Journal of Computer and System Sciences 81 (2015) 1715–1747

Contents lists available at ScienceDirect

Journal of Computer and System Sciences

www.elsevier.com/locate/jcss

The complexity of minimum-length path decompositions

Dariusz Dereniowski [a,*], Wieslaw Kubiak [b], Yori Zwols [c,1]

[a] Faculty of Electronics, Telecommunications and Informatics, Gdańsk University of Technology, Gdańsk, Poland
[b] Memorial University, St. John's, Canada
[c] Google DeepMind, London, United Kingdom

ARTICLE INFO

Article history:
Received 14 August 2013
Received in revised form 9 March 2015
Accepted 10 June 2015
Available online 5 August 2015

Keywords:
Graph searching
Path decomposition
Pathwidth

ABSTRACT

We consider a bicriterion generalization of the pathwidth problem: given integers k, l and a graph G, does there exist a path decomposition of G of *width* at most k and *length* (i.e., number of bags) at most l? We provide a complete complexity classification of the problem in terms of k and l for general graphs. Contrary to the original pathwidth problem, which is fixed-parameter tractable with respect to k, the generalized problem is NP-complete for any fixed $k \geq 4$, and also for any fixed $l \geq 2$. On the other hand, we give a polynomial-time algorithm that constructs a minimum-length path decomposition of width at most $k \leq 3$ for any disconnected input graph. As a by-product, we obtain an almost complete classification for connected graphs: the problem is NP-complete for any fixed $k \geq 5$, and polynomial for any $k \leq 3$.

© 2015 Elsevier Inc. All rights reserved.

1. Introduction

The notions of pathwidth and treewidth of graphs have been introduced in a series of graph minor papers by Robertson and Seymour, starting with [36]. Since then the pathwidth and treewidth of graphs have been receiving growing interest due to their connections to several other combinatorial problems and numerous practical applications. In particular, pathwidth is closely related to interval thickness, the gate matrix layout problem, the vertex separation number, the node search number, and narrowness; see, e.g., [16,26–28,30,33].

In this paper we focus on computing minimum-length path decompositions of bounded width. More formally, the input to the decision version of this problem consists of a graph G and two integers k and l. The question is whether there exists a path decomposition $\mathcal{P}$ of G such that the width of $\mathcal{P}$ is at most k and the length of $\mathcal{P}$ is at most l. Clearly, this decision problem is NP-complete, because the pathwidth computation problem itself is an NP-hard problem, as shown by Arnborg et al. [1]; see also Brandenburg and Herrmann [13]. On the other hand, it can be decided in linear time whether the pathwidth of a given graph G is at most k for any fixed k, and if the answer is affirmative, then a path decomposition of width at most k can be also computed in linear time; see, e.g., the works of Bodlaender and Kloks [6,9,11]. However, as we prove in this paper, finding a minimum-length path decomposition of width k is an NP-hard problem for any fixed value of $k \geq 4$, which answers one of the open questions stated in [13]. For a detailed analysis of the complexity of the pathwidth computation problem, see, e.g., [8,29].

* Corresponding author.
E-mail addresses: deren@eti.pg.gda.pl (D. Dereniowski), wkubiak@mun.ca (W. Kubiak), yori@google.com (Y. Zwols).

[1] This research was conducted while this author was at McGill University and Concordia University.

http://dx.doi.org/10.1016/j.jcss.2015.06.011
0022-0000/© 2015 Elsevier Inc. All rights reserved.

To the best of our knowledge, no algorithmic results are known for minimum-length path decompositions. However, some research has been done on simultaneously bounding the diameter and the width of tree decompositions. In particular, Bodlaender [5] gives a (parallel) algorithm that transforms a given tree decomposition of width k for G into a binary tree decomposition of width at most $3k+2$ and depth $O(\log n)$, where n is the number of vertices of G. A more detailed analysis of the trade-off between the width and the diameter of tree decompositions can be found in work of Bodlaender and Hagerup [10]. Recently, Li et al. [31] reported on the problem of computing tree decompositions with a minimum number of bags. We also note that minimum-length path decompositions of width k can be computed for k-connected graphs; see [22,23]. For other optimization criteria studied in the context of path decompositions, see, e.g., [14,17–19,35].

We briefly note that pathwidth has a number of equivalent counterparts with alternative formulations of our generalized problem. For example, finding the pathwidth of a graph is equivalent to computing its node search number. Through this correspondence, the pathwidth is related to several other search numbers of a graph such as the edge search number, the mixed search number or the connected search number; see e.g. [15,26–28,33] for details. The parameter l studied in this paper corresponds to the search time equal the number of steps in a node search strategy; see [13]. Our generalized problem can also be reformulated as a problem of finding an interval supergraph H of a given graph G such that the maximum clique in H is of size at most k (i.e., the interval thickness of G is bounded by k) and H has an interval diagram of length at most l, where the length of an interval diagram equals the number of distinct left endpoints of intervals in the diagram. The equivalence of interval thickness and pathwidth was shown by Bodlaender [7].

Applications of minimum-length path decompositions include step-wise tile assembly (see Manuch et al. [32]), the partner units problem (see Aschinger et al. [2]), scheduling and register allocation (see [24,34,37]) where the length corresponds to schedule completion time, and the above-mentioned graph searching games (see, e.g., [3,20] for surveys).

1.1. Preliminaries

We now formally introduce the graph theoretic notation used, and the problems studied in this paper. Let $G = (V(G), E(G))$ be a simple graph and let $X \subseteq V(G)$. We denote by $G[X]$ the subgraph *induced* by X, i.e., $G[X] = (X, \{e \in E(G) \mid e \subseteq X\})$ and by $G - X$ the subgraph obtained by removing the vertices in X (together with the incident edges) from G, i.e., $G - X = G[V(G) \setminus X]$. Given $v \in V(G)$, $N_G(v)$ is the *neighborhood* of v in G, that is, the set of vertices adjacent to v in G, and $N_G(X) = (\bigcup_{v \in X} N_G(v)) \setminus X$ for any $X \subseteq V(G)$. We say that a vertex v of G is *universal* if $N_G(v) = V(G) \setminus \{v\}$. A maximal connected subgraph of G is called a *connected component* of G. A graph G is *connected* if it has at most one connected component. Given a subgraph H of G we refer to the set of vertices of H that have a neighbor in $V(G) \setminus V(H)$ as the *border* of H in G, and denote it by $\delta_G(H)$. We define $\mathcal{C}_G(X)$ to be the set of connected components H of $G - X$, called *X-components*, such that $N_G(V(H)) = X$. (Note that $\mathcal{C}_G(X)$ may be empty even if $G - X$ has connected components.) Thus, for each $H \in \mathcal{C}_G(X)$, every vertex in X has a neighbor in $V(H)$. Moreover, let $\mathcal{C}^1_G(X) \subseteq \mathcal{C}_G(X)$ denote the set of connected components of $G - X$ that consist of a single vertex, and let $\mathcal{C}^*_G(X) = \mathcal{C}_G(X) \setminus \mathcal{C}^1_G(X)$. We sometimes drop the subscript G whenever G is clear from the context.

For a positive integer n, we denote by K_n a complete graph on n vertices, and by P_n a path graph on n vertices. For any graph G, any of its complete subgraphs is called a *clique* of G. We now define a path decomposition of a graph.

Definition 1.1. 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 $i = 1, \ldots, l$, and

(PD1) $\bigcup_{i=1,\ldots,l} X_i = V(G)$,
(PD2) for each $\{u, v\} \in E(G)$ there exists $i \in \{1, \ldots, l\}$ such that $u, v \in X_i$,
(PD3) for each i, j, k with $1 \le i \le j \le k \le l$ it holds that $X_i \cap X_k \subseteq X_j$.

The *width* (respectively the *length*) of the path decomposition $\mathcal{P}$ is $\mathrm{width}(\mathcal{P}) = \max_{i=1,\ldots,l} |X_i| - 1$ ($\mathrm{len}(\mathcal{P}) = l$, respectively). The *pathwidth* of G, $\mathrm{pw}(G)$, is the minimum width over all path decompositions of G. The *size* of $\mathcal{P}$, denoted by $\mathrm{size}(\mathcal{P})$, is given by $\mathrm{size}(\mathcal{P}) = \sum_{t=1}^{l} |X_t|$.

We observe that condition **(PD3)** is equivalent to the following condition:

(PD3′) for each i, k with $1 \le i \le k \le l$, if $v \in X_i$ and $v \in X_k$, then $v \in X_j$ for all $i \le j \le k$.

We also make the following useful observation:

Observation 1.2. *Let $\mathcal{P} = (X_1, \ldots, X_l)$ be a path decomposition of a graph G. If $a \in X_i$ and $b \in X_j$ for some $1 \le i \le j \le l$, then any path P between a and b in G has a non-empty intersection with each X_k for $i \le k \le j$.*

Given a simple graph G and an integer k, in the problem PD (*Path Decomposition*) we ask whether $\mathrm{pw}(G) \le k$. In the optimization problem MLPD-CONSTR (*Minimum-Length Path Decomposition*) the goal is to compute, for a given simple graph

MOST WIEDZY Downloaded from mostwiedzy.pl

Table 1
The complexity results for MLPD(k)-CONSTR obtained in this paper. Corollary 2.3 is a direct consequence of a theorem of Gustedt [25]; see the end of Section 2.

		Connected graphs	General graphs
MLPD(k)-CONSTR	$k \le 3$	polynomial-time (Theorem 5.11)	polynomial-time (Theorem 5.11)
	$k = 4$	?	NP-hard (Theorem 2.1)
	$k \ge 5$	NP-hard (Theorem 2.2)	NP-hard (Theorem 2.2)
MLPD-CONSTR	$l \ge 2$	NP-hard (Corollary 2.3)	

G and an integer k, a minimum-length path decomposition $\mathcal{P}$ of G such that width($\mathcal{P}$) $\le k$. In the corresponding decision problem MLPD, a simple graph G and integers k, l are given, and we ask whether there exists a path decomposition $\mathcal{P}$ of G such that width($\mathcal{P}$) $\le k$ and len($\mathcal{P}$) $\le l$.

Finally, in the optimization problem MLPD(k)-CONSTR the goal is to compute, for a given simple graph G, a minimum-length path decomposition $\mathcal{P}$ of G such that width($\mathcal{P}$) $\le k$. The corresponding decision problem MLPD(k) the input consists of a simple graph G and an integer l and the question is whether there exists a path decomposition $\mathcal{P}$ of G such that width($\mathcal{P}$) $\le k$ and len($\mathcal{P}$) $\le l$.

Note that the difference between MLPD-CONSTR and MLPD(k)-CONSTR (and, similarly, MLPD and MLPD(k)) is that in the former k is a part of the input while in the latter the value of k is fixed.

1.2. Overview of our results and organization of this paper

In this paper, we investigate the complexity of MLPD(k)-CONSTR for different values of k and we make a remark on the complexity of MLPD-CONSTR for a fixed value of l (see the last row in Table 1). We also make a distinction between connected and general (i.e., possibly disconnected) graphs. Our results are summarized in Table 1. Note that in all cases we fix either k or l (but not both). Observe that the case of $l = 1$ is trivial. Also, computing a minimum-length path decomposition of a graph of pathwidth $k = 1$ is trivial, since any such a graph G is the disjoint union of caterpillars (i.e., paths with pending edges) and isolated vertices; letting i denote the number of isolated vertices of G, the minimum length of a path decomposition of G is $|E(G)| + \lceil i/2 \rceil$. In order to prove these results, we deal with the problem MLPD(k) where k is fixed. We first show that MLPD(4) is NP-complete for general graph and then we conclude that this implies that MLPD(k) is NP-complete for all $k \ge 4$ for general graphs and for all $k \ge 5$ for connected graphs (Section 2).

In the remainder of the paper, we construct a polynomial-time algorithm for MLPD(3)-CONSTR. We begin by showing in Section 3 an algorithm for MLPD(k)-CONSTR, $k = 1, 2, 3$, for connected graphs. The algorithm recursively calls algorithms for MLPD(k')-CONSTR for each $k' < k$ for general (possibly disconnected) graphs. We prove that the algorithm for MLPD(k)-CONSTR is running in polynomial time provided that the algorithms for MLPD(k')-CONSTR for each $k' < k$ are all polynomial-time. There is a trivial algorithm for MLPD(0)-CONSTR.

To deal with disconnected graphs we extend this algorithm to the so-called chunk graphs in Section 4. A chunk graph has at most one 'big' connected component with three or more vertices and all its other connected components are either isolated vertices or isolated edges.

Finally, in Section 5 we show that MLPD(k)-CONSTR, $k = 1, 2, 3$, for disconnected G essentially reduces to MLPD(k)-CONSTR, $k = 1, 2, 3$, for chunk graphs of G. Though each chunk graph of G includes at most one big component of G, the isolated vertices and the isolated edges of G can be distributed between these big components to form chunk graphs of G in many different ways. We show how to obtain an optimal distribution and thus optimal decomposition of G into chunk graphs. Then, we show how to construct a solution to MLPD(k)-CONSTR, $k = 1, 2, 3$, for G from the solutions of MLPD(k)-CONSTR, $k = 1, 2, 3$, for the chunk graphs of G.

2. MLPD(k)-CONSTR is NP-hard for $k \ge 4$

In this section, we prove that the problem of finding a minimum-length path decomposition of width $k \ge 4$ is NP-hard. To that end, we first show that MLPD(4) is NP-complete, by presenting a pseudopolynomial-time transformation from the strongly NP-complete 3-PARTITION problem [21] to MLPD(4). The input to 3-PARTITION is an integer b and a set $S = \{w_1, \ldots, w_{3m}\}$ of positive integers, where $b/4 < w_i < b/2$ for all i. The answer to 3-PARTITION is YES if and only if there exists a partition of the set $\{1, \ldots, 3m\}$ into m sets $S_1, \ldots, S_m$, each of size 3, such that $\sum_{i \in S_j} w_i = b$ for each $j = 1, \ldots, m$.

Given an instance of 3-PARTITION, we construct a disconnected graph $G(S, b)$ in a few steps. In what follows, m will always denote the number of required parts of the partition, i.e., $m = |S|/3$.

First, for each $i \in \{1, \ldots, 3m\}$, we construct a connected graph H_i as follows. Take w_i copies of K_3, denoted by $K_3^{i,q}$, $q = 1, \ldots, w_i$, and $w_i - 1$ copies of K_4, denoted by $K_4^{i,q}$, $q = 1, \ldots, w_i - 1$. (The copies are taken to be mutually disjoint.) Then, for each $q = 1, \ldots, w_i - 1$, we identify two different vertices of $K_4^{i,q}$ with a vertex of $K_3^{i,q}$ and with a vertex of $K_3^{i,q+1}$, respectively. This is done in such a way that each vertex of each $K_3^{i,q}$ is identified with at most one vertex from other cliques. Thus, in the resulting graph H_i, each clique shares a vertex with at most two other cliques. Informally, the cliques form a 'chain' in which the cliques of size 3 and 4 alternate. See Fig. 1(a) for an example of H_i where $w_i = 3$.

MOST WIEDZY Downloaded from mostwiedzy.pl

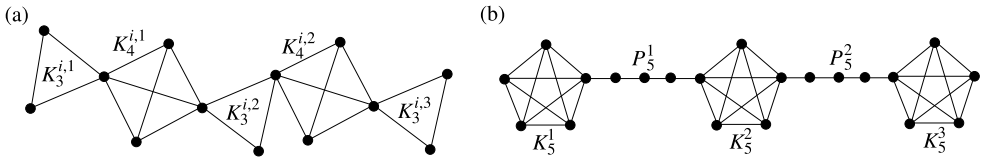

Fig. 1. (a) H_i, where $w_i = 3$; (b) $H_{2,4}$.

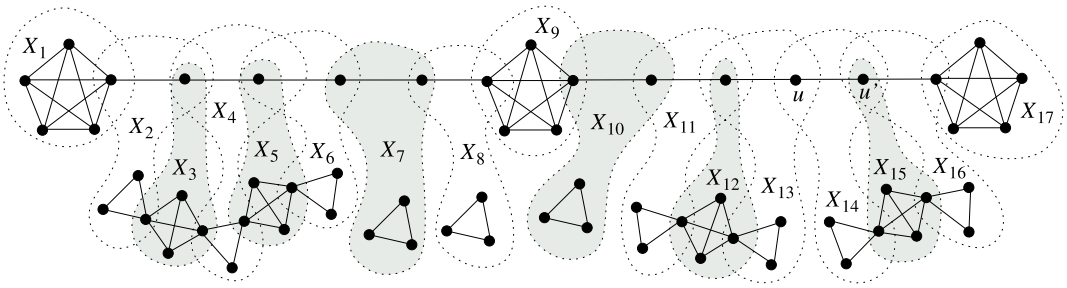

Fig. 2. $G((1,1,1,2,2,3),5)$ together with $\mathcal{P}$ of width 4 and length 17 ($m = 2, b = 5$).

