H of G, we say that H *waits in* an interval $[i, j]$ of $\mathcal{P}$ if

$$V(H) \cap X_i = V(H) \cap X_{i+1} = \cdots = V(H) \cap X_j.$$

We say that a path decomposition $\mathcal{P}$ is *S-structured* if each subgraph C that is in $\mathcal{C}(S)$ or is a post- or a pre-branch of $\mathcal{S}$ waits in $I(S, \mathcal{P})$. The main technical tool in our approach is the following result concerning the structure of $\mathcal{I}$-connected path decompositions. The proof of the lemma is provided after giving several technical facts that we need.

Lemma 2. *If there exists an $\mathcal{I}$-connected path decomposition $\mathcal{P}$, then there is also an $\mathcal{I}$-connected path decomposition $\mathcal{P}'$ of width at most* $\text{width}(\mathcal{P})$ *such that $\mathcal{P}'$ is S-structured for every bottleneck S.*

We define $c_{\min}(S, \mathcal{P})$ as the minimum index $i \in I(S, \mathcal{P})$, for which the size of the set

$$X_i \cap \left(\bigcup_{H \in \mathcal{B}^{\text{pre}}(S) \cup \mathcal{B}^{\text{post}}(S) \cup \mathcal{C}(S)} V(H) \right)$$

is minimum. Also, set $X^* := \left(\bigcup_{H \in \mathcal{B}^{\text{pre}}(S) \cup \mathcal{B}^{\text{post}}(S) \cup \mathcal{C}(S)} V(H) \right) \cap X_{c_{\min}(S, \mathcal{P})}$ to be this minimum-size set.

Let Γ be the set of all $\mathcal{I}$-connected path decompositions of G and $\mathcal{S}$ be the set of all bottlenecks of G. We will define a function $F : \mathcal{S} \times \Gamma \to \Gamma$, which for given $S \in \mathcal{S}$ and $\mathcal{P} \in \Gamma$ transforms $\mathcal{P}$ into an S-structured $\mathcal{I}$-connected path decomposition of width at most $\text{width}(\mathcal{P})$. For simplicity of notation, from now on $t_1 = t_1(S, \mathcal{P})$, $t_2 = t_2(S, \mathcal{P})$, and $c_{\min} = c_{\min}(S, \mathcal{P})$, whenever S and $\mathcal{P}$ are clear from the context. Let $\mathcal{B}^{\text{in}}(S, \mathcal{P}) = \{H_1, \ldots, H_{l_{\text{in}}}\}$, where the in-branches are ordered according to their first occurrences in $\mathcal{P}$, that is, $\alpha(H_1) \leq \cdots \leq \alpha(H_{l_{\text{in}}})$. Let

$$d = \sum_{H \in \mathcal{B}^{\text{in}}(S)} (\beta(H) - \alpha(H) + 1).$$

In other words, d is the sum of lengths of intervals in which the in-branches H are contained in the path decomposition $\mathcal{P} = (X_1, X_2, \ldots, X_l)$.

For each in-branch H_i, $i \in \{1, \ldots, l_{\text{in}}\}$, we define the following sequence:

$$\mathcal{P}_i := \left(X_{\alpha(H_i)} \cap V(H_i), \ldots, X_{\beta(H_i)} \cap V(H_i) \right).$$

Since $\mathcal{P}$ is an $\mathcal{I}$-connected path decomposition of G, it is straightforward to observe that $\mathcal{P}_i$ is a $(N_G(S) \cap V(H_i)) \cup \mathcal{I}$-connected path decomposition of H_i. We will denote the elements of $\mathcal{P}_i$ by

$$\mathcal{P}_i = \left(X_1^i, \ldots, X_{\beta(H_i) - \alpha(H_i) + 1}^i \right).$$

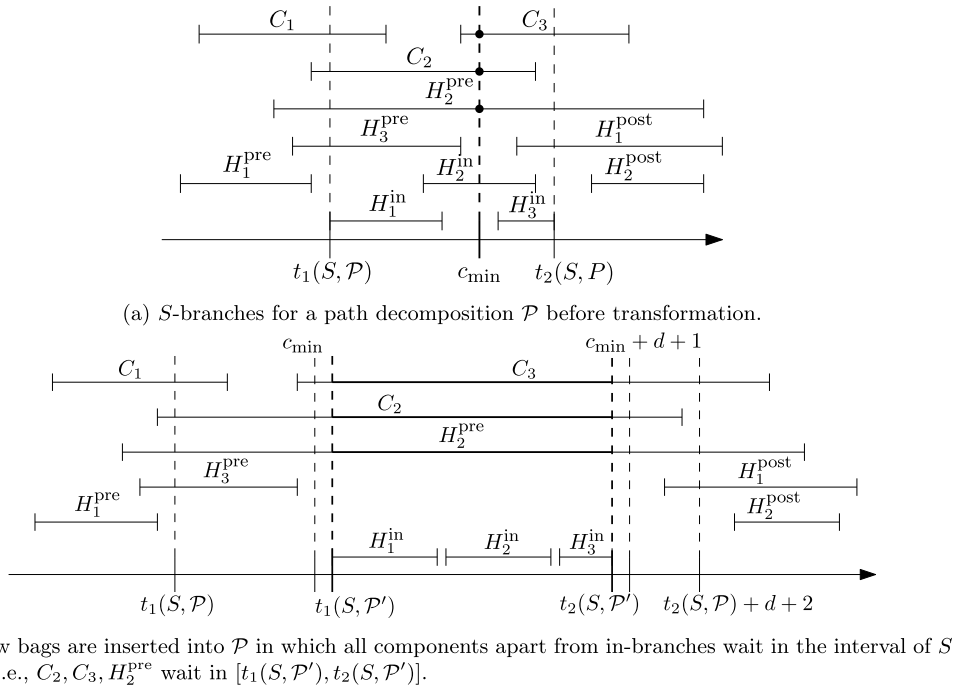Then, let us define a sequence $\mathcal{P}^*$ as follows:

$$\mathcal{P}^* := \bigcirc_{i=1}^{l_{\text{in}}} \mathcal{P}_i,$$

where $\bigcirc$ denotes the concatenation of sequences. Observe that the length of $\mathcal{P}^*$ is exactly d. We will denote the elements of $\mathcal{P}^*$ by $\mathcal{P}^* = \left(X_1^*, \ldots, X_d^* \right)$.

Define B^{in} to be the set of vertices of the in-branches of S, i.e., $B^{\text{in}} := \bigcup_{H \in \mathcal{B}^{\text{in}}(S, \mathcal{P})} V(H)$. Now, we define a path decomposition $F(S, \mathcal{P}) = \left(X_1', \ldots, X_{l+d+1}' \right)$ as follows (see Fig. 3):

$$X_i' = \begin{cases} X_i & \text{for } i \in \{1, \ldots, t_1 - 1\}; & \text{(a)} \\ X_i \setminus B^{\text{in}} & \text{for } i \in \{t_1, \ldots, c_{\min} - 1\}; & \text{(b)} \\ X_{c_{\min}} \setminus B^{\text{in}} & \text{for } i = c_{\min}; & \text{(c)} \\ X^* \cup X_{i - c_{\min}}^* \cup S & \text{for } i \in \{c_{\min} + 1, \ldots, c_{\min} + d\}; & \text{(d)} \\ X_{c_{\min}} \setminus B^{\text{in}} & \text{for } i = c_{\min} + d + 1; & \text{(e)} \\ X_{i - d - 1} \setminus B^{\text{in}} & \text{for } i \in \{c_{\min} + d + 2, \ldots, t_2 + d + 1\}; & \text{(f)} \\ X_{i - d - 1} & i \in \{t_2 + d + 2, \ldots, l + d + 1\}. & \text{(g)} \end{cases} \qquad (1)$$

Observe that $F(S, \mathcal{P})$ is obtained from $\mathcal{P}$ by a modification of the interval $[t_1, t_2]$ of $\mathcal{P}$. The prefix $(X_1, X_2, \ldots, X_{t_1-1})$ and the suffix $(X_{t_2+1}, X_{t_2+2}, \ldots, X_l)$ of $\mathcal{P}$ are just copied into $F(S, \mathcal{P})$ without any changes (see conditions (1)(a) and (1)(g)). All

(a) S-branches for a path decomposition $\mathcal{P}$ before transformation.

(b) New bags are inserted into $\mathcal{P}$ in which all components apart from in-branches wait in the interval of S in $\mathcal{P}'$, i.e., $C_2, C_3, H_2^{\text{pre}}$ wait in $[t_1(S, \mathcal{P}'), t_2(S, \mathcal{P}')]$.

Fig. 3. Illustration of the conversion from $\mathcal{P}$ to $\mathcal{P}' = F(S, \mathcal{P})$ for a bottleneck S. For simplification, S is assumed to only have 8 S-branches and 3 components, which are denoted by letters H and C, respectively, with appropriate indices.

components of $G - S$, apart from the in-branches, are covered by bags on positions up to $c_{\min}$, see (1)(b) and (1)(c), and after position $c_{\min} + d$, see (1)(f), and they wait for $d + 1$ steps in the interval $[c_{\min} + 1, c_{\min} + d + 1]$ of $F(S, \mathcal{P})$ — see (1)(d)–(1)(e). The interval $[c_{\min} + 1, c_{\min} + d]$ is used in (1)(d) to cover all in-branches, one by one, in the order of their appearance in $\mathcal{P}$. Intuitively, two bags without any vertices of in-branches are present on positions $c_{\min}$ and $c_{\min} + d + 1$ to ensure that for any other bottleneck S', such that $I(S, \mathcal{P}) \subsetneq I(S', \mathcal{P})$, we have that $c_{\min}(S', \mathcal{P}) \notin I(S, \mathcal{P})$, see (1)(c) and (1)(e). (The appropriate details are in the proofs.) Fig. 3 illustrates the conversion from $\mathcal{P}$ to $\mathcal{P}' := F(S, \mathcal{P})$ for a bottleneck S. Notice that the interval of S in $\mathcal{P}'$ is given by $t_1(S, \mathcal{P}') = c_{\min} + 1$ and $t_2(S, \mathcal{P}') = c_{\min} + d$, and it is possibly different than $[t_1, t_2]$. In the next lemma we show that $F(S, \mathcal{P})$ has all necessary properties.

Lemma 3. *For any $\mathcal{I}$-connected path decomposition $\mathcal{P}$ and a bottleneck S, $\mathcal{P}' = F(S, \mathcal{P})$ is an $\mathcal{I}$-connected path decomposition with* width$(\mathcal{P}') \leq$ width$(\mathcal{P})$.

Proof. Let us recall the notation: $\mathcal{P} := (X_1, \ldots, X_l)$, $\mathcal{P}' = F(S, \mathcal{P}) = (X_1', \ldots, X_{l'}')$, $t_1 = t_1(S, \mathcal{P})$ and $t_2 = t_2(S, \mathcal{P})$, $B^{\text{in}} := \bigcup_{H \in \mathcal{B}^{\text{in}}(S, \mathcal{P})} V(H)$. Moreover, define

$$B^{\text{not-in}} := \left(\bigcup_{H \in \mathcal{B}^{\text{pre}}(S, \mathcal{P}) \cup \mathcal{B}^{\text{post}}(S, \mathcal{P}) \cup \mathcal{C}(S)} V(H) \right).$$

Also, recall that $X_i' = X_i$, $1 \leq i < t_1$ and $X_i' = X_{i-d-1}$, for all $t_2 + d + 2 \leq i \leq l'$.

First, we want to show that $\mathcal{P}'$ satisfies the conditions in Definition 1.

Let $\{u, v\}$ be an edge of G. Since $\mathcal{P}$ is a path decomposition, $u, v \in X_i$ for some i. If $i < t_1$, then $u, v \in X_i' = X_i$. If $i > t_2$, then $u, v \in X_{i+d+1}' = X_i$.

So suppose that $u, v \in X_i$ for $i \in [t_1, t_2]$. Note that this means that $u, v \in B^{\text{in}} \cup S$ or $u, v \in B^{\text{not-in}} \cup S$. If $u, v \in B^{\text{not-in}} \cup S$, then we have

(i) $u, v \in X_i'$, if $i \leq c_{\min}$;
(ii) $u, v \in X_{i+d+1}'$, otherwise.

Finally, consider the case that, say, $u \in B^{\text{in}}$ and $v \in B^{\text{in}} \cup S$ (if both $u, v \in S$, then we are at the previous case). This means that u is a vertex of some in-branch $H_s \in \mathcal{B}^{\text{in}}(S, \mathcal{P})$, so it appears in some bag of $\mathcal{P}_s$ and thus of $\mathcal{P}^*$. This implies that $u, v \in X_j'$ for some $j \in [c_{\min} + 1, c_{\min} + d]$. This implies that $\mathcal{P}'$ satisfies conditions (i) and (ii) from Definition 1.

Now let us verify that the condition (iii) is also satisfied, i.e., for every vertex v and indices $i < s < j$, such that $v \in X'_i \cap X'_j$ we have $v \in X'_s$. Clearly the condition is satisfied for every v and $j < t_1$ or $i > t_2$, since these parts of $\mathcal{P}'$ are just copied from $\mathcal{P}$ without any modifications. The situation is very similar if $i \leq t_2$ and $j \geq t_1$ and $v \in (S \cup B^{\text{not-in}}) \setminus X^*$. If $v \in X^*$, then v is included in all bags X'_s for $s \in [c_{\min}, c_{\min} + d + 1]$, so the condition (iii) follows from the correctness of $\mathcal{P}$. Finally, if v is a vertex of some $H_a \in \mathcal{B}^{\text{in}}(S, \mathcal{P})$, then the condition (iii) follows from the property (iii) that holds for the path decomposition P_a.

We now show that $\text{width}(\mathcal{P}') \leq \text{width}(\mathcal{P})$. Let $c_1 = |X^*|$ and $c_2 = \max_{i=t_1,\ldots,t_2} |B^{\text{in}} \cap X_i|$, and let $k = \text{width}(\mathcal{P}) + 1 = \max_{i=1,\ldots,l} |X_i|$. Observe that each X'_i for $i \notin \{t_1, \ldots, t_2 + d + 1\}$ is an exact copy of some X_j, so $|X'_i| \leq k$. Moreover, each X'_i for $i \in \{t_1, \ldots, c_{\min}\} \cup \{c_{\min} + d + 1, \ldots, t_2 + d + 1\}$ was obtained from some X_j by removal of vertices of B^{in}, so again we have $|X_i| \leq k$. Finally, for $i \in \{c_{\min} + 1, \ldots, c_{\min} + d\}$ we have

$$|X'_i| = |S| + |X^*| + |X^*_{i-c_{\min}}| \leq |S| + c_1 + c_2.$$

However, by the definition of $\mathcal{P}^*$, we observe that $|S| + c_1 + c_2 \leq |X_j|$ for some $j \in \{t_1, \ldots, t_2\}$. Therefore, $\text{width}(\mathcal{P}') \leq \text{width}(\mathcal{P})$.

Finally, recall that $\mathcal{P}$ is $\mathcal{I}$-connected. Consider any connected component H of the subgraph $G[X'_1 \cup \cdots \cup X'_i]$ for an $i \in \{1, \ldots, l'\}$. We argue that H has a vertex in $X'_{\alpha(H, \mathcal{P}')} \cap \mathcal{I}$ as required in an $\mathcal{I}$-connected path decomposition. Denote $j' = \alpha(H, \mathcal{P}')$. We consider a few cases following the definition of $\mathcal{P}'$ in (1)(a)-(1)(g).

Suppose first that $j' < t_1$ or $j' \geq t_2 + d + 2$. Denote $j = j'$ when $j' < t_1$ and $j = j' - d - 1$ when $j' \geq t_2 + d + 2$. Then, $X'_1 \cup \cdots \cup X'_{j'} = X_1 \cup \cdots \cup X_j$ and $X'_{j'} = X_j$. Thus, for such j',

$$X'_{\alpha(H, \mathcal{P}')} \cap V(H) = X'_{j'} \cap V(H) = X_j \cap V(H) = X_{\alpha(H, \mathcal{P})} \cap V(H).$$

Since $\mathcal{P}$ is $\mathcal{I}$-connected, $X_{\alpha(H, \mathcal{P})} \cap V(H) \cap \mathcal{I} \neq \emptyset$ which completes the proof for this choice of j'.

Suppose now that $t_1 \leq j' < t_2 + d + 2$. Denote for brevity $Z = X'_{j'} \cap V(H)$. Since each bag of $\mathcal{P}$ introduces at most one new vertex, Z has one vertex, call it v. Note that v is not adjacent to a vertex in S. Otherwise the fact that $S \subseteq X'_{j'}$ would imply that v and also its neighbor in S belong to H, which would contradict $|Z| = 1$. Also, $v \notin S$ because $S \subseteq X'_{t_1-1}$ and $v \notin X'_{t_1-1}$. Thus, informally speaking, we have proved that H is a connected component in $G - S$ that starts in $\mathcal{P}'$ with the vertex v and this vertex is not adjacent to any vertex in S. Note that the path decompositions $\mathcal{P}$ and $\mathcal{P}'$ when restricted to H are identical, $(X_{\alpha(H, \mathcal{P})} \cap V(H), \ldots, X_{\beta(H, \mathcal{P})} \cap V(H)) = (X'_{\alpha(H, \mathcal{P}')} \cap V(H), \ldots, X'_{\beta(H, \mathcal{P}')} \cap V(H))$. This implies in particular that $G[X_1 \cup \cdots \cup X_{\alpha(H, \mathcal{P})}]$ also has a connected component that consists of only the vertex v. Since, $\mathcal{P}$ is $\mathcal{I}$-connected, $v \in \mathcal{I}$. This implies that H has a vertex in $X'_{\alpha(H, \mathcal{P}')} \cap \mathcal{I}$, i.e., this set consists of v. $\square$

Observe that every connected component H of $G - S$ is either contained in $I(S, \mathcal{P}')$ (which means that H is an in-branch) or waits in $I(S, \mathcal{P}')$ (for all other H).