Second, we construct a graph $H_{m,b}$ as follows. Take $m+1$ copies of K_5, denoted by $K_5^1, \ldots, K_5^{m+1}$, and m copies of the path P_b of length b (P_b has b edges and $b+1$ vertices), denoted by $P_b^1, \ldots, P_b^m$. (Again, the copies are taken to be mutually disjoint.) Now, for each $j = 1, \ldots, m$, identify one endpoint of P_b^j with a vertex of K_5^j, and identify the other endpoint with a vertex of K_5^{j+1}. Moreover, do this in a way that ensures that, for each j, no vertex of K_5^j is identified with the endpoints of two different paths. See Fig. 1(b) for an example of $H_{2,4}$.

Let $G(S,b)$ be the graph obtained by taking the disjoint union of the graphs $H_1, \ldots, H_{3m}$ and the graph $H_{m,b}$. The input to the MLPD(4) problem is the graph $G(S,b)$ and the integer $l = 1 - 2m + 2mb$.

The subgraphs $K_3^{i,q}$, $K_4^{i,q}$ are called the *cliques of* H_i, and the K_5^j are called the cliques of $H_{m,b}$. For brevity, all these cliques are called the *cliques of* $G(S,b)$. Similarly, $P_b^1, \ldots, P_b^m$ are called the *paths of* $G(S,b)$. Observe that the number of cliques of $G(S,b)$ is exactly l. If $\mathcal{P}$ is a path decomposition of $G(S,b)$ and a bag of $\mathcal{P}$ contains all vertices of a clique of $G(S,b)$, then we say that the bag *contains* this clique.

Before we continue, we illustrate the construction of the path decomposition in the proof of Theorem 2.1.

Example. Let $S = (1,1,1,2,2,3)$ (so, $m = 2$), $b = 5$.[2] A solution to this instance of 3-PARTITION is $S_1 = \{1,2,6\}$, and $S_2 = \{3,4,5\}$ (clearly $w_1 + w_2 + w_6 = w_3 + w_4 + w_5 = 5$). The graph $H_{S,b}$, and the corresponding path decomposition $\mathcal{P}$ constructed by the algorithm from the proof of **(i)** are given in Fig. 2 (the gray color is used for some bags only to make it easier to distinguish the particular bags of this decomposition).

We now state and prove Theorem 2.1.

Theorem 2.1. *The problem* MLPD(4) *is* NP-*complete.*

Proof. We will prove the theorem by proving that the answer to 3-PARTITION is YES for S and b, if and only if the answer to MLPD(4) is YES for $G(S,b)$ and $l = 1 - 2m + 2mb$. First, we prove that if there exists a solution to 3-PARTITION for the given S and b, then there exists a path decomposition $\mathcal{P}$ of $G(S,b)$ such that $\text{width}(\mathcal{P}) \le 4$ and $\text{len}(\mathcal{P}) = l$.

(i) *If the answer to* 3-PARTITION *is* YES *for S and b, then the answer to* MLPD(4) *is* YES *for $G(S,b)$ and $l = 1 - 2m + 2mb$.*
Let $S_1, \ldots, S_m$ be a solution to 3-PARTITION. We say that a vertex of P_b^j, $j = 1, \ldots, m$, at distance $d-1$ from the endpoint of P_b^j identified with a vertex of K_5^j is the *d-th vertex of* P_b^j. We construct a path decomposition $\mathcal{P}$ as follows.
- ▹ STEP 1. Let $\mathcal{P}$ be initially the empty list.
- ▹ STEP 2. For each $j = 1, \ldots, m$ do the following:
  - ▹ STEP 2.1. Append $V(K_5^j)$ to $\mathcal{P}$, and set $p := 0$.
  - ▹ STEP 2.2. For each $i \in S_j$ do the following:

[2] Note that we drop the assumption that $b/4 < w_i < b/2$ for all i in this example.

MOST WIEDZY Downloaded from mostwiedzy.pl

▷ STEP 2.1(A). For each $q = 1, \ldots, w_i$, first append $V(K_3^{i,q}) \cup \{u, v\}$ to $\mathcal{P}$, and if $q < w_i$, then also append $V(K_4^{i,q}) \cup \{v\}$ to $\mathcal{P}$, where u and v are the $(p+q)$-th and $(p+q+1)$-st vertices of P_b^j, respectively.

▷ STEP 2.1(B). Set $p := p + w_i$.

▷ STEP 3. Append $V(K_5^{m+1})$ to $\mathcal{P}$.

See Fig. 2 for an example of this construction. (Informally, the triangles $K_3^{i,q}$ are placed together with edges of P_b^j into common bags of $\mathcal{P}$.) It can easily be checked that, at the end of this algorithm, $\text{len}(\mathcal{P}) = m + 1 + \sum_{j=1}^{m} \sum_{i \in S_j} (2w_i - 1) = m + 1 + 2mb - 3m = l$ and, hence, $\mathcal{P}$ consists of l bags. Moreover, each bag has size 5. We leave it to the reader to verify that $\mathcal{P}$ is a path decomposition of $G(S, b)$.

Before proving the reverse implication we need a few additional claims.

(ii) *If $\mathcal{P}$ is a path decomposition of $G(S, b)$ of width 4 and length $l = 1 - 2m + 2mb$, then each bag of $\mathcal{P}$ contains exactly one clique of $G(S, b)$.*
Each clique of $G(S, b)$ has size at least 3. Moreover, any two cliques of $G(S, b)$ share at most one vertex, and no two cliques of size 3 share a vertex. Thus, each bag of $\mathcal{P} = (X_1, \ldots, X_l)$ contains at most one clique of $G(S, b)$. However, it follows immediately from **(PD1)**–**(PD3)** that for every clique K of $G(S, b)$, there exists $i \in \{1, \ldots, l\}$ such that $V(K) \subseteq X_i$. Thus, since l equals the number of cliques of $G(S, b)$, each bag of $\mathcal{P}$ must contain exactly one clique of $G(S, b)$.

We now show that we may assume without loss of generality that in a path decomposition of width 4 of $G(S, b)$, the cliques $K_5^1, \ldots, K_5^{m+1}$ appear in this order in the bags of the path decomposition.

(iii) *Let $\mathcal{P} = (X_1, \ldots, X_l)$ be a path decomposition of width 4 of $G(S, b)$ and let $c_1, \ldots, c_{m+1}$ be selected so that X_{c_i} contains K_5^i for each $i = 1, \ldots, m+1$. Then, $c_1 < c_2 < \cdots < c_{m+1}$ or $c_1 > c_2 > \cdots > c_{m+1}$.*
Suppose for a contradiction that the lemma does not hold. Thus, there exist t_1, t_2, t_3, $1 \le t_1 < t_2 < t_3 \le l$, such that X_{t_i} contains $K_5^{j_i}$, $i = 1, 2, 3$, where neither $j_1 < j_2 < j_3$ nor $j_1 > j_2 > j_3$. Consider the case when $j_2 < j_1 < j_3$ – the other cases are analogous. Take a shortest path P between a vertex of $K_5^{j_1}$ and a vertex of $K_5^{j_3}$. Since $j_2 < j_1 < j_3$, $V(P)$ and $V(K_5^{j_2})$ are disjoint. By Observation 1.2, there exists $v \in V(P) \cap X_{t_2}$. Thus, X_{t_2} contains both v and $V(K_5^{j_2})$, contrary to the fact that $|X_{t_2}| \le \text{width}(\mathcal{P}) + 1 = 5$.

Moreover, the bags with the vertices of each H_i form an interval of $\mathcal{P}$ that falls between two cliques of $H_{m,b}$.

(iv) *If $\mathcal{P}$ is a path decomposition of width 4 and length l of $G(S, b)$, then for each $i \in \{1, \ldots, 3m\}$ there exist s and t ($1 \le s < t \le l$) such that $V(H_i) \subseteq X_s \cup \cdots \cup X_t$, $V(H_i) \cap X_p \ne \emptyset$ for each $p = s, \ldots, t$, and no clique of $H_{m,b}$ is contained in any of the bags $X_s, \ldots, X_t$.*
This follows from Observation 1.2, and $\text{width}(\mathcal{P}) = 4$ and $V(H_i) \cap V(H_{m,b}) = \emptyset$ for each $i = 1, \ldots, 3m$.

We are now ready to prove the reverse of **(i)**.

(v) *If the answer to* MLPD(4) *is* YES *for $G(S, b)$ and $l = 1 - 2m + 2mb$, then the answer to* 3-PARTITION *is* YES *for S and b.*
Let $\mathcal{P} = (X_1, \ldots, X_l)$ be a path decomposition of width 4 and of length l of $G(S, b)$. By **(ii)**, each bag of $\mathcal{P}$ contains exactly one clique of $G(S, b)$. Since each clique of $G(S, b)$ has size at least 3 and $\text{width}(\mathcal{P}) = 4$, no bag of $\mathcal{P}$ contains the endpoints of two or more edges of a path P_b^j, $j = 1, \ldots, m$, and the endpoints of any edge of P_b^j can only share a bag with some clique $K_3^{i,q}$, $i \in \{1, \ldots, 3m\}$, $q \in \{1, \ldots, w_i\}$. Moreover, the total number of edges of the paths of $G(S, b)$ equals mb (mb is also the number of cliques $K_3^{i,q}$), which implies that the endpoints of each edge of each path of $G(S, b)$ share a bag with a unique $K_3^{i,q}$ for some $i \in \{1, \ldots, 3m\}$ and $q \in \{1, \ldots, w_i\}$.
Let $X_{c_j} = V(K_5^j)$ for each $j = 1, \ldots, m+1$. By **(iii)**, $c_1 < c_2 < \cdots < c_{j+1}$ or $c_1 > c_2 > \cdots > c_{j+1}$. Since $(X_l, \ldots, X_1)$ is a path decomposition of $G(S, b)$, we may assume without loss of generality that the former occurs. Then, the endpoints of all b edges of a path P_b^j, $j = 1, \ldots, m$, must be included in the bags $X_{c_j+1}, \ldots, X_{c_{j+1}-1}$, because otherwise the connectedness of P_b^j and Observation 1.2 would imply a vertex of P_b^j in either X_{c_j} or $X_{c_{j+1}}$ or both, which results in a path decomposition of width at least 5, a contradiction. Therefore, exactly b cliques in $\{K_3^{i,q} \mid i \in \{1, \ldots, 3m\}, q \in \{1, \ldots, w_i\}\}$ must be included in the bags $X_{c_j+1}, \ldots, X_{c_{j+1}-1}$. Moreover, **(iv)** implies that for each $i \in \{1, \ldots, 3m\}$ there exists $j \in \{1, \ldots, m\}$ such that $V(H_i) \subseteq X_{c_j+1} \cup \cdots \cup X_{c_{j+1}-1}$. Define for each $j = 1, \ldots, m$ $S_j = \{i \in \{1, \ldots, 3m\} \mid V(H_i) \subseteq X_{c_j+1} \cup \cdots \cup X_{c_{j+1}-1}\}$. Due to the above arguments, $\sum_{i \in S_j} w_i = b$ for each $j = 1, \ldots, m$. Therefore, the answer to 3-PARTITION is YES.

The theorem now follows from **(i)** and **(v)**. □

MOST WIEDZY Downloaded from mostwiedzy.pl

We have shown so far that MLPD(4) is NP-complete. Let $k \geq 1$. Observe that a disconnected graph G has a path decomposition of width $k-1$ and length l if and only if the connected graph obtained from G by adding a universal vertex has a path decomposition of width k and length l. It follows that if MLPD$(k-1)$ is NP-complete, then MLPD(k) remains NP-complete even when the input is restricted to connected graphs. This observation, together with Theorem 2.1, immediately implies the following theorem:

Theorem 2.2. *The problem* MLPD(k) *is* NP-*complete for each* $k \geq 4$. *Moreover, for* $k \geq 5$, *the problem* MLPD(k) *remains* NP-*complete when the input is restricted to connected graphs.* □

We conclude this section with a remark on the complexity of MLPD when the input consists of the graph G and the pathwidth $k \geq 1$, but the length parameter l is fixed. The following result, Corollary 2.3, is a direct consequence of the NP-completeness of the vertex separator problem defined in [25]. The input to this vertex separator problem is a simple graph G and an integer k. The problem is to determine whether there exist sets X and Y that satisfy conditions **(PD1)** and **(PD2)** of a path decomposition (i.e., they cover all vertices and edges of the input graph) and $\max(|X|, |Y|) \leq k$. This can be directly reformulated as the problem of finding a path decomposition of length 2 and width at most k. (Note that the terminology used in [25] is quite different from the terminology used in this paper.)

Corollary 2.3. *The problem* MLPD *with* $l \geq 2$ *fixed is* NP-*complete.* □

3. MLPD(k)-CONSTR for connected graphs, $k \leq 3$

Section 2 dealt with the entries marked with "NP-hard" in Table 1. In the remainder of this paper, we will prove the "polynomial-time" entries in the table. Therefore, from now on, all path decompositions that we deal with have width at most 3. The main result of this section is the following theorem.[3]

Theorem 3.1. *Let* $k \in \{1, 2, 3\}$.

If: *for each* $k' \in \{0, \ldots, k-1\}$, *there exists a polynomial-time algorithm that, for any graph* G *either constructs a minimum-length path decomposition of width* k' *of* G, *or concludes that no such path decomposition exists,*

then: *there exists a polynomial-time algorithm that, for any* ***connected*** *graph* G, *either constructs a minimum-length path decomposition of width* k *of* G, *or concludes that no such path decomposition exists.*

We take several steps to prove this theorem. In Section 3.1 we formulate an algorithm that outlines the main idea of our method, but whose running time is not necessarily polynomial. This algorithm constructs a directed graph $\mathcal{G}_k$ whose directed paths leading from its source s to its sink t correspond to path decompositions of width k of G. Moreover, the length of a directed s–t path in $\mathcal{G}_k$ equals the length of the corresponding path decomposition of G. Hence, our problem reduces to computing a shortest path in $\mathcal{G}_k$. The running time of this algorithm is, in general, not polynomial since the size of $\mathcal{G}_k$ may be exponential in the size of G. Hence, the remainder of Section 3 is devoted to providing a different construction of $\mathcal{G}_k$ that preserves the above-mentioned relation between the shortest paths in $\mathcal{G}_k$ and the path decompositions of G, and ensures that the size of $\mathcal{G}_k$ is polynomial in the size of G. To that end we develop some notation and obtain several properties of minimum-length path decomposition of width at most 3 of a connected graph (Sections 3.3 and 3.4). Finally, Section 3.5 provides the polynomial-time algorithm and proves its correctness. Our proof of Theorem 3.1 is constructive provided that the algorithms from the 'if' part of this theorem exist. We deal with the latter in Sections 4 and 5.

3.1. A generic (non-polynomial) algorithm

Let G be a graph. We say that $\mathcal{P} = (X_1, \ldots, X_l)$ is a *partial path decomposition* of G if $\mathcal{P}$ is a path decomposition of $G\left[\bigcup_{i=1}^{l} X_i\right]$. Define $\text{span}(\mathcal{P}) = \bigcup_{i=1}^{l} X_i$ to be the *span* of $\mathcal{P}$ and denote $G_{\mathcal{P}} = G[\text{span}(\mathcal{P})]$. $G_{\mathcal{P}}$ is called the subgraph of G *covered by* $\mathcal{P}$.

It follows that $\mathcal{P}$ is a path decomposition of the induced subgraph $G_{\mathcal{P}}$ and $V(G_{\mathcal{P}}) = \text{span}(\mathcal{P})$. Notice that $G_{\mathcal{P}} = G$ if and only if $\mathcal{P}$ is a path decomposition of G. Also note that any prefix of a path decomposition of G is a partial path decomposition of G.

We say that a partial path decomposition $\mathcal{P} = (X_1, \ldots, X_l)$ *extends to* a partial path decomposition $\mathcal{P}' = (X'_1, \ldots, X'_{l'})$, with $l' \geq l$, if $X_i = X'_i$ for all $i \in \{1, \ldots, l\}$. We define the *frontier* of $\mathcal{P}$ to be $\delta(\mathcal{P}) = \{x \in V(G_{\mathcal{P}}) \mid x \text{ has a neighbor in } V(G) \setminus V(G_{\mathcal{P}})\}$.

Consider the following generic and potentially exponential-time algorithm for finding a minimum-length path decomposition of width at most k for a given graph G. We construct an auxiliary directed graph $\mathcal{G}_k$ whose vertices are pairs (F, X),

[3] We recall that the case $k = 1$ can be solved directly.