Observation 2. For any $\mathcal{P}$ and a bottleneck S, the path decomposition $F(S, \mathcal{P})$ is S-structured.

Now we want to define a series of transformations, which start with an arbitrary $\mathcal{I}$-connected path decomposition $\mathcal{P}$ and transform it into an $\mathcal{I}$-connected path decomposition with no larger width, which is S-structured for every $S \in \mathcal{S}$. For this, we will apply the F-transformations for all bottlenecks. In order to do this we need some technical lemmas about the structure of bottlenecks and their branches.
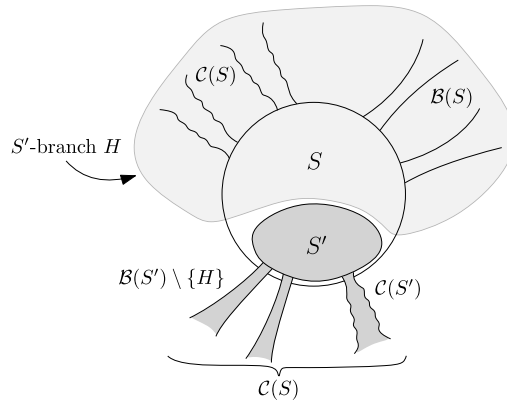
Lemma 4. Let S and S' be two bottlenecks, such that $S' \subsetneq S$. There exists an S'-branch H such that $\bigcup_{H' \in \mathcal{B}(S)} V(H') \cup S \setminus S' \subseteq V(H)$ and every S'-branch H', different than H, is a non-branch connected component of $G - S$.

Proof. Let $S, S' \in \mathcal{S}$ such that $S' \subsetneq S$. Clearly S intersects some connected component of $G - S'$. Since every S-branch H'' is connected and $S \setminus S' \subset N_G(V(H''))$, we observe that two connected components of $G - S'$ can not be distinctive, i.e., there exists a connected component H of $G - S'$ such that $\bigcup_{H' \in \mathcal{B}(S)} V(H') \cup S \setminus S' \subseteq V(H)$.

To see that H is an S'-branch, consider a vertex $s' \in S'$. By assumption, $S' \subseteq S$ and hence s' is also a vertex of S. Thus, s' has a neighbor in every S-branch H', and thus in H. See Fig. 4 for an illustration.

Now consider an S'-branch $H'' \neq H$. If every vertex of $S \setminus S'$ is adjacent to some vertex of H'', then H'' is an S-branch, a contradiction. Thus, H'' is a connected component of $G - S$, which is not an S-branch. $\square$

Lemma 5. For any two bottlenecks S and S', if $S' \nsubseteq S$, then there exists exactly one connected component H of $G - S$ such that $S' \subseteq S \cup V(H)$.

Fig. 4. An illustration for Lemma 4. S and S' are bottlenecks, such that $S' \subsetneq S$. All S-branches are subgraphs of an S'-branch H and every S'-branch, different than H, is a connected component of $G - S$, which is not an S-branch.

(a) $S \setminus S' \neq \emptyset$ and $S \cap S' \neq \emptyset$.

(b) $S \cap S' = \emptyset$.
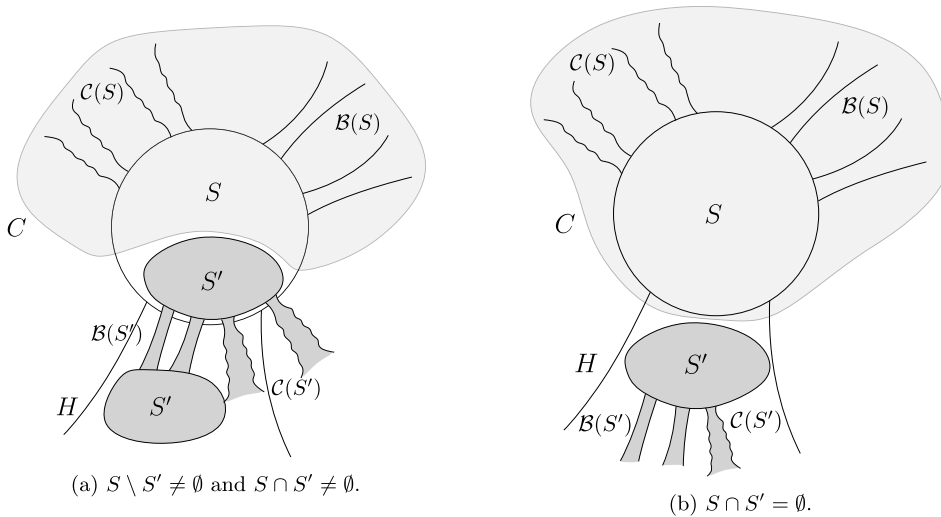
Fig. 5. An illustration of two cases in Lemma 5 and Remark 1; S and S' are bottlenecks. All S-branches (apart from H, which is a connected component of $G - S$, which may or may be not an S-branch) are subgraphs of C and $S' \subseteq S \cup V(H)$.

Proof. Let $S, S' \in \mathcal{S}$ such that $S' \nsubseteq S$. Clearly S' intersects some connected component of $G - S$. Suppose that S' has a non-empty intersection with two connected components H and H' of $G - S$. Thus, since every S'-branch H'' is connected and $N_G(V(H'')) = S'$, we observe that H'' contains a vertex of S (otherwise H, H' would not be two distinct components). Since S'-branches are vertex-disjoint, this implies that the number of such S'-branches is at most $|S|$, which is in turn at most k by Corollary 1. However, this contradicts the assumption that S' is a bottleneck. $\square$

The next remark is a straightforward consequence of Lemma 4 and Lemma 5.

Remark 1. Let S and S' be bottlenecks such that $S' \nsubseteq S$ and $S \nsubseteq S'$. Let H be the connected component of $G - S$, such that $S' \subseteq S \cup V(H)$. There exists exactly one connected component C of $G - S'$ such that $\bigcup_{H' \in \mathcal{B}(S) \setminus H} V(H') \cup S \setminus S' \subseteq V(C)$. Moreover, all S'-branches but possibly C are subgraphs of H.

See Fig. 5 for the illustration for Lemma 5 and Remark 1.
We say two bottlenecks S and S' are *well-nested* in $\mathcal{P}$ if

(i) $I(S', \mathcal{P}) \subsetneq I(S, \mathcal{P})$ or
(ii) $I(S, \mathcal{P}) \subseteq I(S', \mathcal{P})$ or
(iii) $I(S, \mathcal{P}) \cap I(S', \mathcal{P}) = \emptyset$.

Observe that the ordering of S, S' in the definition above matters.

Lemma 6. *For any $\mathcal{I}$-connected path decomposition $\mathcal{P}$ and bottlenecks S, S', if $\mathcal{P}$ is S'-structured, then S and S' are well-nested in $\mathcal{P}$. Moreover, if $S \subsetneq S'$, then either $I(S, \mathcal{P}) \cap I(S', \mathcal{P}) = \emptyset$ or $I(S', \mathcal{P}) \subseteq I(S, \mathcal{P})$.*

Proof. Let $\mathcal{P} = (X_1, \ldots, X_l)$ be an $\mathcal{I}$-connected path decomposition of G and let $S, S' \in \mathcal{S}$. Suppose $\mathcal{P}$ is S'-structured. Let us assume that $I(S', \mathcal{P}) \cap I(S, \mathcal{P}) \neq \emptyset$, we will show that either $I(S', \mathcal{P}) \subsetneq I(S, \mathcal{P})$ or $I(S, \mathcal{P}) \subseteq I(S', \mathcal{P})$.

Case A: $S' \subsetneq S$**.** By Lemma 4, there exists an S'-branch H such that for every S-branch H' it holds that $S \setminus S' \cup V(H') \subseteq V(H)$, and thus $[\alpha(H'), \beta(H')] \subseteq [\alpha(H), \beta(H)]$. Recall that $t_1(S) = \alpha(H_1')$ and $t_2(S) = \beta(H_2')$ for some in-branches $H_1', H_2' \in \mathcal{B}(S)$, so $I(S, \mathcal{P}) = [t_1(S), t_2(S)] \subseteq [\alpha(H), \beta(H)]$. Consider two subcases.

Subcase A1: $\alpha(H) \in I(S', \mathcal{P})$**.** Since $\mathcal{P}$ is S'-structured, we observe that H is an in-branch for S', which implies that $I(S, \mathcal{P}) \subseteq [\alpha(H), \beta(H)] \subseteq I(S', \mathcal{P})$.