MOST WIEDZY Downloaded from mostwiedzy.pl

where F is an induced subgraph of G, $X \subseteq V(F)$, and $|X| \leq k+1$. Each pair (F, X) represents the (perhaps empty) collection $\mathbf{P}(F, X)$ of all partial path decompositions $\mathcal{P} = (X_1, \ldots, X_{\mathrm{len}(\mathcal{P})})$ of width at most k that have the common property that $G_{\mathcal{P}} = F$ and $X_{\mathrm{len}(\mathcal{P})} = X$ (i.e., the subgraph of G covered by $\mathcal{P}$ is F and the last bag of $\mathcal{P}$ is X). Notice that the partial path decompositions within $\mathbf{P}(F, X)$ may be of different lengths. There is an arc from (F, X) to (F', X') in $\mathcal{G}_k$ if and only if *every* partial path composition in $\mathbf{P}(F, X)$ extends to *some* partial path decomposition in $\mathbf{P}(F', X')$ by adding exactly one bag, namely X'. We will also add to $\mathcal{G}_k$ a special source vertex s and a sink vertex t. There is an arc from the source vertex s to every pair (F, X) with $F = G[X]$, and an arc from every pair (F, X) with $F = G$ to the sink vertex. For convenience, let $\mathbf{P}(s)$ contain the path decomposition of length 0. We have the following result:

Lemma 3.2. *It holds $\mathcal{P} \in \mathbf{P}(G, X)$ for some $X \subseteq V(G)$, where $\mathrm{len}(\mathcal{P}) \leq l$, if and only if there exists a directed s–t path in $\mathcal{G}_k$ of length at most $l+1$.*

Proof. We argue by induction on l that there exists a directed s–v path of length l in $\mathcal{G}_k$, where $v = (F, X)$ is a vertex of $\mathcal{G}_k$, if and only if a partial path decomposition of length l belongs to $\mathbf{P}(F, X)$. In the base case of $l = 0$ we have that $v = s$ and the claim follows directly from the definition of $\mathbf{P}(s)$. Suppose now that the induction hypothesis holds for some $l \geq 0$.

Let P' be an s–v' path of length $l+1$ in $\mathcal{G}_k$. Let (v, v') be the last arc of P'. Let P be the path P' without (v, v'). Thus, P is an s–v path of length l in $\mathcal{G}_k$. By the induction hypothesis, there exists a partial path decomposition $\mathcal{P} \in \mathbf{P}(v)$ of length l. Since $(v, v') \in E(\mathcal{G}_k)$, $\mathcal{P}$ extends to some partial path decomposition $\mathcal{P}' \in \mathbf{P}(F, X)$ by adding exactly one bag, namely X. Thus $\mathrm{len}(\mathcal{P}') = \mathrm{len}(\mathcal{P}) + 1$ as required.

Suppose now that $\mathcal{P}' \in \mathbf{P}(F, X)$ and $\mathrm{len}(\mathcal{P}') = l+1$. Let $\mathcal{P}' = (X_1, \ldots, X_{l+1})$. By definition of $\mathcal{G}_k$, $F = G_{\mathcal{P}'}$ and $X_{l+1} = X$. Thus, $v = (G_{\mathcal{P}'}, X_{l+1})$. Now, let $\mathcal{P} = (X_1, \ldots, X_l)$ and $v = (G_{\mathcal{P}}, X_l)$. Since every partial decomposition in $\mathbf{P}(G_{\mathcal{P}}, X_l)$ extends to some partial decomposition in $\mathbf{P}(G_{\mathcal{P}'}, X_{l+1})$ by adding X_{l+1}, $(v, v') \in E(\mathcal{G}_k)$. By the induction hypothesis, there exists an s–v path P in $\mathcal{G}_k$ of length l. Then, P together with the arc (v, v') forms the desired s–v' path of length $l+1$ in $\mathcal{G}_k$. □

Having constructed $\mathcal{G}_k$, we find a shortest path from s to t in $\mathcal{G}_k$. Let s, $(F_1, X_1), \ldots, (F_l, X_l)$, and t be the consecutive vertices of this path. Then, by Lemma 3.2, this path corresponds to a path decomposition $(X_1, \ldots, X_l)$ of G. This clearly gives an exponential-time algorithm for finding a minimum-length path decomposition. In the reminder of this section, we redefine the graph $\mathcal{G}_k$ to reduce its size for connected G and $k \in \{1, 2, 3\}$. Note that a series of path decompositions that correspond to consecutive vertices of the shortest path resembles the concept of *crusades* introduced by Bienstock and Seymour in [4].

3.2. From generic algorithm to polynomial running time: an informal description

The aim of this section is to introduce some intuition on the construction of $\mathcal{G}_k$ whose size is bounded by a polynomial in the size of G. The formal definition of $\mathcal{G}_k$ is given in Section 3.5.

Our approach is to represent the pairs (F, X) in an alternative way. We encode the graph $\mathcal{G}_k$ in such a way that for a fixed set $X \subseteq V(G)$ the number of vertices of $\mathcal{G}_k$ of the form (F, X) is polynomially bounded. Since $|X| \leq k+1$ and k is fixed, this reduces the number of vertices to a number bounded by a polynomial in the size of G. As a result, however, some path decompositions of G no longer have corresponding s–t paths in $\mathcal{G}_k$ though we prove that minimum-length path decompositions of G still have corresponding shortest paths in $\mathcal{G}_k$. The alternative encoding is as follows.

We represent a pair (F, X) by a pair $(X, R(X))$, where $R(X)$ is a function that maps each non-empty subset $S \subseteq X$ into a triple $(\mathbf{G}, f, l)$ with the following properties:

(1) $\mathbf{G}$ is a set with at least $\max\left\{|\mathcal{C}_G^*(S)| - 12, 0\right\}$ components in $\mathcal{C}_G^*(S)$,
(2) $f: \mathcal{C}_G^*(S) \to \{0, 1\}$ is a function such that $f(H) = f(H')$ for all $H, H' \in \mathbf{G}$, and $f(H) = 1$ implies that $V(H) \subseteq V(F)$,
(3) $l = \left|\bigcup_{C \in \mathcal{C}_G^1(S)} V(C) \cap V(F)\right|$, i.e., l equals the number of single-vertex components in $\mathcal{C}_G^1(S)$ that are covered by any partial path decomposition in $\mathbf{P}(F, X)$.

We will show that it is possible to reconstruct (F, X) from $(X, R(X))$. The vertex set of our final auxiliary graph $\mathcal{G}_k$ consists of all pairs $(X, R(X))$. Note that not every pair (F, X) can be represented by a pair $(X, R(X))$ satisfying properties (1)–(3). However, it turns out that it suffices to consider only the pairs (F, X) that can in fact be represented in this way. We will see (in the proof of Theorem 3.1) that the number of such pairs $(X, R(X))$ is bounded by a polynomial in the size of G. The arcs of $\mathcal{G}_k$ will have weights and, informally speaking, the weight of an arc (v, v') equals the number of bags that are added while extending any partial path decomposition in $\mathbf{P}(v)$ to a partial path decomposition in $\mathbf{P}(v')$. Hence, unlike in the generic construction of $\mathcal{G}_k$, some arcs in the new directed graph introduce several bags of a path decomposition. We will show that the vertices that we drop from the generic graph are irrelevant for the length minimization, and that $\mathcal{G}_k$ has the property that there exists a clean path decomposition $\mathcal{P}$ with $\mathrm{len}(\mathcal{P}) \leq l$ if and only if there exists an s–t path in $\mathcal{G}_k$ of length at most l. (See Lemma 3.21 and Lemma 3.22.) The clean path decompositions are defined in Section 3.4 where we also observe that among all minimum-length path decompositions of G there always exists one that is clean.

MOST WIEDZY Downloaded from mostwiedzy.pl

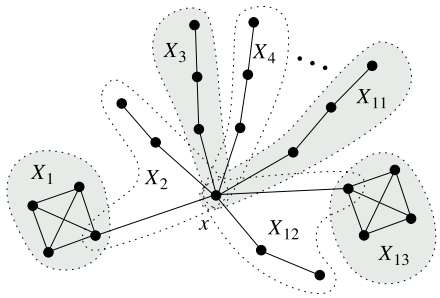

Fig. 3. Illustration of a bottleneck set $S = \{x\}$ and the corresponding S-branches.

3.3. Bottleneck sets and bottleneck intervals

In this section we consider a path decomposition $\mathcal{P} = (X_1, \ldots, X_l)$ of a connected graph G, and a fixed set $S \subseteq V(G)$. Recall that $\mathcal{C}_G(S)$ is the collection of connected components H of $G - S$ such that every vertex in S has a neighbor in $V(H)$. We call the components in $\mathcal{C}^*_G(S)$ *S-branches*, while the vertices of the graphs in $\mathcal{C}^1_G(S)$ are called *S-leaves*. Finally, let $|\mathcal{C}^*_G(S)| = c$.

We begin by investigating the relative order in which the vertices of S and S-branches appear in, and disappear from, the bags in $\mathcal{P} = (X_1, \ldots, X_l)$. For a (connected or disconnected) subgraph H of G, define

$$\alpha_{\mathcal{P}}(H) = \min\{i \mid X_i \cap V(H) \neq \emptyset\} \quad \text{and} \quad \beta_{\mathcal{P}}(H) = \max\{i \mid X_i \cap V(H) \neq \emptyset\}.$$

For $v \in V(G)$, we abbreviate $\alpha_{\mathcal{P}}(G[\{v\}])$ and $\beta_{\mathcal{P}}(G[\{v\}])$ as $\alpha_{\mathcal{P}}(v)$ and $\beta_{\mathcal{P}}(v)$, respectively. (Note that these coincide with the definitions of first(v) and last(v) in [12].) For convenience we define

$$\widehat{\alpha}_{\mathcal{P}}(S) = \max_{v \in S}\{\alpha_{\mathcal{P}}(v)\} \quad \text{and} \quad \widehat{\beta}_{\mathcal{P}}(S) = \min_{v \in S}\{\beta_{\mathcal{P}}(v)\},$$

for any non-empty $S \subseteq V(H)$. Hence, $S \subseteq X_i$ if and only if $i \in \{\widehat{\alpha}_{\mathcal{P}}(S), \ldots, \widehat{\beta}_{\mathcal{P}}(S)\}$. Whenever $\mathcal{P}$ is clear from the context, we drop it as a subscript. Clearly, we have that

$$V(H) \subseteq X_{\alpha(H)} \cup \cdots \cup X_{\beta(H)}. \tag{1}$$

Informally, $X_{\alpha(H)}$ can be interpreted as the first bag that contains a vertex of $V(H)$, and $\alpha(H)$ as the start of H. Similarly, $X_{\beta(H)}$ is the last bag that contains a vertex of $V(H)$, and the $\beta(H)$ as the completion of H. By Observation 1.2, if H is connected, then any bag between these first and last bags must contain a vertex of $V(H)$. By definition, the converse is also true: no bag X_i with i outside the interval $[\alpha(H), \beta(H)]$ contains a vertex of $V(H)$. The following lemma relates the start and the completion of an S-branch H and the start and completion of $x \in S$.

Lemma 3.3. *Let G be a graph and let $\mathcal{P} = (X_1, \ldots, X_l)$ be a path decomposition of G. Let $S \subseteq V(G)$, let $x \in S$ and let H be an S-branch. Then, the following statements hold:*

(i) *If $\alpha(x) \leq \alpha(H)$, then $x \in X_i$ for all $\alpha(x) \leq i \leq \alpha(H)$;*
(ii) *$\alpha(x) \leq \beta(H)$;*
(iii) *If $\beta(x) \geq \beta(H)$, then $x \in X_i$ for all $\beta(H) \leq i \leq \beta(x)$;*
(iv) *$\alpha(H) \leq \beta(x)$.*

Proof. By definition of an S-branch, there exists $v \in V(H)$ that is adjacent to x. Since $\{v, x\} \in E(G)$, it follows from **(PD2)** that there exists t such that $\{v, x\} \subseteq X_t$. Clearly, $\alpha(H) \leq t$. To prove (i), note that the assumption $\alpha(x) \leq \alpha(H)$ implies that $\alpha(x) \leq t$. Since $x \in X_{\alpha(x)}$ and $x \in X_t$, it follows from **(PD3′)** that $x \in X_i$ for all $\alpha(x) \leq i \leq t$. In particular, $x \in X_i$ for all $\alpha(x) \leq i \leq \alpha(H)$, as required. To prove (ii), observe that $x \in X_t$ implies that $\alpha(x) \leq t$, and $v \in X_t$ implies that $t \leq \beta(H)$. Thus, $\alpha(x) \leq \beta(H)$.

Parts (iii) and (iv) follow from (i) and (ii) applied to $\mathcal{P}' = (X_l, \ldots, X_1)$. □

Some S-branches may start even before the whole of S appears in the bags of $\mathcal{P}$; see X_1 in Fig. 3. Also, some S-branches may complete even after the whole of S no longer appears in the bags of $\mathcal{P}$; see X_{13} in Fig. 3. We now show that, in either case, this can only happen for a few S-branches whose number is limited by k. To that end, we adopt the convention that $H_1, \ldots, H_c$ denote the S-branches in $\mathcal{C}^*_G(S)$ 'start-ordered', i.e., ordered so that $\alpha(H_1) \leq \alpha(H_2) \leq \cdots \leq \alpha(H_c)$, and $H^1, \ldots, H^c$ denote these S-branches 'completion-ordered', i.e., ordered so that $\beta(H^1) \leq \beta(H^2) \leq \cdots \leq \beta(H^c)$. We have the following lemma.

MOST WIEDZY Downloaded from mostwiedzy.pl

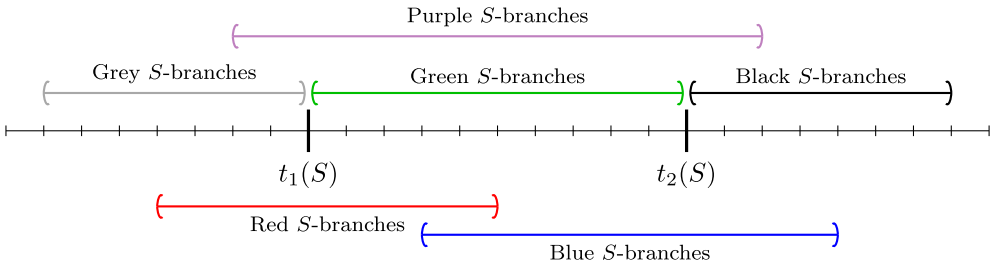

Fig. 4. Coloring of S-branches. Note that red and blue branches may or may not overlap.

Lemma 3.4. *Let G be a graph, let $\mathcal{P} = (X_1, \ldots, X_l)$ be a path decomposition of width k of G, and let $S \subseteq V(G)$, where $c > k$. Then, the following two statements hold:*

(i) $\alpha(H_i) \geq \alpha(x)$ *for all* $i \geq k+1$ *and all* $x \in S$,
(ii) $\beta(H^i) \leq \beta(x)$ *for all* $i \leq c-k$ *and all* $x \in S$.

Proof. For (i), since $\alpha(H_1) \leq \alpha(H_2) \leq \cdots \leq \alpha(H_c)$, it suffices to show that $\alpha(H_{k+1}) \geq \alpha(x)$. So suppose for a contradiction that $\alpha(H_{k+1}) < \alpha(x)$. Hence, $\alpha(H_j) < \alpha(x)$ for all $j = 1, \ldots, k+1$. It follows from Lemma 3.3(ii) that, for each $j = 1, \ldots, k+1$, there exists $v_j \in V(H_j) \cap X_{\alpha(x)}$. In particular, it follows that $\{v_1, \ldots, v_{k+1}, x\} \subseteq X_{\alpha(x)}$, implying that $|X_{\alpha(x)}| > k+1$, contrary to the fact that $\mathcal{P}$ has width k. This proves (i). Next, (ii) follows from (i) applied to $\mathcal{P}' = (X_l, \ldots, X_1)$. □

We now focus on S-branches H such that $\alpha(x) < \alpha(H) \leq \beta(H) < \beta(x)$ for all $x \in S$, since by Lemma 3.4 there is only a constant number of branches that do *not* meet this condition for fixed k. As we cannot guarantee the existence of these S-branches for S with small $c = |\mathcal{C}^*_G(S)|$, we limit ourselves to special sets S referred to as bottlenecks.

A set $S \subseteq V(G)$ is a *bottleneck set* if $S \neq \emptyset$ and $c \geq 13$. We denote by $\mathcal{S}$ the collection of all bottleneck sets of G. Note that if $\mathrm{pw}(G) \leq 1$, then G has no bottleneck sets. Also note that, if $\mathrm{pw}(G) = 2$, then for any bottleneck S of size 1, all components in $\mathcal{C}_G(S)$, except possibly two, are paths whose endpoints are adjacent to the vertex in S.