Subcase A2: $\alpha(H) \notin I(S', \mathcal{P})$**.** First, observe that if $\alpha(H) > t_2(S')$, then $t_1(S) > t_2(S')$ and thus $I(S, \mathcal{P}) \cap I(S', \mathcal{P}) = \emptyset$, which contradicts our assumption. Analogously, if $\beta(H) < t_1(S')$, we again obtain that $I(S, \mathcal{P}) \cap I(S', \mathcal{P}) = \emptyset$. Therefore assume that $\alpha(H) < t_1(S') \leq \beta(H)$. Observe that this implies that H is a pre-branch for S' and, since $\mathcal{P}$ is S'-structured, H waits in $I(S', \mathcal{P})$. In particular $(S \setminus S') \cap X_{t_1(S')} = (S \setminus S') \cap X_{t_1(S')+1} = \ldots = (S \setminus S') \cap X_{t_2(S')}$. Since $I(S', \mathcal{P}) \cap I(S, \mathcal{P}) \neq \emptyset$, it is necessary that $t_1(S) \leq t_1(S')$ (otherwise $t_1(S) > t_2(S')$). Thus, $t_1(S) \leq t_1(S') \leq t_2(S') \leq t_2(S)$ (recall H waits in $I(S', \mathcal{P})$). Summing up, if $S' \subsetneq S$, then either $I(S', \mathcal{P}) \subsetneq I(S, \mathcal{P})$ or $I(S, \mathcal{P}) \subseteq I(S', \mathcal{P})$, which completes the proof for this case.

Case B: $S' \nsubseteq S$**.** By Lemma 5, there exists exactly one connected component H of $G - S$ such that $S' \subseteq S \cup V(H)$. Since $S' \nsubseteq S$, we observe that $V(H) \cap S' \neq \emptyset$.
  If H is an S-branch that is an in-branch in $\mathcal{P}$, then

$$I(S', \mathcal{P}) \subseteq [\alpha(H, \mathcal{P}), \beta(H, \mathcal{P})] \subseteq I(S, \mathcal{P}).$$

We observe that $I(S', \mathcal{P}) \subseteq I(S, \mathcal{P})$ is equivalent to $I(S', \mathcal{P}) \subsetneq I(S, \mathcal{P})$ or $I(S', \mathcal{P}) = I(S, \mathcal{P})$, thus S and S' are well-nested in $\mathcal{P}$. So assume that H is a connected component of $G - S$, that is not an in-branch for S (it may still be a pre- or a post-branch). We consider now two subcases.

Subcase B1: $S \subsetneq S'$**.** By Lemma 4, all S-branches possibly except for H are not S'-branches and all S'-branches are subgraphs of H.
  Because $\mathcal{P}$ is S'-structured, every S-branch but possibly H waits in $I(S', \mathcal{P})$. In particular, every in-branch H'' for S does wait in $I(S', \mathcal{P})$. Thus, for every such an in-branch H'' it holds that $I(S', \mathcal{P}) \subseteq [\alpha(H''), \beta(H'')]$ or $I(S', \mathcal{P}) \cap [\alpha(H''), \beta(H'')] = \emptyset$. Note that the second condition implies that $I(S', \mathcal{P}) \cap I(S, \mathcal{P}) = \emptyset$, which contradicts our assumption. Therefore we obtain that $I(S', \mathcal{P}) \subseteq [\alpha(H''), \beta(H'')] \subseteq I(S, \mathcal{P})$. Note that this shows the second claim of the lemma.

Subcase B2: $S \setminus S' \neq \emptyset$**.** Let C be the connected component of $G - S'$, for which it holds $\bigcup_{H' \in \mathcal{B}(S) \setminus H} V(H') \cup S \setminus S' \subseteq V(C)$, whose existence is guaranteed by Remark 1.
  If C is an in-branch for S', we observe that $I(S, \mathcal{P}) \subseteq [\alpha(C), \beta(C)] \subseteq I(S', \mathcal{P})$, because $\mathcal{P}$ is S'-structured. On the other hand, if C is not an in-branch for S', then all in-branches of S wait in $I(S', \mathcal{P})$. This is because all subgraphs but in-branches of S' wait in $I(S', \mathcal{P})$, and every in-branch for S is vertex-disjoint with every in-branch for S', since they are all contained in $V(C) \cup S'$. Thus, since $I(S, \mathcal{P}) \cap I(S', \mathcal{P}) \neq \emptyset$, we conclude that $I(S', \mathcal{P}) \subseteq I(S, \mathcal{P})$, which completes the proof.   □

In the next lemma, we show that we can apply a series of F-transformations, one for each bottleneck, so that the structure obtained in previous F-transformations is not 'destroyed' during the subsequent F-transformations.

Lemma 7. *Let S, S' be bottlenecks and let $\mathcal{P}$ be an S-structured $\mathcal{I}$-connected path decomposition. Let $\mathcal{P}' = F(S', \mathcal{P})$.*

  1. If $I(S', \mathcal{P}) \subseteq I(S, \mathcal{P})$, then $\mathcal{P}'$ is S-structured and $I(S', \mathcal{P}') \subseteq I(S, \mathcal{P}')$.
  2. If $t_2(S, \mathcal{P}) < t_1(S', \mathcal{P})$, then $\mathcal{P}'$ is S-structured and $t_2(S, \mathcal{P}') < t_1(S', \mathcal{P}')$.
  3. If $t_2(S', \mathcal{P}) < t_1(S, \mathcal{P})$, then $\mathcal{P}'$ is S-structured and $t_2(S', \mathcal{P}') < t_1(S, \mathcal{P}')$.

Proof. First, let us prove point 1, i.e. we assume that $I(S', \mathcal{P}) \subseteq I(S, \mathcal{P})$. If $S' = S$, then the results are obvious, so we consider two cases.

Case A: $S' \subsetneq S$**.** Observe that by Lemma 6 we obtain that $I(S', \mathcal{P}) \cap I(S, \mathcal{P}) = \emptyset$ or $I(S, \mathcal{P}) \subseteq I(S', \mathcal{P})$, which leads to the equality $I(S, \mathcal{P}) = I(S', \mathcal{P})$. By Lemma 4, there exists an S'-branch H, such that $\bigcup_{H' \in \mathcal{B}(S)} V(H') \cup S \setminus S' \subseteq V(H)$. Because $I(S, \mathcal{P}) = I(S', \mathcal{P})$, we conclude that H is the only in-branch for S'. The path decomposition $\mathcal{P}$ is S-structured, so every

connected component of $G - S'$, different than H, waits in $I(S', \mathcal{P}) = I(S, \mathcal{P})$. Thus, the transformation $F(S', \mathcal{P})$ does not make any changes to bags $X_{t_1(S,\mathcal{P})}, X_{t_1(S,\mathcal{P})+1}, \ldots, X_{t_2(S,\mathcal{P})}$, i.e., $I(S, \mathcal{P}') = I(S', \mathcal{P}')$.

Case B: $S' \nsubseteq S$**.** By Lemma 5, there exists exactly one connected component H of $G - S$ such that $S' \subseteq S \cup V(H)$. The assumption $I(S', \mathcal{P}) \subseteq I(S, \mathcal{P})$ implies that some vertex of H appears for the first time in $\mathcal{P}$ in the interval $I(S, \mathcal{P})$, i.e., H does not wait in $I(S, \mathcal{P})$. Moreover, it implies that $t_1(S, \mathcal{P}) \leq \alpha(H)\mathcal{P} \leq \beta(H, \mathcal{P}) \leq t_2(S, \mathcal{P})$ due to our general assumption that each bag introduces at most one new vertex. This implies that H is an in-branch for S. We are going to show now that every S'-branch H', which is an in-branch, is a subgraph of H. If $S \subsetneq S'$, then we obtain it immediately from Lemma 4, so let $S \setminus S' \neq \emptyset$. Let C be a connected component of $G - S'$, such that $\bigcup_{H' \in \mathcal{B}(S) \setminus H} V(H') \cup S \setminus S' \subseteq V(C)$, whose existence is guaranteed by Remark 1. Recall that C might or might not be an S'-branch, but for sure it is not an in-branch for S', because then $I(S, \mathcal{P}) \subsetneq I(S', \mathcal{P})$, which is a contradiction. Thus, by Remark 1, every in-branch for S' is a subgraph of H.

Because every in-branch for S' is a subgraph of H, we conclude that $I(S', \mathcal{P}) \subseteq [\alpha(H, \mathcal{P}), \beta(H, \mathcal{P})]$ and the only changes made by the transformation $F(S', \mathcal{P})$ concern the vertices of $H \in \mathcal{B}^{\text{in}}(S)$. Every connected component of $G - S$, apart from in-branches (for S'), waits in $I(S, \mathcal{P}')$, so $F(S', \mathcal{P})$ is S-structured and $I(S, \mathcal{P}') \subseteq [\alpha(H, \mathcal{P}'), \beta(H, \mathcal{P}')] \subseteq I(S', \mathcal{P}')$.