Example. To illustrate this concept, consider the graph G in Fig. 3. The sets $X_1, \ldots, X_{13}$ form a path decomposition of G. The set $S = \{x\}$ is a bottleneck set. There are 13 S-branches, namely, the connected components of $G - \{x\}$. Notice that $G - \{x\}$ has no components consisting of exactly one vertex, and therefore there are no S-leaves.

The bottleneck sets are a key for a couple more reasons. First, if G has no bottleneck set, then the size of the auxiliary generic graph from Section 3.1 can be easily bounded by a polynomial in the size of G since the number of S-branches is then bounded by a constant for any S. On the other hand, if G contains even a single bottleneck set S, then the number of vertices (F, X) such that $S \subseteq X$ in $\mathcal{G}_k$ can be exponential. This follows from an observation that the number of induced subgraphs F with $S \subseteq \delta_G(F)$ is exponential in $|\mathcal{C}^*_G(S)|$. However, we prove that all, except a constant number, S-branches in $\mathcal{C}^*_G(S)$ are in consecutive bags $X_i, \ldots, X_j$ such that $S \subseteq X_p$ for each $p = i-1, \ldots, j+1$. We refer to the interval between i and j as the *bottleneck interval* of S and formally define it later. Since the S-branches in the bottleneck interval of S always share bags with the whole S we can recursively reduce the computation of a minimum-length path decomposition of width k of G to the computation of a minimum-length path decomposition of width $k - |S|$ for the branches in the bottleneck interval of S. This is another key reason behind the bottleneck sets. Also observe that $|S| \leq k$. Hence, the number of bottlenecks is bounded by a polynomial in the size of G.

For any bottleneck set S and a path decomposition $\mathcal{P}$, we define $I_{\mathcal{P}}(S) = \{t_1(S), \ldots, t_2(S)\}$, where $t_1(S)$, the starting time, and $t_2(S)$, the ending time, are as follows:

$$t_1(S) = \min\left\{\alpha(H) \;\middle|\; H \text{ is an } S\text{-branch and } S \subseteq X_{\alpha(H)} \cap X_{\alpha(H)-1}\right\},$$
$$t_2(S) = \max\left\{\beta(H) \;\middle|\; H \text{ is an } S\text{-branch and } S \subseteq X_{\beta(H)} \cap X_{\beta(H)+1}\right\}.$$

We call $I_{\mathcal{P}}(S)$ the *bottleneck interval* associated with the set S. Informally, let $X_i, \ldots, X_j$ be all bags in $\mathcal{P}$ such that each of them contains S. Then, $t_1(S)$ is the start of the earliest S-branch to start in $\{i+1, \ldots, j-1\}$ and $t_2(S)$ is the completion of the latest S-branch to complete in $\{i+1, \ldots, j-1\}$. For example, in Fig. 3, we have $t_1(S) = 3$ and $t_2(S) = 11$ for the bottleneck set $S = \{x\}$.

Notice that $I_{\mathcal{P}}(S)$ depends on the path decomposition $\mathcal{P}$. Whenever $\mathcal{P}$ is clear from the context we write $I(S)$ instead of $I_{\mathcal{P}}(S)$. We show in Lemma 3.5 that $t_1(S)$ and $t_2(S)$ are well defined and that $t_1(S) \leq t_2(S)$, which implies $I(S) \neq \emptyset$. Given the bottleneck interval $I(S)$, we color each S-branch H as follows: (see also Fig. 4)

MOST WIEDZY Downloaded from mostwiedzy.pl

- color H *green* if $t_1(S) \le \alpha(H) \le \beta(H) \le t_2(S)$;
- color H *red* if $\alpha(H) < t_1(S) \le \beta(H) \le t_2(S)$;
- color H *blue* if $t_1(S) \le \alpha(H) \le t_2(S) < \beta(H)$;
- color H *purple* if $\alpha(H) < t_1(S) \le t_2(S) < \beta(H)$;
- color H *gray* if $\beta(H) < t_1(S)$;
- color H *black* if $\alpha(H) > t_2(S)$.

There are exactly two gray S-branches, exactly two black branches, and the remaining branches are green for the bottleneck set $S = \{x\}$ in the graph G in Fig. 3.

Since $t_1(S) \le t_2(S)$ and $\alpha(H) \le \beta(H)$, each S-branch is assigned exactly one color. Notice also that there exist S-branches H and H' (possibly equal) such that $\alpha(H) = t_1(S)$ and $\beta(H') = t_2(S)$.

Lemma 3.5. *Let G be a connected graph, let $\mathcal{P} = (X_1, \ldots, X_l)$ be a path decomposition of width $k \le 3$ of G, and $S \in \mathcal{S}$ a bottleneck set. Then, $I(S)$ is well-defined and non-empty, and:*

(i) *there is at least one green S-branch;*
(ii) *the number of S-branches colored red, purple or gray is at most $2k$;*
(iii) *the number of S-branches colored blue, purple or black is at most $2k$.*

Proof. Let $H_1, H_2, \ldots, H_c$ be the S-branches start-ordered. Since S is a bottleneck set, we have $c \ge 13$. We first claim that

$$\alpha(G[S]) \le \alpha(H_q) \le \beta(G[S]) \text{ for all } q \in \{k+1, \ldots, c\}. \tag{2}$$

Let $q \in \{k+1, \ldots, c\}$ be selected arbitrarily. By Lemma 3.4(i), $\alpha(x) \le \alpha(H_q)$ for each $x \in S$. Thus, by Lemma 3.3(i), $x \in X_{\alpha(H_q)}$ for all $x \in S$. Hence, $S \subseteq X_{\alpha(H_q)}$, and (2) follows.

Second, we claim that

$$\alpha(G[S]) < \alpha(H_q) \le \beta(G[S]) \text{ for all } q \in \{2k+1, \ldots, c\}. \tag{3}$$

To prove this, it suffices to show that $\alpha(H_{2k+1}) > \alpha(G[S])$. Suppose otherwise, i.e., $\alpha(H_{2k+1}) \le \alpha(G[S])$. Since, by (2), $\alpha(H_{k+1}) \ge \alpha(G[S])$, we obtain that $\alpha(H_{k+1}) = \cdots = \alpha(H_{2k+1}) = \alpha(G[S])$. This implies that $X_{\alpha(G[S])}$ contains a vertex of each S-branch H_q with $q \in \{k+1, \ldots, 2k+1\}$. Thus, $|X_{\alpha(G[S])}| \ge |S| + k + 1$, contrary to the fact that $|X_{\alpha(G[S])}| \le k+1$. This proves (3).

By applying this argument to $\mathcal{P}' = (X_l, \ldots, X_1)$, we conclude that

$$\alpha(G[S]) \le \beta(H^q) < \beta(G[S]) \text{ for all } q \in \{1, \ldots, c-2k\}. \tag{4}$$

Let $\mathcal{A} = \mathcal{C}^*_G(S) \setminus (\{H_1, \ldots, H_{2k}\} \cup \{H^{c-2k+1}, \ldots, H^c\})$. By (3) and (4), $\alpha(G[S]) < \alpha(H) \le \beta(H) < \beta(G[S])$ for all $H \in \mathcal{A}$, and $|\mathcal{A}| \ge c - 4k \ge 1$. Thus, $t_1(S)$ and $t_2(S)$ are well-defined, and satisfy $t_1(S) \le \min\{\alpha(H) \colon H \in \mathcal{A}\}$ and $t_2(S) \ge \max\{\beta(H) \colon H \in \mathcal{A}\}$. It trivially follows that $t_1(S) \le t_2(S)$ and hence $I(S)$ is non-empty. For (i), notice that any $H \in \mathcal{A}$ receives the color green. Finally, (3) implies (ii), while (4) gives (iii). □

Example. Consider the graph G in Fig. 5. G has three bottleneck sets, namely $\{s_1\}$, $\{s_2\}$ and $\{s_1, s_2\}$. For the bottleneck $\{s_1\}$, we have $\{s_1\} \subseteq X_i$ for each $i = 4, \ldots, 38$. Then, $t_1(\{s_1\}) = 5$, because $\alpha(H_5) = 5$, and $t_2(\{s_1\}) = 37$, because $\beta(G[\{s_2\} \cup V(H_{13}) \cup \cdots \cup V(H_{37})]) = 37$. Thus, all $\{s_1\}$-branches are green except for $H_1, \ldots, H_4$, which are either gray (H_1, H_2, H_3) or black (H_4). For $\{s_1, s_2\}$ we have: $\{s_1, s_2\} \subseteq X_i$ for each $i = 13, \ldots, 37$, $t_1(\{s_1, s_2\}) = 14$ and $t_2(\{s_1, s_2\}) = 25$. The branch H_{13} is gray and the remaining $\{s_1, s_2\}$-branches, namely $H_{14}, \ldots, H_{25}$ are green. Thus, $I(\{s_1, s_2\}) \subseteq I(\{s_1\})$. Finally, $\{s_2\} \subseteq X_i$ for each $i = 13, \ldots, 37$, $t_1(\{s_2\}) = 26$ and $t_2(\{s_2\}) = 36$. Hence, $I(\{s_2\}) \subseteq I(\{s_1\})$ and $I(\{s_2\}) \cap I(\{s_1, s_2\}) = \emptyset$. The green components in $\mathcal{C}^*_G(\{s_2\})$ are $H_{26}, \ldots, H_{36}$, the component H_{37} is black, and the $\{s_2\}$-branch $G[\{s_1\} \cup V(H_1) \cup \cdots \cup V(H_{25})]$ is purple.

By definition, any bottleneck S has at least 13 S-branches. The proof of Lemma 3.5 shows that this number guarantees the existence of green S-branches for a bottleneck S. In the next section, we show that we can limit ourselves to the a special class of path decompositions, referred to as clean path decompositions, which have no red and no blue S-branches for any bottleneck S. However, gray and black S-branches are unavoidable since it may happen that $I(S) \subsetneq \{\alpha(G[S]), \ldots, \beta(G[S])\}$ as is the case in the example in Fig. 5. We also remark that the restriction to clean path decompositions would make it possible to consider bottleneck sets as those having at least 7 (rather than 13) S-branches. Though this would improve the complexity of our polynomial-time algorithm, we do not make this attempt to optimize its running time.

MOST WIEDZY Downloaded from mostwiedzy.pl

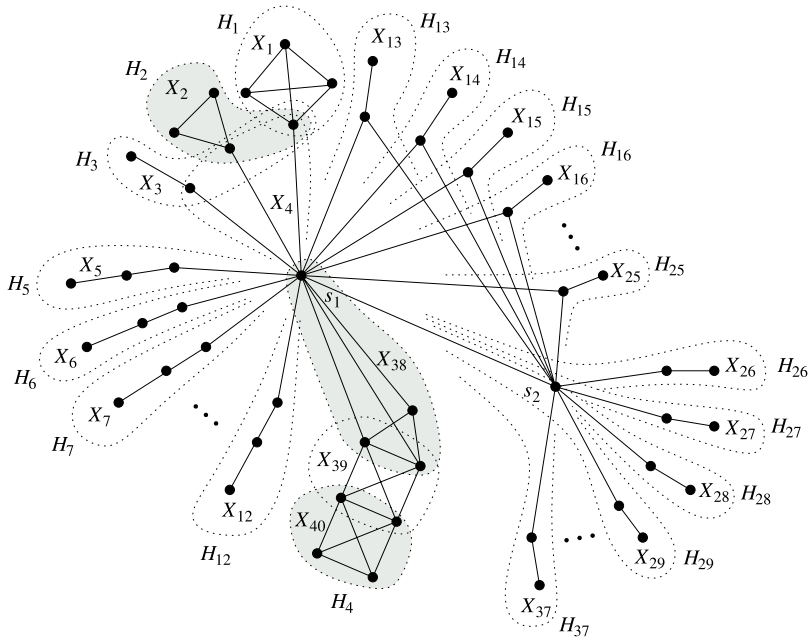

Fig. 5. A simple graph G with a path decomposition $\mathcal{P}$, width$(\mathcal{P}) = 3$.

3.4. Well-arranged path decompositions

We showed in the previous section that green S-branches appear only in the bottleneck interval $I(S)$ of $S \in \mathcal{S}$. However, red, blue and purple S-branches may also appear in $I(S)$. (We say that an S-branch H appears in $I(S)$ if there is $t \in I(S)$ such that $X_t \cap V(H) \neq \emptyset$.) Our goal in this section is to show that the search for minimum-length path decompositions of width $k \leq 3$ can be limited to a class of well-arranged path decompositions with no red and blue S-branches for any $S \in \mathcal{S}$. Moreover, any path decomposition in this class suspends all purple branches in $I(S)$ so that only vertices of purple S-branches that appear already in $\{1, \ldots, t_1(S) - 1\}$ may appear in $I(S)$ for any $S \in \mathcal{S}$. We now formally define the well-arranged path decompositions.

Let $\mathcal{P} = (X_1, \ldots, X_l)$ be a path decomposition of G. We say that a subgraph H of G *waits* in step i, $1 \leq i \leq l$, if $X_i \cap V(H) \subseteq X_{i-1} \cap V(H)$. (In the latter statement we take $X_0 = \emptyset$.) If H does not wait in step i, then we say that H *makes progress* in step i. For an interval $I \subseteq \{1, \ldots, l\}$, we say that a subgraph H of G *waits in* I if H waits in all steps $i \in I$, and we say that H *makes progress in* I otherwise.

Definition 3.6. For $S \in \mathcal{S}$, we say that a path decomposition is *well-arranged with respect to* S if all components in $G - S$, except possibly S-leaves and green S-branches, wait in the interval $I(S)$. (Recall that not every component of $G - S$ is necessarily an S-component.) A path decomposition is called *well-arranged* if it is well-arranged with respect to every bottleneck set.

In this section, we will show that for every graph G it is true that if G has a path decomposition of length l and width at most k, then G has a well-arranged path decomposition of length l and width at most k.

To show the existence of well-arranged path decompositions, we will choose our path decomposition to be minimal in a certain sense. To make this precise, let us first order, for a given path decomposition $\mathcal{P}$ of G, the bottleneck sets of G as $S_1, \ldots, S_p$ such that for all $i, j \in \{1, \ldots, p\}$ with $i \leq j$:

$$t_1(S_i) \leq t_1(S_j), \text{ and if } t_1(S_i) = t_1(S_j), \text{ then } t_2(S_i) \leq t_2(S_j). \tag{5}$$

That is, we order the bottleneck sets by starting time $t_1(S)$ and, in case of a tie, by ending time $t_2(S)$. We associate with $\mathcal{P}$ the vector $\boldsymbol{\alpha}(\mathcal{P}) = (|I(S_1)|, \ldots, |I(S_p)|)$.

Definition 3.7. A path decomposition $\mathcal{P}$ of G is called *clean* if, among all path decompositions of width at most width$(\mathcal{P})$ and length at most len$(\mathcal{P})$ of G, the following holds:

(C1) size$(\mathcal{P})$ is minimum;
(C2) subject to **(C1)**, the vector $\boldsymbol{\alpha}(\mathcal{P})$ is lexicographically smallest.

MOST WIEDZY Downloaded from mostwiedzy.pl

We explicitly note the following:

Observation 3.8. *If a graph has a path decomposition of width at most k and length at most l, then it has a clean path decomposition of width at most k and length at most l.*

3.4.1. Basic characteristics of clean path decompositions

We now prove some characteristics of clean path decompositions. The proofs will require the following lemma that strengthens Observation 1.2 for connected H.

Lemma 3.9. *Let G be a graph, let $\mathcal{P}$ be a path decomposition of G, and let H be any connected subgraph of G. Then, for each $t \in \{\alpha(H), \ldots, \beta(H) - 1\}$, we have $|X_t \cap X_{t+1} \cap V(H)| \geq 1$.*

Proof. Let $t \in \{\alpha(H), \ldots, \beta(H) - 1\}$ be given. Define $A = (X_{\alpha(H)} \cup \ldots \cup X_t) \cap V(H)$ and $B = (X_{t+1} \cup \ldots \cup X_{\beta(H)}) \cap V(H)$. Since $X_{\alpha(H)} \cap V(H)$ and $X_{\beta(H)} \cap V(H)$ are non-empty, it follows that A and B are non-empty. Moreover, $A \cup B = V(H)$ by (1). If $A \cap B = \emptyset$, then (A, B) is a partition of $V(H)$; by **(PD2)** it then follows that there are no edges between A and B, contrary to the fact that H is connected. Thus, there exists $v \in A \cap B$. By **(PD3′)**, $v \in X_t \cap X_{t+1}$ and hence $v \in (X_t \cap X_{t+1}) \cap V(H)$. □

We begin by showing that any S-branch, $S \in \mathcal{S}$, that starts in $I(S)$ does so with two vertices at a time; similarly, any S-branch that completes in $I(S)$ does so with two vertices at a time.