To see that cases 2 and 3 hold as well, notice that the prefix $(X_1, X_2, \ldots, X_{t_1(S', \mathcal{P})-1})$ and suffix $(X_{t_2(S', \mathcal{P})+1}, X_{t_2(S', \mathcal{P})+2}, \ldots, X_l)$ of $\mathcal{P}$ are just copied into $\mathcal{P}'$ without any changes. □

In the following lemma it is crucial that the path decomposition $\mathcal{P}$ is not only S-structured but has been obtained by applying the transformation described in (1)(a)-(1)(g) to $\mathcal{P}_0$. In particular, the bags added in (1)(c) and (1)(e) will play a crucial role in ensuring that the path decomposition returned by $F(S', \mathcal{P})$ remains S-structured.

Lemma 8. *Let* S, S' *be bottlenecks and let* $\mathcal{P}_0$ *be any* $\mathcal{I}$*-connected path decomposition. Let* $\mathcal{P} = F(S, \mathcal{P}_0)$ *and* $\mathcal{P}' = F(S', \mathcal{P})$. *If* $I(S, \mathcal{P}) \subsetneq I(S', \mathcal{P})$, *then* $\mathcal{P}'$ *is* S*-structured and* $I(S, \mathcal{P}') \subseteq I(S', \mathcal{P}')$ *or* $I(S, \mathcal{P}') \cap I(S', \mathcal{P}') = \emptyset$.

Proof. Let $\mathcal{P} = (X_1, \ldots, X_l)$ be an $\mathcal{I}$-connected path decomposition of G, let $S, S' \in \mathcal{S}$ and $\mathcal{P}' = F(S', \mathcal{P})$. Moreover, assume that $\mathcal{P} = F(S, \mathcal{P}_0)$ for some $\mathcal{I}$-connected path decomposition $\mathcal{P}_0$. In particular, this implies that $\mathcal{P}$ is S-structured. Finally, assume that $I(S, \mathcal{P}) \subsetneq I(S', \mathcal{P})$.

Case A: $S' \subsetneq S$**.** By Lemma 4 there exists an S'-branch H such that $\bigcup_{H' \in \mathcal{B}(S)} V(H') \cup S \setminus S' \subseteq V(H)$. If H is an in-branch for S' then $I(S, \mathcal{P}) \subseteq [\alpha(H, \mathcal{P}), \beta(H, \mathcal{P})] \subsetneq I(S', \mathcal{P})$. However, recall that F-transformation applied to S' and $\mathcal{P}$ does not change the structure of the bags restricted to H (or any other in-branch of S'). Therefore we conclude that $I(S, \mathcal{P}') \subseteq [\alpha(H, \mathcal{P}'), \beta(H, \mathcal{P}')] \subsetneq I(S', \mathcal{P}')$ and $\mathcal{P}'$ is S-structured.

Now assume that H is a pre-branch or a post-branch for S'. From the construction of $\mathcal{P}$ we have that $c_{\min}(S', \mathcal{P}) \notin I(S, \mathcal{P})$. Because every in-branch H' for S' either waits in $I(S, \mathcal{P})$ or $[\alpha(H', \mathcal{P}), \beta(H', \mathcal{P})] \cap I(S, \mathcal{P}) = \emptyset$, we obtain that $I(S, \mathcal{P}') \cap I(S', \mathcal{P}') = \emptyset$ and $\mathcal{P}'$ is S-structured.

Case B: $S' \nsubseteq S$ **and** $S \nsubseteq S'$**.** By Lemma 5 there exists exactly one connected component H of $G - S$ such that $S' \subseteq S \cup V(H)$. Because $V(H) \cap S' \neq \emptyset$ we have that $I(S', \mathcal{P}) \subseteq [\alpha(H, \mathcal{P}), \beta(H, \mathcal{P})]$. We observe that H cannot be an in-branch for S, otherwise $I(S', \mathcal{P}) \subseteq [\alpha(H, \mathcal{P}), \beta(H, \mathcal{P})] \subseteq I(S, \mathcal{P})$, which contradicts the assumption that $I(S, \mathcal{P}) \subsetneq I(S', \mathcal{P})$. Thus, $H \notin \mathcal{B}^{\text{in}}(S)$.

From the facts that $\mathcal{P}$ is S-structured and $I(S', \mathcal{P}) \subseteq [\alpha(H, \mathcal{P}), \beta(H, \mathcal{P})]$, we observe that H waits in $I(S, \mathcal{P})$. By Remark 1 there exists a connected component C of $G - S'$ such that $\bigcup_{H' \in \mathcal{B}(S) \setminus H} V(H') \cup S \setminus S' \subseteq V(C)$. Because H is not an in-branch for S, all in-branches of S are subgraphs of C. If C is an in-branch for S' then $I(S, \mathcal{P}) \subseteq [\alpha(C, \mathcal{P}), \beta(C, \mathcal{P})] \subsetneq I(S', \mathcal{P})$. Because F-transformation does not change bags inside one in-branch, we have that $I(S, \mathcal{P}') \subseteq [\alpha(C, \mathcal{P}'), \beta(C, \mathcal{P}')] \subsetneq I(S', \mathcal{P}')$ and $\mathcal{P}'$ is S-structured. On the other hand, assume that C is not an in-branch for S'. From construction of $\mathcal{P}$ we have that $c_{\min}(S', \mathcal{P}) \notin I(S, \mathcal{P})$ Because every in-branch H' for S' either waits in $I(S, \mathcal{P})$ or $[\alpha(H', \mathcal{P}), \beta(H', \mathcal{P})] \cap I(S, \mathcal{P}) = \emptyset$, we obtain that $I(S, \mathcal{P}') \cap I(S', \mathcal{P}') = \emptyset$ and $\mathcal{P}'$ is S-structured.

Case C: $S \subsetneq S'$**.** From Lemma 4 there exists an S-branch H such that $\bigcup_{H' \in \mathcal{B}(S')} V(H') \cup S' \setminus S \subseteq V(H)$, i.e., $I(S', \mathcal{P}) \subseteq [\alpha(H, \mathcal{P}), \beta(H, \mathcal{P})]$. We observe that H cannot be an in-branch for S, otherwise $I(S', \mathcal{P}) \subseteq [\alpha(H, \mathcal{P}), \beta(H, \mathcal{P})] \subseteq I(S, \mathcal{P})$, which contradicts the assumption that $I(S, \mathcal{P}) \subsetneq I(S', \mathcal{P})$.

Let then H be a pre-branch or a post-branch for S, which means that H waits in $I(S, \mathcal{P})$ (notice that it is impossible that $[\alpha(H, \mathcal{P}), \beta(H, \mathcal{P})] \cap I(S, \mathcal{P}) = \emptyset$, because then $I(S', \mathcal{P}) \cap I(S, \mathcal{P}) = \emptyset$). From construction of $\mathcal{P}$ we have that $c_{\min}(S', \mathcal{P}) \notin I(S, \mathcal{P})$. Because every in-branch H' for S' either waits in $I(S, \mathcal{P})$ or $[\alpha(H', \mathcal{P}), \beta(H', \mathcal{P})] \cap I(S, \mathcal{P}) = \emptyset$, we obtain that $I(S, \mathcal{P}') \cap I(S', \mathcal{P}') = \emptyset$ and $\mathcal{P}'$ is S-structured. □

Now we are ready to prove Lemma 2.

Proof of Lemma 2. Let $\mathcal{P}$ be an $\mathcal{I}$-connected path decomposition. Let $\mathcal{S} = \{S_1, S_2, \ldots, S_{n'}\}$ be the set of all bottlenecks. We define a path decomposition $\mathcal{P}' := \mathcal{P}_{n'}$ in the following recursive way:

$$\mathcal{P}_0 = \mathcal{P};$$
$$\mathcal{P}_i = F(S_i, \mathcal{P}_{i-1}), \ \text{for } i \in \{1, \ldots, n'\}.$$

We are going to prove now that $\mathcal{P}_q$ is S_j-structured and S_i, S_j are well-nested in $\mathcal{P}_q$ for every $1 \le j \le q \le n'$ and $1 \le i \le n'$.

Induction on q. If $q = 1$, then obviously $j = 1$. By Observation 2, $\mathcal{P}_q$ is S_q-structured. Thus, by Lemma 6, S_q and S_i are well-nested in $\mathcal{P}_q$ for every $1 \le i \le n'$. So assume that $q > 1$ and the claim holds for $q - 1$.