Lemma 3.10. *Let G be a connected graph, let $\mathcal{P}$ be a clean path decomposition, $S \in \mathcal{S}$, and let H be an S-branch. The following statements hold:*

(i) *If $\alpha(H) \in I(S)$, then $|V(H) \cap X_{\alpha(H)}| \geq 2$.*
(ii) *If $\beta(H) \in I(S)$, then $|V(H) \cap X_{\beta(H)}| \geq 2$.*

Proof. (i) By the definition of $\alpha(H)$, we have $|V(H) \cap X_{\alpha(H)}| \geq 1$. Suppose for a contradiction that $V(H) \cap X_{\alpha(H)} = \{v\}$. It follows from **(PD2)** that, for every $u \in N_H(v)$, there exists $\tau(u) \in \{\alpha(H), \ldots, \beta(H)\}$ such that $\{u, v\} \subseteq X_{\tau(u)}$. By the assumption that $X_{\alpha(H)} \cap V(H) = \{v\}$, it follows that $\tau(u) \neq \alpha(H)$. Moreover, by Lemma 3.9, we have $|X_{\alpha(H)+1} \cap X_{\alpha(H)} \cap V(H)| \geq 1$, which implies that $v \in X_{\alpha(H)+1}$. Finally, because $\alpha(H) \in I(S)$, we have $S \subseteq X_{\alpha(H)+1}$. Now construct a new path decomposition $\mathcal{P}'$ from $\mathcal{P}$ by replacing bag $X_{\alpha(H)}$ by $X_{\alpha(H)} \setminus \{v\}$. Then, **(PD1)** still holds for $\mathcal{P}'$ because $v \in X_{\alpha(H)+1}$. To check **(PD2)**, notice that the only edges that might violate this condition are the ones of the form $\{u, v\}$ where u is a neighbor of v. Any neighbor u of v is either in S or in $N_H(v)$. For $u \in S$, we have $\{u, v\} \subseteq X_{\alpha(H)+1}$; for $u \in N_H(v)$, we have $\{u, v\} \subseteq X_{\tau(u)}$. Thus, **(PD2)** holds for $\mathcal{P}'$. Finally, v is the only vertex that might violate **(PD3′)**. However, since $v \notin X_j$ for $j < \alpha(H)$, condition **(PD3′)** holds for v. Thus, $\mathcal{P}'$ is a path decomposition with $\text{size}(\mathcal{P}') < \text{size}(\mathcal{P})$, contrary to the fact that $\mathcal{P}$ is clean. This proves (i). Part (ii) follows from the symmetry, i.e., by applying (i) to the reverse of $\mathcal{P}$. □

For any $S \in \mathcal{S}$, by Lemma 3.5, $I(S)$ is well-defined and hence the definition of $I(S)$ implies that there exists some S-branch that starts at $t_1(S)$, and similarly there exists some S-branch that completes at $t_2(S)$. Together with Lemma 3.10 and the fact that $k \leq 3$, this gives the following useful corollary:

Corollary 3.11. *Let G be a connected graph, let $\mathcal{P}$ be a clean path decomposition of width $k \leq 3$ of G, and $S \in \mathcal{S}$. Then, the following statements hold:*

(i) *there is exactly one green S-branch H such that $|X_{t_1(S)} \cap V(H)| \geq 2$.*
(ii) *there is exactly one green S-branch H such that $|X_{t_2(S)} \cap V(H)| \geq 2$.*

We say that H in (i) *determines* $t_1(S)$, and similarly, H in (ii) *determines* $t_2(S)$.

Moreover, the following lemma shows that if an S-branch H, $S \in \mathcal{S}$, makes progress in step $t \in I(S)$ of a clean path decomposition, then X_t must contain at least two vertices of $V(H)$.

Lemma 3.12. *Let G be a connected graph, let $\mathcal{P}$ be a clean path decomposition, $S \in \mathcal{S}$, and let H be an S-branch. Then, for any $t \in I(S)$, if H makes progress in step t, then $|X_t \cap V(H)| \geq 2$.*

Proof. Clearly, since H makes progress in step $t \in I(S)$, we have $t \in \{\alpha(H), \ldots, \beta(H)\} \cap I(S)$. If $t = \alpha(H)$, then the result follows from Lemma 3.10. So we may assume that $t \in \{\alpha(H) + 1, \ldots, \beta(H)\} \cap I(S)$. Then, there exists a vertex $u \in (X_t \setminus X_{t-1}) \cap V(H)$. However, by Lemma 3.9, we have $|X_t \cap X_{t-1} \cap V(H)| \geq 1$. Thus, there is a vertex $v \in X_t \cap X_{t-1} \cap V(H)$ and $v \neq u$. Therefore, $\{u, v\} \subseteq X_t \cap V(H)$, as required. □

MOST WIEDZY Downloaded from mostwiedzy.pl

Finally, Corollary 3.11 and $k \le 3$, give the following upper bounds on the numbers of red, blue and purple S-branches for $S \in \mathcal{S}$.

Lemma 3.13. *Let G be a connected graph, let $\mathcal{P}$ be a clean path decomposition of width $k \le 3$ of G, and $S \in \mathcal{S}$. Then, the following statements hold:*

(i) *there is at most one red or purple S-branch;*
(ii) *there is at most one blue or purple S-branch;*
(iii) *if there is a red, purple, or blue S-branch, then $k = 3$ and $|S| = 1$.*

Proof. By Corollary 3.11, there exist green S-branches H_1 and H_2 (possibly $H_1 = H_2$) such that $\{u_i, v_i\} \subseteq X_{t_i(S)} \cap V(H_i)$ for each $i = 1, 2$. Suppose that there exists an S-branch that is red or purple and let $H'_1, \ldots, H'_r$ be all S-branches that are red or purple. By Lemma 3.9, for each $i = 1, \ldots, r$ there exists $u'_i \in V(H'_i) \cap X_{t_1(S)}$. Hence, $S \cup \{u_1, v_1\} \cup \{u'_1, \ldots, u'_r\} \subseteq X_{t_1(S)}$. By the definition of the coloring, $u'_i \notin \{u_1, v_1\}$ for each $i \in \{1, \ldots, r\}$. Thus, $r \le 1$, which proves (i).

For (ii), suppose that there exist $q \ge 1$ S-branches that are blue or purple. Denote those S-branches by $H''_1, \ldots, H''_q$. By Lemma 3.9, for each $i = 1, \ldots, q$ there exists $u''_i \in V(H''_i) \cap X_{t_2(S)}$. Hence, $S \cup \{u_2, v_2\} \cup \{u''_1, \ldots, u''_r\} \subseteq X_{t_2(S)}$. By the definition of the coloring, $u''_i \notin \{u_2, v_2\}$ for each $i \in \{1, \ldots, q\}$. Thus, $q \le 1$ and (ii) follows.

Finally, if $r = 1$ or $q = 1$, then $S \cup \{u_1, v_1, u'_1\} \subseteq X_{t_1(S)}$ or $S \cup \{u_2, v_2, u''_1\} \subseteq X_{t_2(S)}$, respectively. Because $|X_{t_i(S)}| \le k + 1 \le 4$, $i = 1, 2$, this implies that $k = 3$ and $|S| = 1$, as required in (iii). □

3.4.2. The absence of red and blue S-branches

We now show that clean path decompositions lack red and blue components. We start with the following lemma, which provides a convenient way of re-arranging a path decomposition. Notice that this lemma applies to all components of $G - S$ (i.e., S-branches, but also S-leaves and other components of $G - S$ that are not S-components). Note also that we can always add empty sets to the beginning and/or the end of a path decomposition, so that the condition that the path decompositions be of length exactly $|I(S)|$ is less restrictive than one may think at first glance. (Recall that $\mathcal{S}$ is the collection of all bottleneck sets of G.)

Lemma 3.14. *Let G be a connected graph, let $\mathcal{P} = (X_1, \ldots, X_l)$ be a path decomposition of width $k \le 3$ of G, and $S \in \mathcal{S}$. Write $t_1 = t_1(S)$ and $t_2 = t_2(S)$. For each component H of $G - S$, let $\mathcal{Q}^H = (Y^H_{t_1}, \ldots, Y^H_{t_2})$ be a path decomposition of length $|I(S)|$ for the graph*

$$H(S, \mathcal{P}) = H\left[\bigcup_{t \in I(S)} X_t \cap V(H)\right],$$

satisfying the following conditions:

(i) *if $\alpha_{\mathcal{P}}(H) < t_1$, then $X_{t_1} \cap V(H) \subseteq Y^H_{t_1}$; and*
(ii) *if $\beta_{\mathcal{P}}(H) > t_2$, then $X_{t_2} \cap V(H) \subseteq Y^H_{t_2}$.*

Define $\mathcal{P}' = \left(X_1, \ldots, X_{t_1-1}, S \cup \bigcup_H Y^H_{t_1}, \ldots, S \cup \bigcup_H Y^H_{t_2}, X_{t_2+1}, \ldots, X_l\right)$, where the unions are taken over all components H of $G - S$. Then, $\mathcal{P}'$ is a path decomposition of G of width at most k.

Proof. For convenience, define $\mathcal{P}'_1 = \left(X_1, \ldots, X_{t_1-1}\right)$, $\mathcal{P}'_2 = \left(S \cup \bigcup_H Y^H_{t_1}, \ldots, S \cup \bigcup_H Y^H_{t_2}\right)$, and $\mathcal{P}'_3 = \left(X_{t_2+1}, \ldots, X_l\right)$. Recall from Lemma 3.5 that $I_{\mathcal{P}}(S) \ne \emptyset$, and hence $\mathcal{P}'_2$ is not an empty list. Moreover, define $Z = G - \bigcup_{t \in I_{\mathcal{P}}(S)} X_t$, i.e., Z is the subgraph of G induced by all vertices not appearing in any bag X_t with $t \in I_{\mathcal{P}}(S)$. Notice that $V(G) = V(Z) \cup S \cup \bigcup_H V(H(S, \mathcal{P}))$. Write $\mathcal{P}' = (X'_1, \ldots, X'_l)$.

To check **(PD2)**, let $\{u, v\} \in E(G)$. There are a few possibilities. If $v \in V(Z)$, then, by **(PD2)** for $\mathcal{P}$, there exists some t such that $\{u, v\} \in X_t$; since $v \notin X_s$ for all $s \in I_{\mathcal{P}}(S)$, it follows that $t \notin I_{\mathcal{P}}(S)$, and hence $X'_t = X_t$, so that $\{u, v\} \in X'_t$. If $u, v \in S$, then $\{u, v\} \subseteq X'_{t_1}$. If $u, v \in V(H(S, \mathcal{P}))$ for some component H of $G - S$, then **(PD2)** for $\mathcal{Q}^H$ implies that $\{u, v\} \subseteq Y^H_t$ for some $t \in I_{\mathcal{P}}(S)$, and hence $\{u, v\} \subseteq X'_t$. Finally, if $u \in S$, $v \in V(H(S, \mathcal{P}))$ for some component H of $G - S$, then by **(PD1)** for $\mathcal{Q}^H$, $v \in Y^H_t$ for some $t \in I_{\mathcal{P}}(S)$, and hence $\{u, v\} \subseteq X'_t$. This establishes **(PD2)**. Note that **(PD1)** follows from **(PD2)** because G is connected.

Finally, in order to verify **(PD3′)**, let $v \in V(G)$. If $v \in S \cup V(Z)$, then notice that $X'_t \cap (S \cup V(Z)) = X_t \cap (S \cup V(Z))$ for all $t \in \{1, \ldots, l\}$, and hence property **(PD3′)** for v follows immediately from the fact that **(PD3′)** holds for $\mathcal{P}$. So we may assume that $v \in V(H(S, \mathcal{P}))$ for some component H of $G - S$. Let $1 \le i < i' \le l$ be such that $v \in X'_i$ and $v \in X'_{i'}$. We need to show that $v \in X'_j$ for all $j \in \{i, \ldots, i'\}$. Notice that, because $v \in V(H(S, \mathcal{P}))$, there exists $t^* \in I_{\mathcal{P}}(S)$ such that $v \in X_{t^*}$. Let us go through the cases:

MOST WIEDZY Downloaded from mostwiedzy.pl

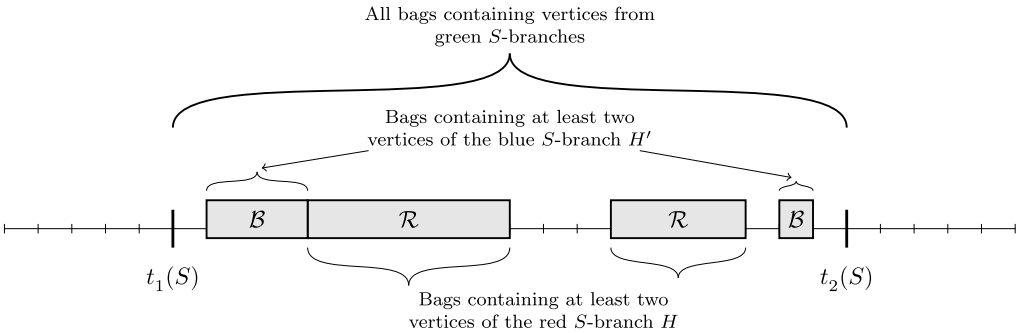

Fig. 6. $\mathcal{P}$ is not well-arranged with respect to S.

(1) $i \le i' < t_1$ or $t_2 < i \le i'$. Then, $X'_j = X_j$ for all $j \in \{i, \dots, i'\}$, and hence the fact that $v \in X'_j$ follows directly from **(PD3′)** for $\mathcal{P}$.
(2) $t_1 \le i \le i' \le t_2$. Then, $v \in Y_i^H$ and $v \in Y_{i'}^H$ and hence the fact that $v \in Y_j^H \subseteq X'_j$, $j = i, \dots, i'$, follows directly from **(PD3′)** for $\mathcal{Q}^H$.
(3) $i < t_1 \le i' \le t_2$. Then, $v \in X_i$. Hence, by **(PD3′)** applied to i, t_1 and $\mathcal{P}$, it follows that $v \in X_j$ for all $j \in \{i, \dots, t_1\}$. In particular, also $v \in X_{t_1}$. Since $v \in X_i$, we have $\alpha(H) < t_1$. Thus, by condition (i) in the statement of the lemma, it follows that $v \in Y_{t_1}^H \subseteq X'_{t_1}$. Now it follows from (2) applied to t_1, i' that $v \in X'_j$ for all $j \in \{t_1, \dots, i'\}$.
(4) $t_1 \le i \le t_2 < i'$. It follows from (3) and the symmetry (through $\mathcal{P}' = (X_l, \dots, X_1)$) that $v \in X'_j$ for all $j \in \{i, \dots, i'\}$.
(5) $i < t_1 \le t_2 < i'$. It follows from (3) applied to i, t_2 that $v \in X'_j$ for all $j \in \{i, \dots, t_2\}$, and it follows from (4) applied to t_2, i' that $v \in X'_j$ for all $j \in \{t_2, \dots, i'\}$, as required.

This proves the lemma. □

We make the following observation about shortening path decompositions.

Observation 3.15. *Let G be a connected graph and let $\mathcal{P} = (X_1, \dots, X_l)$ be a path decomposition of G. Denote $X_0 = X_{l+1} = \emptyset$. Then, the deletion of any bag X_t, $t \in \{1, \dots, l\}$, satisfying $X_t \subseteq X_{t-1} \cap X_{t+1}$ results in a path decomposition of width at most* $\mathrm{width}(\mathcal{P})$ *of G.*

We now prove the main result of this subsection. This result is the first step towards proving that clean path decompositions are well-arranged.

Lemma 3.16. *Let G be a connected graph and $S \in \mathcal{S}$. Let $\mathcal{P} = (X_1, \dots, X_l)$ be a clean path decomposition of width $k \le 3$ of G. Then, there are no blue and no red S-branches.*

Proof. First note that by Lemma 3.13(iii), if $k \le 2$, then there are no red and no blue S-branches, and hence there is nothing to prove. So we may assume from now on that $k = 3$. Write $t_1 = t_1(S)$ and $t_2 = t_2(S)$. Suppose for a contradiction that there is a red or a blue S-branch H. We will construct a new path decomposition $\mathcal{P}'$ of width $k \le 3$ and length l for G, which satisfies $\mathrm{size}(\mathcal{P}') < \mathrm{size}(\mathcal{P})$, contrary to the fact that $\mathcal{P}$ is clean.

It follows from Lemma 3.13(iii) that there are no purple S-branches and $|S| = 1$. It also follows from Lemma 3.13 that there may exist at most one blue S-branch and at most one red S-branch. Let H be the red S-branch, if any exists; otherwise let H be the empty graph. Similarly, let H' be the blue S-branch, if any exists; otherwise let H' be the empty graph. Notice that, by assumption, H and H' are not both empty graphs.