Let $j \in \{1, \ldots, q-1\}$. For every S_j we have, by the induction assumption, that S_q, S_j are well-nested in $\mathcal{P}_{q-1}$ and $\mathcal{P}_{q-1}$ is S_j-structured. By Lemma 8, if $I(S_j, \mathcal{P}_{q-1}) \subsetneq I(S_q, \mathcal{P}_{q-1})$, then $\mathcal{P}_q = F(S_q, \mathcal{P}_{q-1})$ is S_j-structured. By Lemma 7, if $I(S_q, \mathcal{P}_{q-1}) \subseteq I(S_j, \mathcal{P}_{q-1})$ or $I(S_j, \mathcal{P}_{q-1}) \cap I(S_q, \mathcal{P}_{q-1}) = \emptyset$, then $\mathcal{P}_q = F(S_q, \mathcal{P}_{q-1})$ is S_j-structured.

Because $\mathcal{P}_q$ is S_j-structured, then from Lemma 6 for any $i \in \{1, \ldots, n'\}$ we have that S_i, S_j are well-nested in $\mathcal{P}_q$.

So $\mathcal{P}'$ is S-structured for every bottleneck $S \in \mathcal{S}$. Note that Lemma 3 ensures that $\mathcal{P}'$ is an $\mathcal{I}$-connected path decomposition of width at most width($\mathcal{P}$), which finishes the proof. □

Now we are ready to show the correctness of our algorithm. We will prove it in two steps.

Lemma 9. *If G has an $\mathcal{I}$-connected path decomposition of width at most $k-1$, then $Tab[s] = true$ for some state s such that* cover$(s) = V(G)$.

Proof. Suppose that G has an $\mathcal{I}$-connected path decomposition of width $k - 1$. By Lemma 2, there exists an $\mathcal{I}$-connected path decomposition $\mathcal{P} = (X_1, \ldots, X_l)$ that has width $k - 1$ and is S-structured for each bottleneck S.

By Lemma 6, the set $\mathcal{S}$ of all bottlenecks with relation $S \prec S'$ if and only if $I(S, \mathcal{P}) \subseteq I(S', \mathcal{P})$ forms a partial order (assuming that any ties, i.e. when $I(S, \mathcal{P}) = I(S', \mathcal{P})$, are resolved arbitrarily). Let $S_1, \ldots, S_t$ be the maximal elements with respect to this partial order. Note that for $i \ne j$, $I(S_i, \mathcal{P}) \cap I(S_j, \mathcal{P}) = \emptyset$ and for any bottleneck $S' \notin \{S_1, \ldots, S_t\}$ we have $I(S', \mathcal{P}) \subseteq I(S_i, \mathcal{P})$ for some $i \in \{1, \ldots, t\}$. Assume without loss of generality that the 'maximal' bottlenecks are ordered according to the left endpoints of their intervals,

$$t_1(S_1) \le t_1(S_2) \le \cdots \le t_1(S_t).$$

We show how to arrive at the desired state s. To that end we argue, by induction on j, that for each

$$j \in J := \{1, \ldots, l\} \setminus \bigcup_{i=1}^{t} \{t_1(S_i), \ldots, t_2(S_i) - 1\}$$

there exists a state s_j such that cover$(s_j) = G[X_1 \cup \cdots \cup X_j]$, bag$(s_j) = X_j$ and $Tab[s_j] = true$.

Since the first bag of $\mathcal{P}$ consists of a vertex in $\mathcal{I}$, this clearly holds for $j = 1$ so take $j > 1$ and assume that the claim is true for each $j' \in J \cap \{1, \ldots, j-1\}$. We consider two cases.

Case A: $j \notin I(S_i, \mathcal{P})$ **for each** $i \in \{1, \ldots, t\}$. Hence we have $j - 1 \in J$. This implies, according to Lemma 1, that for each bottleneck S, either there are at most $2k$ S-branches H such that $\alpha(H) \le j$, or there are at most $2k$ S-branches such that $j \le \beta(H)$. Thus, there exists a state s_j such that cover$(s_j) = G[X_1 \cup \cdots \cup X_j]$ and bag$(s_j) = X_j$. The step extension rule and $Tab[s_{j-1}] = true$, which holds by the induction hypothesis, imply $Tab[s_j] = true$ as required.

Case B: $j \in I(S_i, \mathcal{P})$ **for some bottleneck** S_i. By the definition of the set J, $j = t_2(S_i)$. Hence, the preceding index of j in the set J is $j' = t_1(S_i) - 1$. Again, by the definition of $I(S_i, \mathcal{P})$, Lemma 1, and the maximality of S_i with respect to the partial order, we have that for each bottleneck set S either at most $2k$ S-branches H satisfy $\alpha(H) \le j'$, or at least $|\mathcal{B}(S_{i'})| - 2k$ S-branches H are contained in $[1, j']$, i.e., satisfy $\beta(H) \le j'$, depending whether $t_2(S) \le j'$ or $t_1(S) > j'$. Thus, there exists a state s_j such that cover$(s_j) = G[X_1 \cup \cdots \cup X_j]$ and bag$(s_j) = X_j$. Consider the jump extension rule constructed for $S' = S_i$. For the set $B_{S'}$ in (J1) and (J2) take all S'-branches that are not in-branches, i.e., those that are covered in $[j' + 1, j]$ in $\mathcal{P}$. Note that each S-branch of each bottleneck $S \ne S_i$ such that $S \subseteq X_{j'}$ waits in the interval $I(S_i, \mathcal{P})$ because $\mathcal{P}$ is S_i-structured, which ensures the condition (J3). Condition (J4) holds because the decomposition $\mathcal{P}_H$ in (J4) exists which is certified by the decomposition $\mathcal{P}$, namely $\mathcal{P}_H = (X_{\alpha(H)} \cap V(H), \ldots, X_{\beta(H)} \cap V(H))$. Thus, $Tab[j'] = true$ (which holds by the induction hypothesis) ensures that $Tab[j] = true$.

Finally observe that $l \in J$, cover$(s_l) = G[X_1 \cup \cdots \cup X_l] = V(G)$ and $Tab[s_l] = true$. Thus, $s = s_l$ is the required state. □

Lemma 10. *For any state s, if $Tab[s] = true$, then G_s has an $\mathcal{I}$-connected path decomposition of width at most $k - 1$.*

Proof. Proof by induction on the position of s in the ordering $\prec$. First, let $\mathrm{cover}(s) = \{v\}$ for some $v \in V(G)$ (notice that such states are smallest, according to $\prec$). If $v \in \mathcal{I}$, then $Tab[s]$ was set true in the initialization step. This is justified by considering connected path decomposition consisting of a single bag $\{v\}$, which is a proper connected path decomposition of the single-vertex graph $(\{v\}, \emptyset)$. On the other hand, if $v \notin \mathcal{I}$, then $Tab[s]$ is never set *true*, as the extension rules apply only to states with $|\mathrm{cover}(s)| > 1$.

Now suppose that $|\mathrm{cover}(s)| \geq 2$, and the Lemma holds for all states $w \prec s$. Since $Tab[s] = true$, its value must have been set by one of the extension rules. Consider two cases.

Case 1: $Tab[s]$ **was set by step extension.** Consider the state w. If $\mathrm{bag}(s) \nsubseteq \mathrm{bag}(w)$, then since by (S4), $\mathrm{bag}(s) \cap \mathrm{cover}(w) \subseteq \mathrm{bag}(w)$, we have that $\mathrm{bag}(s) \nsubseteq \mathrm{cover}(w)$. Therefore, $\mathrm{cover}(s) = \mathrm{cover}(w) \cup \mathrm{bag}(s)$ implies that $|\mathrm{cover}(s)| > |\mathrm{cover}(w)|$, which means that $w \prec s$. On the other hand, if $\mathrm{bag}(s) \subseteq \mathrm{bag}(w)$, we have $\mathrm{cover}(s) = \mathrm{cover}(w)$ due to (S3). However, since s and w, are distinguishable, we have $\mathrm{bag}(s) \subsetneq \mathrm{bag}(w)$, so $|\mathrm{bag}(s)| < |\mathrm{bag}(w)|$ and thus $w \prec s$.

So, by the inductive assumption, $Tab[w]$ was set properly and there exists an $\mathcal{I}$-connected path decomposition $\mathcal{P} = (X_1, X_2, \ldots, X_l)$ of G_w that has width at most $k - 1$, where $X_l = \mathrm{bag}(w)$. Let $X_{l+1} := \mathrm{bag}(s)$ and let $\mathcal{P}' := (X_1, X_2, \ldots, X_l, X_{l+1})$.

We claim that $\mathcal{P}'$ is a path decomposition of G_s. Indeed, $\bigcup_{i=1}^{l+1} X_i = \bigcup_{i=1}^{l} X_i \cup X_{l+1} = \mathrm{cover}(w) \cup \mathrm{bag}(s) = \mathrm{cover}(s)$. Now, consider an edge vu of G_s. If both v, u belong to $\mathrm{cover}(w)$, they appear in some X_i for $i \leq l$ (by the inductive assumption). If both v, u belong to $\mathrm{bag}(s)$, we are done too. Finally, if $v \in \mathrm{cover}(w)$ and $u \in \mathrm{bag}(s) \setminus \mathrm{cover}(w)$, we know from (S2) that $v \in \mathrm{border}(w) \subseteq \mathrm{bag}(s)$, so we are again in the previous case. Now suppose for a contradiction that there are some $1 \leq i < j \leq l$, such that $X_i \cap X_{l+1} \nsubseteq X_j$. This means that $\mathrm{bag}(s) = X_{l+1}$ contains a vertex of $\mathrm{cover}(w) \setminus \mathrm{bag}(w)$, which is a contradiction with (S4).

Moreover, since $\mathrm{width}(\mathcal{P}) \leq k - 1$ and $|\mathrm{bag}(s)| \leq k$, we have $\mathrm{width}(\mathcal{P}') \leq k - 1$. Finally we prove that $\mathcal{P}'$ is $\mathcal{I}$-connected. Consider any connected component H of $G[X_1 \cup \cdots \cup X_i]$ for some $i \in \{1, \ldots, l+1\}$. If $\alpha(H, \mathcal{P}') \leq l$, then $X_{\alpha(H, \mathcal{P})} \cap V(H) = X_{\alpha(H, \mathcal{P}')} \cap V(H)$ contains a vertex from $\mathcal{I}$ because $\mathcal{P}$ is $\mathcal{I}$-connected. Otherwise $\alpha(H, \mathcal{P}') = l + 1$. Then clearly $i = l + 1$ and H is a connected component of $G[X_{l+1}]$. By (S1), H contains a vertex from $\mathcal{I}$. Thus, $\mathcal{P}'$ is $\mathcal{I}$-connected. This justifies setting $Tab[s] = true$.

Case 2: $Tab[s]$ **was set by jump extension.** Let $w = (X, \{B_S\}_S, \{f^{B_S}\}_S)$, $s = (X, \{B_S\}_S, \{g_S^B\}_S)$ and let $S' \subseteq X$ be defined as in the definition of the jump extension. To simplify the notation, set $\mathcal{B}' := \mathcal{B}(S') \setminus B_{S'} = \{H_1, H_2, \ldots, H_m\}$. Observe that since S' is a bottleneck, we have $|\mathcal{B}(S')| \geq 2k + 1$, thus there is at least one $H \in \mathcal{B}'$. Since $V(H) \nsubseteq \mathrm{cover}(w)$ and $V(H) \subseteq \mathrm{cover}(s)$ by (J1) and (J3), we have $|\mathrm{cover}(w)| < |\mathrm{cover}(s)|$ and thus $w \prec s$. So, by the inductive assumption, $Tab[w]$ was set properly to be *true* and there exists an $\mathcal{I}$-connected path decomposition $\mathcal{P} = (X_1, X_2, \ldots, X_l)$ of G_w with width at most $k - 1$, where $X_l = X = \mathrm{bag}(w)$. By (J4), for every $H \in \mathcal{B}'$ there is a path decomposition $\mathcal{P}_H = (X_1^H, X_2^H, \ldots, X_{l(H)}^H)$ of width at most $k - |X| - 1$, such that X_1^H contains a neighbor of S' or a vertex in $\mathcal{I}$, i.e., $\mathcal{P}_H$ is $\big(N_G(S') \cap V(H)\big) \cup \mathcal{I}$-connected.

We claim that

$$\mathcal{P}' = \mathcal{P} \circ \left(\prod_{i=1}^{m} \prod_{j=1}^{l(H_i)} \left(X_l \cup X_j^{H_i} \right) \right) \circ X_l,$$

where both $\circ$ and $\prod$ denote concatenation of appropriate sequences, is an $\mathcal{I}$-connected path decomposition of G_s of width at most $k - 1$.

First, observe that $\mathrm{cover}(s) = \mathrm{cover}(w) \cup \bigcup_{H \in \mathcal{B}'} V(H)$ due to (J2) and (J3). By the definition of $\mathcal{P}$ and decompositions $\mathcal{P}_H$ for $H \in \mathcal{B}'$, we observe that $\mathcal{P}'$ covers exactly $\mathrm{cover}(s)$.

Now consider an edge vu of G_s. If both vertices v, u belong to $\mathrm{cover}(w)$, or to $V(H)$ for some $H \in \mathcal{B}'$, then, by the definition of $\mathcal{P}$ and $\mathcal{P}_H$, both v and u appear in some bag of the decomposition $\mathcal{P}'$. If $v \in \mathrm{cover}(w)$ and $u \in V(H)$ for some $H \in \mathcal{B}'$, then we know that $v \in \mathrm{border}(w)$ and therefore $v \in X_l$, so both vertices appear in every bag containing u. Finally, we observe that there are no edges joining vertices from different S'-branches.

The third condition of the definition of path decomposition follows directly from the definition of $\mathcal{P}$ and $\mathcal{P}_H$ and the fact that subgraphs H are S'-branches.

Observe that $|X_i| \leq k$ for $i \leq l$ (by the definition of $\mathcal{P}$), and since $|X_j^H| \leq k - |X|$ for every H and j, we have $|X \cup X_j^H| \leq k$, so $\mathrm{width}(\mathcal{P}') \leq k - 1$.

Denote $\mathcal{P}' = (X_1, \ldots, X_l, X_{l+1}, \ldots, X_{l'})$. Consider any connected component H of $G[X_1 \cup \cdots \cup X_i]$ for some $i \in \{1, \ldots, l'\}$. If $i \leq l$, then H has a vertex from $X_{\alpha(H, \mathcal{P}')} \cap \mathcal{I} = X_{\alpha(H, \mathcal{P})} \cap \mathcal{I}$ as required. So, let $i > l$. If H is contained in some subgraph in $\mathcal{B}'$, then by (J4), $\big(N_G(S') \cap V(H)\big) \cup \mathcal{I}$-connected and hence H has a vertex from $X_{\alpha(H, \mathcal{P}')} \cap \mathcal{I}$. If H is not contained in any subgraph from $\mathcal{B}'$, then again by (J4), H has a vertex in X_l. But then, by $\mathcal{I}$-connectivity of $\mathcal{P}$, H has a vertex in $X_{\alpha(H, \mathcal{P}')} \cap \mathcal{I}$. Thus, $\mathcal{P}'$ is $\mathcal{I}$-connected.

This completes the proof. $\square$

Combining Lemmas 9 and 10, we obtain the following corollary.

Corollary 2. *The algorithm is correct, i.e., the value of $Tab[s]$ is true for some state s with $\mathrm{cover}(s) = V(G)$ if and only if $\mathrm{cpw}(G) \leq k - 1$.* □

Now let us estimate the computational complexity of our algorithm.

Lemma 11. *For every fixed $k \geq 1$, a graph G with n vertices, and $\mathcal{I} \subseteq V$, there is an algorithm deciding in time $f(k) \cdot n^{O(k^2)}$ whether G has an $\mathcal{I}$-connected path decomposition of width at most $k - 1$, where f is a function depending on k only.*

Proof. We do induction on k. First, observe that for a connected graph G, $\mathrm{cpw}(G) = 0$ if and only if G is a single-vertex graph. Moreover, $\mathrm{cpw}(G) = 1$ if and only if G is a caterpillar, and optimal connected path decompositions of caterpillars have very simple structure, so we can verify in polynomial time whether there is an $\mathcal{I}$-connected one.

So assume that $k \geq 2$ and that the claim holds for $k - 1$. For given G and $\mathcal{I}$, we run the dynamic programming algorithm that we described in Section 3. The correctness of the algorithm follows from Corollary 2. Now let us estimate its computational complexity. Recall that the total number of states is $O(n^{3k})$, so the total number of pairs of states is $O(n^{6k})$. For each pair of states we check if one of the two extension rules can be applied.

Observe that for each state s, we can compute $\mathrm{cover}(s)$, $\mathrm{bag}(s)$ and $\mathrm{border}(s)$ in polynomial time. Thus checking if the step extension can be applied can also be done in polynomial time.

Now consider the possible jump extension from a state w to a state s. Verifying the first three conditions can be clearly done in polynomial time. We check in (J4) if the appropriate path decomposition $\mathcal{P}_H$ of each S'-branch H exists by calling the algorithm recursively with the initial set $\left(N_G(S') \cap V(H)\right) \cup \mathcal{I}$. By the inductive assumption, this can be done in total time bounded by $n^{O(1)} \cdot f'(k-1)n^{c \cdot (k-1)^2}$, for some function f' and a constant c'. This gives total time complexity

$$n^{O(1)} \cdot n^{6k} \cdot f'(k-1) \cdot n^{c'(k-1)^2} = f(k) \cdot n^{O(k^2)}$$

for some function f. □

Now, the main result of the paper follows easily from Lemma 11.

Theorem 1. *For every fixed $k \geq 1$, there is an algorithm deciding in time $f(k) \cdot n^{O(k^2)}$ whether $\mathrm{cpw}(G) \leq k - 1$, for some function f depending on k only, i.e., in time polynomial in n.*

Proof. For every vertex $s^* \in V$, we run the dynamic programming algorithm for $\mathcal{I} = \{s^*\}$, i.e., we exhaustively guess a vertex in the first bag of some fixed solution. By Lemma 11, the total running time is as claimed. □

Let us point out that we did not try to optimize the dependence of the degree of the polynomial function in Theorem 1 on k, as we were only interested in finding a polynomial algorithm.