Since G is connected, the fact that $|S| = 1$ implies that every component of $G - S$ is an S-component. Let $\mathcal{R} \subseteq I_{\mathcal{P}}(S)$ be the set of steps t such that $|X_t \cap V(H)| \ge 2$, let $\mathcal{B} \subseteq I_{\mathcal{P}}(S)$ be the set of steps t such that $|X_t \cap V(H')| \ge 2$, and let $\mathcal{Z} = I_{\mathcal{P}}(S) \setminus (\mathcal{R} \cup \mathcal{B})$. Let $r_1 < r_2 < \dots < r_p$ be the steps in $\mathcal{R}$, let $b_1 < b_2 < \dots < b_q$ be the steps in $\mathcal{B}$, and let $z_1 < z_2 < \dots < z_s$ be the steps in $\mathcal{Z}$. We show that $\mathcal{R} \cup \mathcal{B} = \emptyset$ first. Suppose for a contradiction that $\mathcal{R} \cup \mathcal{B} \ne \emptyset$. Corollary 3.11 and $k = 3$ imply $t_1, t_2 \in \mathcal{Z}$, so that $z_1 = t_1$, $z_s = t_2$, and $\mathcal{R} \cap \mathcal{B} = \emptyset$ (see Fig. 6).

For $t \in I_{\mathcal{P}}(S)$, write $X_t = S_t \cup R_t \cup B_t \cup Z_t$, where S_t contains the vertex of S and all S-leaves in X_t, $R_t = X_t \cap V(H)$, $B_t = X_t \cap V(H')$, and $Z_t = X_t \setminus (S_t \cup V(H) \cup V(H'))$. Thus, R_t consists of the vertices of the red S-branch H (if any) in X_t, B_t contains the vertices of the blue S-branch H' (if any) in X_t, Z_t contains the vertices in X_t that belong to green S-branches in X_t. Note that S_t, R_t, Z_t and B_t are pairwise disjoint.

Construct $\mathcal{P}'$ from $\mathcal{P}$ by replacing the bags $X_{t_1}, \dots, X_{t_2}$, of $\mathcal{P}$ by the bags

$$S_{r_1} \cup R_{r_1}, \dots, S_{r_p} \cup R_{r_p}, \quad S_{z_1} \cup Z_{z_1}, \dots, S_{z_r} \cup Z_{z_s}, \quad S_{b_1} \cup B_{b_1}, \dots, S_{b_q} \cup B_{b_q}. \tag{6}$$

MOST WIEDZY Downloaded from mostwiedzy.pl

Let $\mathcal{P}' = (X'_1, \ldots, X'_l)$ be the result of the replacement. Let $\pi : I_{\mathcal{P}}(S) \to I_{\mathcal{P}}(S)$ be the permutation function that maps $(t_1, \ldots, t_2)$ to $(r_1, \ldots, r_p, z_1, \ldots, z_s, b_1, \ldots, b_q)$. We will show that $\mathcal{P}'$ is a path decomposition of width at most 3 of G that satisfies $\text{size}(\mathcal{P}') < \text{size}(\mathcal{P})$.

We first claim that $|X'_t| \le 4$ for all $t \in \{1, \ldots, l\}$. To see this, note that $X'_t = X_t$ for all $t \notin I_{\mathcal{P}}(S)$, and hence $|X'_t| \le 4$ trivially holds. Also, $X'_t \subseteq X_{\pi(t)}$ for all $t \in I_{\mathcal{P}}(S)$, thus again $|X'_t| \le 4$.

Next, we claim that $\text{size}(\mathcal{P}') < \text{size}(\mathcal{P})$. It suffices to show that $|X'_{\pi(t_1)}| < |X_{t_1}|$ or $|X'_{\pi(t_2)}| < |X_{t_2}|$. By assumption, at least one of $\mathcal{B}$, $\mathcal{R}$ is non-empty. If $\mathcal{B} \neq \emptyset$, then since H' is a blue S-branch there exists $w \in X_{t_2} \cap V(H')$. By Corollary 3.11(ii), $|X_{t_2} \cap V(H')| = 1$, thus, $\pi(t_2) \in \mathcal{R} \cup \mathcal{Z}$. Therefore, $X'_{\pi(t_2)} \subseteq X_{t_2}$, and $w \notin X'_{\pi(t_2)}$ by (6) which proves $|X'_{\pi(t_2)}| < |X_{t_2}|$. Similarly, if $\mathcal{R} \neq \emptyset$, then since H is a red S-branch there is $u \in X_{t_1} \cap V(H)$. By Corollary 3.11(i), $|X_{t_1} \cap V(H)| = 1$, thus, $\pi(t_1) \in \mathcal{Z} \cup \mathcal{B}$. Therefore, $X'_{\pi(t_1)} \subseteq X_{t_1}$ and, by (6), $u \notin X'_{\pi(t_1)}$ which proves $|X'_{\pi(t_1)}| < |X_{t_1}|$. Thus, indeed $\text{size}(\mathcal{P}') < \text{size}(\mathcal{P})$.

We now argue that $\mathcal{P}'$ is a path decomposition of G. For every S-component J such that $V(J) \cap X_t \neq \emptyset$ for some $t \in I_{\mathcal{P}}(S)$ define

$$J(S, \mathcal{P}) = J\left[\bigcup_{t \in I_{\mathcal{P}}(S)} X_t \cap V(J)\right],$$

i.e., $J(S, \mathcal{P})$ is the subgraph of J induced by all vertices that appear in bags of $\mathcal{P}$ in the interval $I_{\mathcal{P}}(S)$. We have, by Observation 3.15:

- $\mathcal{Q}^H = (R_{r_1}, \ldots, R_{r_p}, \emptyset, \ldots, \emptyset)$ is a path decomposition of length $|I_{\mathcal{P}}(S)|$ of $H(S, \mathcal{P})$;
- $\mathcal{Q}^{H'} = (\emptyset, \ldots, \emptyset, B_{b_1}, \ldots, B_{b_q})$ is a path decomposition of length $|I_{\mathcal{P}}(S)|$ of $H'(S, \mathcal{P})$;
- for any green S-branch H^*, $\mathcal{Q}^{H^*} = (\emptyset, \ldots, \emptyset, Z_{z_1} \cap V(H^*), \ldots, Z_{z_s} \cap V(H^*), \emptyset, \ldots, \emptyset)$ is a path decomposition of length $|I_{\mathcal{P}}(S)|$ of $H^*(S, \mathcal{P})$;
- for J being an S-leaf v, $\alpha_{\mathcal{P}}(v) \in I(S)$, $\mathcal{Q}^J = (S_{\pi(t_1)} \cap \{v\}, \ldots, S_{\pi(t_2)} \cap \{v\})$ is a path decomposition of length $|I_{\mathcal{P}}(S)|$ of $J(S, \mathcal{P})$.

In order to apply Lemma 3.14, it remains to show that $X_{t_1} \cap V(H) \subseteq X'_{t_1}$ and $X_{t_2} \cap V(H') \subseteq X'_{t_2}$. Notice that $X_{t_1} \cap V(H) = R_{t_1}$ and $X'_{t_1} = S_{r_1} \cup R_{r_1}$. Thus, it suffices to show that $R_{t_1} \subseteq R_{r_1}$. This is trivial if there is no red S-branch. Otherwise, because $|R_{t_1}| = 1$ and H waits in $\{t_1, \ldots, r_1 - 1\}$, it follows that $R_{t_1} = \cdots = R_{r_1 - 1}$. By Lemma 3.9, $|R_{r_1-1} \cap R_{r_1}| \ge 1$ and hence $R_{t_1} \subseteq R_{r_1}$. Similarly, it suffices to show that $B_{t_2} \subseteq B_{b_q}$. Again, this is trivial if there is no blue S-branch. Otherwise, because $|B_{b_q+1}| = 1$ and H' waits in $\{b_q + 1, \ldots, t_2\}$, it follows that $B_{b_q+1} = \cdots = B_{t_2}$. By Lemma 3.9, $|B_{b_q} \cap B_{b_q+1}| \ge 1$ and hence $B_{t_2} \subseteq B_{b_q}$.

Thus, by Lemma 3.14, $\mathcal{P}'$ is a path decomposition of G, which proves that $\mathcal{R} \cup \mathcal{B} = \emptyset$. Therefore if H is non-empty, then $X_{t_1} \cap V(H) = \cdots = X_{\beta(H)} \cap V(H) = \{v\}$. (Otherwise, H makes progress in $I_{\mathcal{P}}$, and thus Lemma 3.12 implies $\mathcal{R} \neq \emptyset$, a contradiction.) Since $\alpha(H) < t_1$, we have $X_{t_1-1} \cap X_{t_1} \cap V(H) = \{v\}$ by Lemma 3.9. Moreover, by definition of t_1, $S \subseteq X_{t_1-1}$. Therefore, deleting v from the bags $X_{t_1}, \ldots, X_{\beta(H)}$ would result in a path decomposition $\mathcal{P}'$ of G such that $\text{size}(\mathcal{P}') < \text{size}(\mathcal{P})$, since $t_1 \le \beta(H)$, contrary to the fact that $\mathcal{P}$ is clean. Thus, no red S-branch exists. The proof that no blue branch exists follows by symmetry (take $(X_l, \ldots, X_1)$ instead of $\mathcal{P} = (X_1, \ldots, X_l)$). □

3.4.3. *Clean path decompositions are well-arranged*

We now show that clean path decompositions are well-arranged, and thus, by Observation 3.8, we may limit our search for a minimum-length path decomposition to well-arranged path decompositions. Then, we give the third key property of clean path decompositions.

Lemma 3.17. *Let G be a connected graph and let $\mathcal{P} = (X_1, \ldots, X_l)$ be a clean path decomposition of width $k \le 3$ of G. Then, $\mathcal{P}$ is well-arranged.*

Proof. Let $S_1, \ldots, S_p$ be the bottleneck sets of G ordered as in (5). We will prove by induction on m that $\mathcal{P}$ is well-arranged with respect to the bottleneck sets $S_1, \ldots, S_m$, for each $m = 0, \ldots, p$. Thus, the case $m = p$ is the result of the lemma. The base case $m = 0$ is trivial. For the general case, let $1 \le m \le p$ and assume inductively that $\mathcal{P}$ is well-arranged with respect to $S_1, \ldots, S_{m-1}$. Let $\boldsymbol{\alpha}_m(\mathcal{P})$ be the vector with the first m entries of $\boldsymbol{\alpha}(\mathcal{P})$.

Now suppose for a contradiction that $\mathcal{P}$ is not well-arranged with respect to S_m. The latter implies that some component H of $G - S_m$, which is neither an S_m-leaf nor a green S_m-branch, makes progress in some step in $I_{\mathcal{P}}(S_m)$. It follows from Lemma 3.16 that there no red and no blue S_m-branches. Thus, H is either not an S_m-component or a purple S_m-branch. We deal with these two cases separately.

Case 1: H is not an S_m-component. If $|S_m| = 1$, then every component of $G - S_m$ is an S_m-component, so it follows that $|S_m| \ge 2$. Therefore, by Lemma 3.13(iii), there are no purple S_m-branches. By Corollary 3.11, some green S_m-branches make progress in steps $t_1(S_m)$ and $t_2(S_m)$. Thus, $k = 3$, $|S_m| = 2$, and $X_{t_1(S_m)}$ and $X_{t_2(S_m)}$ contain no vertices of H. Denote $S_m =$

MOST WIEDZY Downloaded from mostwiedzy.pl

$\{s_1, s_2\}$. Let $r_1 = \alpha_{\mathcal{P}}(H)$ and $r_2 = \beta_{\mathcal{P}}(H)$. Since H makes progress in some step in $I_{\mathcal{P}}(S_m)$, $t_1(S_m) < r_1 \le r_2 < t_2(S_m)$. We may assume that H is selected in such a way that r_2 is as large as possible. Define

$$\mathcal{P}' = \big(X_1, \ldots, X_{r_1-1}, \quad X_{r_2+1}, \ldots, X_{t_2(S_m)}, \quad X_{r_1}, \ldots, X_{r_2}, \quad X_{t_2(S_m)+1}, \ldots, X_l\big).$$

The proof that $\mathcal{P}'$ is a path decomposition of G follows from the following key observations. First, only the vertices of S_m, green S_m-branches, S_m-leaves, and components of $G - S_m$ that are not S_m-components can appear in $X_{t_1(S_m)}, \ldots X_{t_2(S_m)}$. Second, no vertex of H appears outside of $X_{r_1}, \ldots, X_{r_2}$. Third, $|X_t \cap V(H)| \ge 1$ for each $t \in \{r_1, \ldots, r_2\}$, $|S_m| = 2$ and the assumption that r_2 is as large as possible, imply that no $G - S_m$ component $H' \ne H$ with $|V(H)| \ge 2$ (this clearly includes S_m-branches) makes progress in steps $r_1, \ldots, r_2$. Thus, no bag X_t for $t \in \{r_1, \ldots, r_2\}$ contains a vertex of a component $H' \ne H$ of $G - S_m$ with $|V(H)| \ge 2$, because otherwise we could reduce the size($\mathcal{P}$) and thus contradict **(C1)**. Indeed, $|X_t \cap (S_m \cup V(H))| \ge 3$ for each $t \in \{r_1, \ldots, r_2\}$ because H is connected, and, by Lemma 3.12, $|X_t \cap (S_m \cup V(H))| > 3$ for some $t \in \{r_1, \ldots, r_2\}$. This implies that H' makes no progress in any step in $\{r_1, \ldots, r_2\}$ and either $V(H') \cap (X_1 \cup \cdots \cup X_{r_1-1}) = \emptyset$ or $V(H') \cap (X_{r_2+1} \cup \cdots \cup X_l) = \emptyset$. The latter implies that removal of the vertices of H' from the bags $X_{r_1}, \ldots, X_{r_2}$ of $\mathcal{P}$ provides a valid path decomposition of G with smaller cost, which gives the required contradiction with **(C1)**. Therefore, each vertex in $X_t \setminus (S_m \cup V(H))$, $t \in \{r_1, \ldots, r_2\}$, belongs to a component $H' \ne H$ of $G - S_m$ with $|V(H)| = 1$ (this clearly includes S_m-leaves). Therefore, $X_{r_1-1} \cap X_{r_1} = S_m = X_{r_2} \cap X_{r_2+1}$. Finally, by Corollary 3.11, some green S_m-branch makes progress in step $t_2(S_m)$. Therefore, since $k = 3$ and $|S_m| = 2$, we have $S_m = X_{t_2(S_m)} \cap X_{t_2(S_m)+1}$.

We claim that $\boldsymbol{\alpha}_m(\mathcal{P}') <_L \boldsymbol{\alpha}_m(\mathcal{P})$, which contradicts **(C2)**. To prove this claim, it suffices to show that

$$t'_1(S_i) = t_1(S_i) \text{ and } t'_2(S_i) \le t_2(S_i) \tag{7}$$

for all $i = 1, \ldots, m$, with strict inequality $t'_2(S_i) < t_2(S_i)$ for $i = m$.

First note that $\alpha_{\mathcal{P}'}(x) = \alpha_{\mathcal{P}}(x)$ and $\beta_{\mathcal{P}'}(x) = \beta_{\mathcal{P}}(x)$ for each $x \in S_m$. Thus, $\widehat{\alpha}_{\mathcal{P}}(S_m) = \widehat{\alpha}_{\mathcal{P}'}(S_m)$ and $\widehat{\beta}_{\mathcal{P}}(S_m) = \widehat{\beta}_{\mathcal{P}'}(S_m)$. Let H' and H'' be the green S_m-branches that determine $t_1(S_m)$ and $t_2(S_m)$ in $\mathcal{P}$, respectively. We have $\alpha_{\mathcal{P}'}(H') = \alpha_{\mathcal{P}}(H')$ and $\beta_{\mathcal{P}'}(H'') < \beta_{\mathcal{P}}(H'')$. By definition, there is no S_m-branch J such that $\widehat{\alpha}_{\mathcal{P}}(S_m) < \alpha_{\mathcal{P}}(J) < t_1(S_m)$ or $t_2(S_m) < \beta_{\mathcal{P}}(J) < \widehat{\beta}_{\mathcal{P}}(S_m)$. Also, we proved that there are no vertices of S_m-branches in $X_{r_1}, \ldots, X_{r_2}$, thus the S_m-branches H' and H'' determine $t'_1(S_m)$ and $t'_2(S_m)$ in $\mathcal{P}'$, respectively. Therefore, $t'_1(S_m) = t_1(S_m)$ and $t'_2(S_m) < t_2(S_m)$, thus the inequality $t'_2(S_i) < t_2(S_i)$ for $i = m$ in (7) is strict.

Now consider S_i, with $i \in \{1, \ldots, m-1\}$. Because $i < m$, we have that $t_1(S_i) \le t_1(S_m)$. By definition of $t_1(S_i)$,

$$S_i \subseteq X_t \text{ for some } t < t_1(S_m). \tag{8}$$

Let $u_1, u_2 \in V(H') \cap X_{t_1(S_m)}$. Since $k = 3$, we have $X_{t_1(S_m)} = \{s_1, s_2, u_1, u_2\}$. Let H'_i and H''_i be the green S_i-branches that determine $t_1(S_i)$ and $t_2(S_i)$ in $\mathcal{P}$, respectively. If S_i contains a vertex that is not in $X_{t_1(S_m)}$, then, by (8), $t_2(S_i) < \widehat{\beta}_{\mathcal{P}}(S_i) < t_1(S_m)$. Thus, H'_i and H''_i determine $t'_1(S_i)$ and $t'_2(S_i)$ in $\mathcal{P}'$. Therefore, $t'_1(S_i) = t_1(S_i)$ and $t'_2(S_i) = t_2(S_i)$ and (7) holds for the i. So we may assume that $S_i \subseteq X_{t_1(S_m)}$. If $u_j \in S_i$ for some $j \in \{1, 2\}$, then $t_1(S_i) > \widehat{\alpha}_{\mathcal{P}}(S_i) \ge t_1(S_m)$, a contradiction. Thus, $S_i \subseteq \{s_1, s_2\}$. By the symmetry and the fact that $S_i \ne S_m$, we may assume that $S_i = \{s_1\}$.

Since $\widehat{\alpha}_{\mathcal{P}}(S_i) \le \widehat{\alpha}_{\mathcal{P}}(S_m)$ and $\widehat{\beta}_{\mathcal{P}}(S_m) \le \widehat{\beta}_{\mathcal{P}}(S_i)$, we have $\widehat{\alpha}_{\mathcal{P}}(S_i) = \widehat{\alpha}_{\mathcal{P}'}(S_i)$ and $\widehat{\beta}_{\mathcal{P}}(S_m) = \widehat{\beta}_{\mathcal{P}'}(S_i)$. Also, since $t_1(S_i) \le t_1(S_m)$, H'_i determines $t'_1(S_i)$ in $\mathcal{P}'$. Thus, $t'_1(S_i) = t_1(S_i)$. By definition, there is no S_i-branch J such that $t_2(S_i) < \beta_{\mathcal{P}}(J) < \widehat{\beta}_{\mathcal{P}}(S_i)$. Thus, both for $t_2(S_m) < \beta_{\mathcal{P}}(H''_i) = t_2(S_i)$ and for $\beta_{\mathcal{P}}(H''_i) = t_2(S_i) < r_1$, H''_i determines $t'_2(S_i)$ in $\mathcal{P}'$. Thus, in both cases $t'_2(S_i) = t_2(S_i)$ and (7) holds for the i. Finally, suppose that $r_1 \le \beta_{\mathcal{P}}(H''_i) \le t_2(S_m)$. We show that this case leads to a contradiction with the inductive assumption that $\mathcal{P}$ is well-arranged with respect to S_i. Consider the component H^* in $G - S_i$ that contains s_2. Since $|S_i| = 1$, all $G - S_i$ components are S_i-components. Moreover, $|V(H^*)| \ge 2$ because H^* includes also u_1 and u_2, and hence H^* is an S_i-branch. We have, $\beta_{\mathcal{P}}(H^*) \ge \widehat{\beta}_{\mathcal{P}}(S_i)$, because otherwise H^* is a green S_i-branch, and $\beta_{\mathcal{P}}(H^*) \ge \beta_{\mathcal{P}}(s_2) \ge \widehat{\beta}_{\mathcal{P}}(S_m) > t_2(S_m) \ge \beta_{\mathcal{P}}(H''_i)$. Therefore, since, by definition of H''_i, $\beta_{\mathcal{P}}(J) \le \beta_{\mathcal{P}}(H''_i)$ for any green S_i-branch J, we get a contradiction. However, $\beta_{\mathcal{P}}(H^*) \ge \widehat{\beta}_{\mathcal{P}}(S_i)$ and Lemma 3.16 imply that H^* is a purple S_i-branch (observe that $\alpha_{\mathcal{P}}(H^*) < t_2(S_i)$, since $\alpha_{\mathcal{P}}(H^*) \le \alpha_{\mathcal{P}}(s_2) \le \widehat{\alpha}_{\mathcal{P}}(S_m) < t_1(S_m) < r_1 \le t_2(S_i)$). Moreover, H^* makes progress in $t_1(S_m)$ for S_m-branch H' makes progress in $t_1(S_m)$ and H' is a subgraph of H^*. This contradicts the fact that $\mathcal{P}$ is well-arranged with respect to S_i for $t_1(S_m) \in I_{\mathcal{P}}(S_i)$ (observe that $t_1(S_i) \le t_1(S_m) < r_1 \le t_2(S_i)$).

Case 2: H is a purple S_m-branch. By Lemma 3.13, $|S_m| = 1$, and $k = 3$. Denote $S_m = \{s\}$. Every component in $G - S_m$ is either an S_m-branch or an S_m-leaf. Because H makes progress in $I_{\mathcal{P}}(S_m)$, Lemma 3.12 implies that $|X_t \cap V(H)| \ge 2$ for some $t \in I_{\mathcal{P}}(S_m)$. Let $r_2 \in I_{\mathcal{P}}(S_m)$ be the latest step such that $|X_{r_2} \cap V(H)| \ge 2$ and let $r_1 \le r_2$ be the earliest step such that $|X_t \cap V(H)| \ge 2$ for all $t = r_1, \ldots, r_2$. For each $t \in I_{\mathcal{P}}(S_m)$, write $X_t = Z_t \cup R_t \cup A_t$, where Z_t is the set containing S_m and all S_m-leaves contained in X_t, $R_t = X_t \cap V(H)$, and $A_t = X_t \setminus (R_t \cup Z_t)$. Notice that A_t contains precisely the vertices in X_t that belong to green S_m-branches. By Corollary 3.11, some green S_m-branches make progress in steps $t_1(S_m)$ and $t_2(S_m)$. Thus, Observation 1.2, $k = 3$, and definition of purple S_m-branch imply $|R_{t_1(S_m)}| = |R_{t_2(S_m)}| = 1$. Therefore, $t_1(S_m) < r_1 \le r_2 < t_2(S_m)$.

By Observation 1.2, definition of purple S_m-branch, and the choice of r_1 and r_2, we have $|R_{r_1-1}| = 1$, and $|R_t| = 1$ for all $t \in \{r_2 + 1, \ldots, t_2(S_m)\}$. The latter also implies that $|Z_t \cup A_t| \le 3$ for all $t \in \{r_2 + 1, \ldots, t_2(S_m)\}$. Construct $\mathcal{P}'$ from $\mathcal{P}$ by replacing the bags $X_{r_1}, \ldots, X_{t_2(S_m)}$ by

MOST WIEDZY Downloaded from mostwiedzy.pl

$$Z_{r_2+1} \cup A_{r_2+1} \cup R_{r_1-1}, \quad \ldots, \quad Z_{t_2(S_m)} \cup A_{t_2(S_m)} \cup R_{r_1-1}, \quad Z_{r_1} \cup R_{r_1}, \quad \ldots, \quad Z_{r_2} \cup R_{r_2}. \tag{9}$$

Clearly, $\mathcal{P}'$ satisfies width$(\mathcal{P}') \le 3$. We claim that no green S_m-branch H'' makes progress in a step $t \in \{r_1, \ldots, r_2\}$. Indeed, if this were the case, then $|X_t \cap V(H'')| \ge 2$ and, by the choice of r_1 and r_2, we would have $|X_t \cap (Z_t \cup R_t)| \ge 3$, which is not possible because width$(\mathcal{P}) \le 3$. Thus, if $V(H'') \cap X_t \ne \emptyset$ for some green S_m-branch H'', then $V(H'') \cap X_t = \{v\}$ for some vertex v and each $t \in \{r_1, \ldots, r_2\}$. Hence, again due to the minimality of size$(\mathcal{P})$, we have that $v \in A_{r_2+1}$. This and Lemma 3.14 imply that $\mathcal{P}'$ is a path decomposition of G. If $A_t \ne \emptyset$ for some $t \in \{r_1, \ldots, r_2\}$, then, by (9), size$(\mathcal{P}') <$ size$(\mathcal{P})$, which contradicts **(C1)**. So, $A_t = \emptyset$ for all $t \in \{r_1, \ldots, r_2\}$.

We claim again that $\boldsymbol{\alpha}_m(\mathcal{P}') <_L \boldsymbol{\alpha}_m(\mathcal{P})$. As in Case 1, it suffices to show that (7) holds for all $i = 1, \ldots, m$, with strict inequality $t'_2(S_i) < t_2(S_i)$ for $i = m$. First note that $\alpha_{\mathcal{P}'}(s) = \alpha_{\mathcal{P}}(s)$ and $\beta_{\mathcal{P}'}(s) = \beta_{\mathcal{P}}(s)$. Thus, $\widehat{\alpha}_{\mathcal{P}}(S_m) = \widehat{\alpha}_{\mathcal{P}'}(S_m)$ and $\widehat{\beta}_{\mathcal{P}}(S_m) = \widehat{\beta}_{\mathcal{P}'}(S_m)$. Let H' and H'' be the green S_m-branches that determine $t_1(S_m)$ and $t_2(S_m)$ in $\mathcal{P}$, respectively. We have $\alpha_{\mathcal{P}'}(H') = \alpha_{\mathcal{P}}(H')$ and $\beta_{\mathcal{P}'}(H'') < \beta_{\mathcal{P}}(H'')$. By definition, there is no S_m-branch J such that $\widehat{\alpha}_{\mathcal{P}}(S) < \alpha_{\mathcal{P}}(J) < t_1(S_m)$ or $t_2(S_m) < \beta_{\mathcal{P}}(J) < \widehat{\beta}_{\mathcal{P}}(S)$. Moreover, we proved that there are no vertices of green S_m-branches in $X_{r_1}, \ldots, X_{r_2}$, and by Lemma 3.16 there are no red and blue S_m-branches, thus S_m-branches H' and H'' determine $t'_1(S_m)$ and $t'_2(S_m)$ in $\mathcal{P}'$. Therefore, $t'_1(S_m) = t_1(S_m)$ and $t'_2(S_m) < t_2(S_m)$, thus the inequality for $i = m$ in (7) is strict.

Now consider S_i with $i \in \{1, \ldots, m-1\}$. Because $i < m$, we have that $t_1(S_i) \le t_1(S_m)$. By definition of $t_1(S_i)$, this implies (8). Let $u_1, u_2 \in V(H') \cap X_{t_1(S_m)}$. Let w be the unique vertex of H in $X_{t_1(S_m)}$. Since $k = 3$, we have $X_{t_1(S_m)} = \{s, w, u_1, u_2\}$. If S_i contains a vertex that is not in $X_{t_1(S_m)}$, then, by (8), $t_2(S_i) < \widehat{\beta}_{\mathcal{P}}(S_i) < t_1(S_m)$. Thus, $t'_1(S_i) = t_1(S_i)$ and $t'_2(S_i) = t_2(S_i)$ and (7) holds. So we may assume that $S_i \subseteq X_{t_1(S_m)}$. If $u_j \in S_i$ for some $j \in \{1, 2\}$, then $t_1(S_i) > \widehat{\alpha}_{\mathcal{P}}(S_i) \ge t_1(S_m)$, a contradiction. Thus, $S_i \subseteq \{s, w\}$. Since $S_i \ne S_m$, this implies that either $S_i = \{s, w\}$ or $S_i = \{w\}$.

If $S_i = \{s, w\}$, then, by Corollary 3.11, H' makes progress in $t_1(S_m)$. However, H' is a component in $G - S_i$ which is not an S_i-component. Thus, $t_1(S_m) \le t_2(S_i)$ contradicts that $\mathcal{P}$ is well-arranged with respect to S_i (observe that $t_1(S_m) \in I_{\mathcal{P}}(S_i)$). If $t_1(S_m) > t_2(S_i)$, then $t'_1(S_i) = t_1(S_i)$ and $t'_2(S_i) = t_2(S_i)$ and (7) holds.

Consider $S_i = \{w\}$. Let H'_i and H''_i be the green S_i-branches that determine $t_1(S_i)$ and $t_2(S_i)$ in $\mathcal{P}$, respectively. Since $\widehat{\alpha}_{\mathcal{P}}(S_i) \le t_1(S_m)$ and $t_2(S_m) \le \widehat{\beta}_{\mathcal{P}}(S_i)$, we have $\widehat{\alpha}_{\mathcal{P}}(S_i) = \widehat{\alpha}_{\mathcal{P}'}(S_i)$ and $\widehat{\beta}_{\mathcal{P}}(S_i) = \widehat{\beta}_{\mathcal{P}'}(S_i)$. Since $t_1(S_i) \le t_1(S_m)$, H'_i determines $t'_1(S_i)$ in $\mathcal{P}'$. Thus, $t'_1(S_i) = t_1(S_i)$. By definition, there is no S_i-branch J such that $t_2(S_i) < \beta_{\mathcal{P}}(S_i) < \widehat{\beta}_{\mathcal{P}}(S)$. Thus, both for $t_2(S_m) < \beta_{\mathcal{P}}(H''_i) = t_2(S_i)$ and for $\beta_{\mathcal{P}}(H''_i) = t_2(S_i) < r_1$, H''_i determines $t'_2(S_i)$ in $\mathcal{P}'$. Thus, in both cases $t'_2(S_i) = t_2(S_i)$ and (7) holds. Now suppose that $r_1 \le \beta_{\mathcal{P}}(H''_i) \le t_2(S_m)$. We show that this case leads to a contradiction with the inductive assumption that $\mathcal{P}$ is well-arranged with respect to S_i. The S_i-component H^* that contains s is an S_i-branch for it includes also u_1 and u_2. Thus, $\beta_{\mathcal{P}}(H^*) \ge \widehat{\beta}_{\mathcal{P}}(S_i)$, because otherwise H^* is a green S_i-branch and $\beta_{\mathcal{P}}(H^*) \ge \beta_{\mathcal{P}}(s) > t_2(S_m) \ge \beta_{\mathcal{P}}(H''_i)$, which contradicts the choice of H''_i as the branch that determines $t_1(S_i)$. However $\beta_{\mathcal{P}}(H^*) \ge \widehat{\beta}_{\mathcal{P}}(S_i)$ and Lemma 3.16 imply that H^* is a purple S_i-branch (observe that $\alpha_{\mathcal{P}}(H^*) < t_2(S_i)$, since $\alpha_{\mathcal{P}}(H^*) \le \alpha_{\mathcal{P}}(s) = \widehat{\alpha}(S_m) < t_1(S_m) < r_1 \le t_2(S_i)$). Moreover, H^* makes progress in $t_1(S_m)$ for H' makes progress in $t_1(S_m)$ and H' is a subgraph of H^*. This contradicts the fact that $\mathcal{P}$ is well-arranged with respect to S_i for $t_1(S_m) \in I_{\mathcal{P}}(S_i)$. □

A collection $\mathcal{B}$ of sets is called *strictly nested* if for all $X, X' \in \mathcal{B}$, $(X \ne X')$ and (either $X \cap X' = \emptyset$, or $X \subseteq X'$, or $X' \subseteq X$).

Lemma 3.18. *Let G be a connected graph and let $\mathcal{P} = (X_1, \ldots, X_l)$ be a clean path decomposition of width $k \le 3$ of G. Then, the collection $\{I(S) \colon S \in \mathcal{S}\}$ is strictly nested.*

Proof. Let S and S' be two distinct bottleneck sets. We may assume without loss of generality that $|S| \ge |S'|$. Write $I(S) = \{t_1, \ldots, t_2\}$ and $I(S') = \{t'_1, \ldots, t'_2\}$. Suppose for a contradiction that $(I(S) = I(S'))$ or $(I(S) \cap I(S') \ne \emptyset$, $I(S) \not\subseteq I(S')$ and $I(S') \not\subseteq I(S))$. Perhaps by taking $(X_l, \ldots, X_1)$, we may assume that $t_1 \le t'_1 \le t_2 \le t'_2$. By definition of t'_1, there exists a green S'-branch H' that determines t'_1. Fix $z \in X_{t'_1} \cap V(H')$. By Lemma 3.16 there are no red or blue S-branches, and by Lemma 3.17 any purple S-branch and any component in $G - S$ that is not an S-component waits in $I(S)$, thus it follows that z is a vertex of an S-component H which is either an S-leaf or a green S-branch.