5. Open problems

As pointed out, both pathwidth and connected pathwidth are asymptotically the same for an arbitrary graph G, namely $\mathrm{cpw}(G)/\mathrm{pw}(G) \leq 2 + o(1)$. However, there are several open questions regarding the complexity of exact algorithms for connected pathwidth. One such immediate question that is a natural next step in the context of our work is whether connected pathwidth is FPT with respect to this parameter. We conjecture that this is indeed the case.

Conjecture 1. *Determining whether a given graph with n vertices has connected pathwidth at most k can be done in time $f(k) \cdot n^{O(1)}$, for some function f, i.e., the problem is FPT with respect to k.*

Also, it is not known if connected pathwidth can be computed faster than in time $O^*(2^n)$ for an arbitrary n-vertex graph (recall that this is possible for pathwidth).

Finally, let us point out that the notion of connected pathwidth appeared in the context of pursuit-evasion games called node search, edge search or mixed search. A challenging and long-standing open question related to those games is whether their connected variants belong to NP. See [2] for more details regarding this question.

Acknowledgement

References

[1] E. Amir, Approximation algorithms for treewidth, Algorithmica 56 (4) (2010) 448–479.
[2] L. Barrière, P. Flocchini, F. Fomin, P. Fraigniaud, N. Nisse, N. Santoro, D. Thilikos, Connected graph searching, Inf. Comput. 219 (2012) 1–16.
[3] L. Barrière, P. Fraigniaud, N. Santoro, D. Thilikos, Connected and Internal Graph Searching, Technical Report, UPC Barcelona, 2002.
[4] M.J. Best, A. Gupta, D.M. Thilikos, D. Zoros, Contraction obstructions for connected graph searching, Discrete Appl. Math. 209 (2016) 27–47.
[5] T.C. Biedl, T. Bläsius, B. Niedermann, M. Nöllenburg, R. Prutkin, I. Rutter, Using ILP/SAT to determine pathwidth, visibility representations, and other grid-based graph drawings, in: 21st International Symposium on Graph Drawing (GD 2013), Bordeaux, France, September 23–25, 2013, 2013, pp. 460–471, Revised Selected Papers.
[6] L. Blin, P. Fraigniaud, N. Nisse, S. Vial, Distributed chasing of network intruders, Theor. Comput. Sci. 399 (1–2) (2008) 12–37.
[7] H.L. Bodlaender, A linear-time algorithm for finding tree-decompositions of small treewidth, SIAM J. Comput. 25 (6) (1996) 1305–1317.
[8] H.L. Bodlaender, P.G. Drange, M.S. Dregi, F.V. Fomin, D. Lokshtanov, M. Pilipczuk, A $c^k n$ 5-approximation algorithm for treewidth, SIAM J. Comput. 45 (2) (2016) 317–378.
[9] H.L. Bodlaender, F.V. Fomin, A.M.C.A. Koster, D. Kratsch, D.M. Thilikos, A note on exact algorithms for vertex ordering problems on graphs, Theory Comput. Syst. 50 (3) (2012) 420–432.
[10] H.L. Bodlaender, T. Kloks, Efficient and constructive algorithms for the pathwidth and treewidth of graphs, J. Algorithms 21 (2) (1996) 358–402.
[11] P. Borowiecki, D. Dereniowski, L. Kuszner, Distributed graph searching with a sense of direction, Distrib. Comput. 28 (3) (2015) 155–170.
[12] V. Bouchitté, I. Todinca, Treewidth and minimum fill-in: grouping the minimal separators, SIAM J. Comput. 31 (1) (2001) 212–232.
[13] D. Coudert, A Note on Integer Linear Programming Formulations for Linear Ordering Problems on Graphs, Technical Report, Inria; I3S; Universite Nice Sophia Antipolis; CNRS, 2016.
[14] D. Coudert, D. Mazauric, N. Nisse, Experimental evaluation of a branch-and-bound algorithm for computing pathwidth and directed pathwidth, ACM J. Exp. Algorithmics 21 (1) (2016) 1.3.
[15] D. Dereniowski, Connected searching of weighted trees, Theor. Comput. Sci. 412 (2011) 5700–5713.
[16] D. Dereniowski, From pathwidth to connected pathwidth, SIAM J. Discrete Math. 26 (4) (2012) 1709–1732.
[17] D. Dereniowski, W. Kubiak, Y. Zwols, The complexity of minimum-length path decompositions, J. Comput. Syst. Sci. 81 (8) (2015) 1715–1747.
[18] J.A. Ellis, I.H. Sudborough, J.S. Turner, Graph separation and search number, in: Proc. of the 21st Allerton Conference on Communication Control and Computing, 1983.
[19] U. Feige, M. Hajiaghayi, J.R. Lee, Improved approximation algorithms for minimum weight vertex separators, SIAM J. Comput. 38 (2) (2008) 629–657.
[20] P. Flocchini, M. Huang, F. Luccio, Contiguous search in the hypercube for capturing an intruder, in: Proc. 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS 2005), IEEE Computer Society, Washington, DC, USA, 2005, p. 62.
[21] F. Fomin, D. Thilikos, I. Todinca, Connected graph searching in outerplanar graphs, Electron. Notes Discrete Math. 22 (2005) 213–216.
[22] F.V. Fomin, Complexity of connected search when the number of searchers is small, in: Open Problems of GRASTA 2017: The 6th Workshop on GRAph Searching, Theory and Applications, 2017.
[23] F.V. Fomin, D.M. Thilikos, An annotated bibliography on guaranteed graph searching, Theor. Comput. Sci. 399 (3) (2008) 236–245.
[24] P. Fraigniaud, N. Nisse, Connected treewidth and connected graph searching, in: Proc. 7th Latin American Theoretical Informatics Symposium (LATIN 2006), Valdivia, Chile, 2006, pp. 479–490.
[25] P. Fraigniaud, N. Nisse, Monotony properties of connected visible graph searching, Inf. Comput. 206 (12) (2008) 1383–1393.
[26] M. Fürer, Faster computation of path-width, in: Proc. 27th International Workshop on Combinatorial Algorithms (IWOCA 2016), Helsinki, Finland, August 17–19, 2016, 2016, pp. 385–396.
[27] D. Ilcinkas, N. Nisse, D. Soguet, The cost of monotonicity in distributed graph searching, Distrib. Comput. 22 (2) (2009) 117–127.
[28] L.M. Kirousis, C.H. Papadimitriou, Searching and pebbling, Theor. Comput. Sci. 47 (1986) 205–218.
[29] K. Kitsunai, Y. Kobayashi, K. Komuro, H. Tamaki, T. Tano, Computing directed pathwidth in $o(1.89^n)$ time, in: Proc. 7th International Symposium on Parameterized and Exact Computation (IPEC 2012), Ljubljana, Slovenia, September 12–14, 2012, 2012, pp. 182–193.
[30] J. Lagergren, Efficient parallel algorithms for graphs of bounded tree-width, J. Algorithms 20 (1) (1996) 20–44.
[31] N. Nisse, Connected graph searching in chordal graphs, Discrete Appl. Math. 157 (12) (2008) 2603–2610.
[32] N. Nisse, D. Soguet, Graph searching with advice, Theor. Comput. Sci. 410 (14) (2009) 1307–1318.
[33] T. Parsons, Pursuit-evasion in a graph, in: Theory and Applications of Graphs, in: Lecture Notes in Mathematics, vol. 642, Springer, 1978, pp. 426–441.
[34] B.A. Reed, Finding approximate separators and computing tree width quickly, in: Proc. 24th Annual ACM Symposium on Theory of Computing (STOC 1992), May 4–6, 1992, Victoria, British Columbia, Canada, 1992, pp. 221–228.
[35] N. Robertson, P.D. Seymour, Graph minors. I. Excluding a forest, J. Comb. Theory, Ser. B 35 (1) (1983) 39–61.
[36] N. Robertson, P.D. Seymour, Graph minors. XX. Wagner's conjecture, J. Comb. Theory, Ser. B 92 (2) (2004) 325–357.
[37] K. Suchan, Y. Villanger, Computing pathwidth faster than 2^n, in: Proc. 4th International Workshop on Parameterized and Exact Computation (IWPEC 2009), Copenhagen, Denmark, September 10–11, 2009, 2009, pp. 324–335, Revised Selected Papers.
[38] A. Takahashi, S. Ueno, Y. Kajitani, Mixed searching and proper-path-width, Theor. Comput. Sci. 137 (2) (1995) 253–268.
[39] B. Yang, D. Dyer, B. Alspach, Sweeping graphs with large clique number, Discrete Math. 309 (18) (2009) 5770–5780.