We first show that H is a green S-branch. Let $x \in S \setminus S'$ (such x exists because $|S| \ge |S'|$ and $S \ne S'$). If x and z are in the same connected component (i.e., in H') in $G - S'$, then $t'_1 = \alpha(H') \le \alpha(x) \le \widehat{\alpha}(S) < t_1 \le t'_1$, a contradiction. Thus, x and z are in different connected components of $G - S'$. This implies that any path between x and z passes through a vertex in S'. Now suppose for a contradiction that H is an S-leaf. Then $V(H) = \{z\}$ and the edge $(x, z) \in E(G)$ is a path between x and z that does not pass through a vertex in S' since neither x nor s belong to S'. Therefore, H is a green S-branch, thus includes some vertex $w \ne z$.

Now, since we just shown that $(x, z) \notin E(G)$, we can chose w so that $(x, w) \in E(G)$. Therefore, $w \in S' \setminus S$ or at least one vertex on any path between w and z in H must be in S'. Thus, $S' \cap V(H) \ne \emptyset$. This, however, implies that $\beta(H) \ge \widehat{\beta}(S') > t'_2 \ge t_2$, contrary to the fact that $\beta(H) \le t_2$ (which follows from the fact that H is a green S-branch). This proves the lemma. □

3.5. *An algorithm for connected graphs*

In this section we turn the generic exponential-time algorithm from Section 3.1 into a polynomial-time algorithm that finds a minimum-length path decomposition of width at most k of G for given integer $k \le 3$ and connected graph G. Recall

MOST WIEDZY Downloaded from mostwiedzy.pl

that it can be checked in linear time whether $\mathrm{pw}(G) \le k$; see [6,9]. We formulate the algorithm in this section and prove its correctness in the next section.

Let $k \in \{1, 2, 3\}$ and let G be a connected graph. For every non-empty set $S \subseteq V(G)$, $|S| \le k+1$, define

$$\begin{aligned}\mathcal{Q}(S) = \big\{(\mathbf{G}, f, l) \;\big|\; & \mathbf{G} \subseteq \mathcal{C}^*_G(S), |\mathbf{G}| \ge |\mathcal{C}^*_G(S)| - 12, \\ & f: \mathcal{C}^*_G(S) \to \{0,1\},\ f(H) = f(H') \text{ for all } H, H' \in \mathbf{G}, \\ & l \in \{0, \ldots, |\mathcal{C}^1_G(S)|\}\big\}\end{aligned} \tag{10}$$

if $S \in \mathcal{S}$ (i.e., if S is a bottleneck set), and

$$\mathcal{Q}(S) = \{(\emptyset, f, l) \;\big|\; f: \mathcal{C}^*_G(S) \to \{0,1\},\ l \in \{0, \ldots, |\mathcal{C}^1_G(S)|\}\} \tag{11}$$

if $S \notin \mathcal{S}$. Every triple in $\mathcal{Q}(S)$ provides information on which S-components have been covered (completed) by a partial path decomposition. Here, $f(H) = 1$ means that H has been covered.

The first entry, $\mathbf{G}$, represents the collection of green S-branches. By Lemma 3.5, in any path decomposition, every bottleneck set S has at most 12 non-green S-branches, thus there are at least $|\mathbf{G}| \ge |\mathcal{C}^*_G(S)| - 12$ green S-branches. Moreover, by the definition of a bottleneck, there must be at least one green S-branch. Thus, $\mathbf{G}$ is always non-empty for $S \in \mathcal{S}$. Since we only define the coloring of S-components for bottleneck sets S, $\mathbf{G}$ is set to be empty for $S \notin \mathcal{S}$.

The second entry, f, keeps track of which S-branches have been covered by a partial path decomposition (f assigns 1 to those S-branches). Notice that we require that either *all* green S-branches are covered, or *none* of them are covered. The third entry, l, counts the number of S-leaves that are covered by a partial path decomposition.

Note that for $|S| > 1$, there may exist connected components in $G - S$ that are not S-components, and thus $\mathcal{Q}(S)$ does not provide any information about whether such components have been covered or not by a partial path decomposition. This information, however, is not lost since it is provided by $\mathcal{Q}(S')$ for some $S' \subsetneq S$.

Next, for $X \subseteq V(G)$, define $\mathcal{R}(X)$ to be the set of all functions $R: 2^X \setminus \{\emptyset\} \to \bigcup_{S \subseteq X} \mathcal{Q}(S)$ such that $R(S) \in \mathcal{Q}(S)$ for each $S \subseteq X$, $S \ne \emptyset$. Thus, each $R \in \mathcal{R}(X)$ selects a triple from $\mathcal{Q}(S)$ for each non-empty set $S \subseteq X$. Now define the following edge-weighted directed graph $\mathcal{G}_k$ with weights $w: E(\mathcal{G}_k) \to \mathbb{Z}$. The vertex set of $\mathcal{G}_k$ is

$$V(\mathcal{G}_k) = \{(X, R) \;\big|\; X \subseteq V(G), |X| \le k+1, R \in \mathcal{R}(X)\} \cup \{s, t\}.$$

In the following we denote $s = (\emptyset, \emptyset)$ and $\mathcal{C}^1_G(S) = \{v_i(S) \mid i = 1, \ldots, |\mathcal{C}^1_G(S)|\}$. Notice that if $S \ne S'$, then $\mathcal{C}^1_G(S)$ and $\mathcal{C}^1_G(S')$ are disjoint.

Every vertex $v \in V(\mathcal{G}_k)$ represents a set $\mathcal{V}_v$ of the vertices of G covered by any partial path decomposition that corresponds v. We first show how to define $\mathcal{V}_v$ for any given vertex $v = (X, R) \in V(\mathcal{G}_k)$. For each non-empty $S \subseteq X$, write $R(S) = (\mathbf{G}_S, f_S, l_S)$. Now, $\mathcal{V}_v$ is given by

$$\mathcal{V}_v = X \cup \bigcup_{S \subseteq X, S \ne \emptyset} \left[\{v_1(S), \ldots, v_{l_S}(S)\} \quad \cup \bigcup_{\substack{H \in \mathcal{C}^*_G(S) \\ f_S(H) = 1}} V(H) \right]. \tag{12}$$

We are now ready to formally define the edge set of $\mathcal{G}_k$ and the corresponding edge weights. The informal comments follow the definition. Let there be an edge from v to t if and only if $\mathcal{V}_v = V(G)$; the weight of such an edge is set to zero. Then, let there be an edge from $v = (X, R) \in V(\mathcal{G}_k)$ to $v' = (X', R') \in V(\mathcal{G}_k)$ ($v \ne v'$) if and only if $\mathcal{V}_v \subseteq \mathcal{V}_{v'}$, $(X' \setminus \delta_G(G[\mathcal{V}_v])) \cap \mathcal{V}_v = \emptyset$ and one of the two following conditions holds:

(G1) $\mathcal{V}_{v'} \setminus \mathcal{V}_v \subseteq X'$ and $\delta_G(G[\mathcal{V}_v]) = X' \cap \mathcal{V}_v$. Set $w(v, v') = 1$. We refer to the edge (v, v') as a *step edge*.

(G2) $\mathcal{V}_{v'} \setminus \mathcal{V}_v \not\subseteq X'$, and there exists a bottleneck set $S \subseteq X \cap X'$ with $R'(S) = (\mathbf{G}'_S, f'_S, l'_S)$ such that:

- **(G2a)** $\mathcal{V}_{v'} \setminus \mathcal{V}_v = Y_S$, where $Y_S = \{v_{l_S+1}(S), \ldots, v_{l'_S}(S)\} \cup \bigcup_{H \in \mathbf{G}'_S} V(H)$, and $R(S) = (\mathbf{G}_S, f_S, l_S)$.
- **(G2b)** $\delta_G(G[\mathcal{V}_v]) = \bar{S}$, where $\bar{S} = X \cap X'$.
- **(G2c)** $X' = \bar{S}$.
- **(G2d)** there exists a path decomposition $\bar{\mathcal{P}} = (\bar{X}_1, \ldots, \bar{X}_{\bar{t}})$ of width

$$\mathrm{width}(\bar{\mathcal{P}}) \le k - |\bar{S}| \tag{13}$$

for a possibly disconnected graph $\bar{G} = G[Y_S]$. Set $w(v, v') = \bar{t}$, where $\bar{t}$ is taken as small as possible. (In other words, we take $\bar{\mathcal{P}}$ to be of minimum length.)

We refer to the edge (v, v') as a *jump edge* and to S as the *bottleneck set associated with* (v, v'). We refer to any path decomposition $(\bar{X}_1 \cup \bar{S}, \ldots, \bar{X}_{\bar{t}} \cup \bar{S})$ as a *witness* of the jump edge.

MOST WIEDZY Downloaded from mostwiedzy.pl

We naturally extend the weight function w to the subsets of edges, i.e., $w(F) = \sum_{e \in F} w(e)$ for any $F \subseteq E(\mathcal{G}_k)$.

An s–t path in $\mathcal{G}_k$ allows to construct a sequence of bags as follows. We start with an empty sequence in s. Then, whenever that path passes through a directed edge (v, v') in $\mathcal{G}_k$ we append the bag X', if (v, v') is a step edge, or a sequence of bags $(X'_1, \ldots, X'_{\bar{t}})$, where $X'_i = \bar{S} \cup \bar{X}_i$ for all $i = 1, \ldots, \bar{t}$, if (v, v') is a jump edge, to the sequence. We stop reaching t.

In the next section, we prove the following theorem that is analogous to Lemma 3.2 for the generic exponential-time algorithm:

Theorem 3.19. *Let $k \in \{1, 2, 3\}$ and let G be a connected graph. There exists an s–t path of weighted length l in $\mathcal{G}_k$ if and only if there exists a path decomposition of width at most k and length l of G.*

Before we give a formal proof of the theorem, we provide some informal insights into the definition of $\mathcal{G}_k$ and the above construction of a path decomposition for an s–t path in $\mathcal{G}_k$, and we show how Theorem 3.19 implies Theorem 3.1.

The conditions **(G1)**, **(G2b)**, and **(G2c)** guarantee that X' includes the border of $G[\mathcal{V}_v]$. This is intended to guarantee **(PD2)** and **(PD3)** in Definition 1.1 for the partial path decompositions built for step edges and jump edges. For the latter the border is added to each bag of the path decomposition $\bar{\mathcal{P}}$ and thus it is carried forward till the last bag of the partial path decomposition. This is clearly necessary since, by **(G2a)** and **(G2d)**, $\bar{\mathcal{P}}$ includes only the vertices of green S-branches and S-leaves; however, both S and possibly other vertices (for instance, those of purple S-branches) belong to both the border of $G[\mathcal{V}_v]$ and the border of $G[\mathcal{V}_{v'}]$. Finally, the condition $(X' \setminus \delta_G(G[\mathcal{V}_v])) \cap \mathcal{V}_v = \emptyset$ guarantees that vertices in $\mathcal{V}_v$ without neighbors in $V(G) \setminus \mathcal{V}_v$ are *not* carried forward for it is unnecessary. Assuming the correctness of Theorem 3.19, we can now prove Theorem 3.1.

Proof of Theorem 3.1. It follows from Theorem 3.19 that a solution to the problem MLPD(k)-CONSTR can be obtained by constructing $\mathcal{G}_k$ and finding a shortest s–t path P in $\mathcal{G}_k$. This approach leads to a polynomial-time algorithm, because $\mathcal{G}_k$ can be constructed in time polynomial in n, where $n = |V(G)|$, given polynomial-time algorithms that find minimum-length path decompositions of widths $0, 1, \ldots, k-1$ for disconnected graphs. Indeed, consider any non-empty $S \subseteq V(G)$ with $|S| \leq k+1$. Recall that $c = |C^*_G(S)|$. If S is a bottleneck set, then, by (10), the number of subsets $\mathbf{G}$ of green S-branches, the number of functions f, and the number l of leaves do not exceed c^{12}, 2^{13}, and $n - 2c$, respectively; thus, $|\mathcal{Q}(S)| \leq (n - 2c) \times c^{12} \times 2^{13}$. Similarly, if S is not a bottleneck set, then, by (11), we have that $|\mathcal{Q}(S)| \leq (n - 2c) \times 2^{13}$. Since $c \leq (n - |S|)/2$, it follows that $|\mathcal{Q}(S)| \leq 2n^{13}$. Thus, the cardinality of codomain and domain of any function $R \in \mathcal{R}(X)$ do not exceed $2^{k+2} \times n^{13}$ and 2^{k+1}, respectively, for a given $X \subseteq V(G)$. Hence, $|\mathcal{R}(X)| = O(n^{13 \times 2^{k+1}})$ for any given $X \subseteq V(G)$. Finally, since $|X| \leq k+1$, we have that $|V(\mathcal{G}_k)| \leq n^{O(2^k)}$. Therefore, $|V(\mathcal{G}_k)|$ is bounded by a polynomial in n. Each jump edge of $\mathcal{G}_k$, $k > 0$, can be constructed in polynomial time by using an algorithm that for any, possibly disconnected, graph finds its path decomposition of width not exceeding $k - |\bar{S}|$, $|\bar{S}| > 0$, whenever one exists. Finally, note that $\mathrm{pw}(G) = 0$ implies that $G = K_1$. This proves Theorem 3.1. □

Our algorithm can be significantly simplified for $k < 3$. If $k = 1$, then $\mathcal{G}_1$ contains no jump edges, and any path decomposition $\mathcal{P} = (X_1, \ldots, X_l)$ for G such that $X_i \neq X_{i+1}$, $i = 1, \ldots, l-1$, solves MLPD(1)-CONSTR. If $k = 2$, then each bottleneck is of size 1. However, since our focus is on settling the border between polynomial-time and NP-hard cases in the minimum-length path decomposition problem, we do not attempt to optimize the running time of the polynomial-time algorithm.

It remains to prove Theorem 3.19. We start with an observation and then we prove (in Lemma 3.21 and Lemma 3.22, respectively) both directions of Theorem 3.19.

Observation 3.20. *For each $v = (X, R) \in V(\mathcal{G}_k)$, it holds that $\delta_G(G[\mathcal{V}_v]) \subseteq X$.*

Proof. Suppose for a contradiction that there exists a vertex $u \in \delta_G(G[\mathcal{V}_v]) \setminus X$. By border definition $\delta_G(G[\mathcal{V}_v]) \subseteq \mathcal{V}_v$, thus (12) implies that there exists a non-empty set $S \subseteq X$ such that u is in some S-component H, and $V(H) \subseteq \mathcal{V}_v$. Also by the definition, u has a neighbor $u' \in V(G) \setminus \mathcal{V}_v$ which, by $S \subseteq X \subseteq \mathcal{V}_v$, implies that $u' \notin S$. Therefore, since each neighbor of u is either in S or in $V(H)$, we get $u' \in V(H)$. But, by (12), this implies that $u' \in \mathcal{V}_v$, a contradiction. □

Lemma 3.21. *Let $k \in \{1, 2, 3\}$ and let G be a connected graph. Let $P = v_0$–v_1–v_2–$\ldots$–v_m be an s–t path in $\mathcal{G}_k$ of weighted length ℓ. Then, there exists a path decomposition $\mathcal{P}$ of G with* $\mathrm{width}(\mathcal{P}) \leq k$ *and* $\mathrm{len}(\mathcal{P}) \leq \ell$.

Proof. Since P is an s–t path, we have $v_0 = s$ and $v_m = t$. For brevity, we will write $(X_i, R_i) = v_i$, $\mathcal{V}_i = \mathcal{V}_{v_i}$, and $G_i = G[\mathcal{V}_i]$ for each $i = 0, \ldots, m-1$. Also, for each $i = 0, \ldots, m-1$, let $P_i = v_0$–v_1–$\ldots$–v_i denote the prefix of P formed by the first i edges, and let ℓ_i be the weighted length of P_i.

We prove, by induction on $i = 0, \ldots, m-1$, that there exists a path decomposition $\mathcal{P}_i$ of G_i that satisfies $\mathrm{width}(\mathcal{P}_i) \leq k$, $\mathrm{len}(\mathcal{P}_i) = \ell_i$, and $\delta_G(G_i)$ is contained in the last bag of $\mathcal{P}_i$. The claim is trivial for $i = 0$. Indeed, because $\mathcal{V}_0 = \emptyset$, we may take $\mathcal{P}_0$ to be a path decomposition of length 0. For the generic case, suppose that the claim holds for some i, $0 \leq i < m-1$. We now prove that it holds for $i+1$. We have two cases, depending on the type of the edge (v_i, v_{i+1}).

MOST WIEDZY Downloaded from mostwiedzy.pl

Case 1: (v_i, v_{i+1}) is a step edge. Recall that, by the definition of the step edge, we have

$$\delta_G(G_i) = X_{i+1} \cap \mathcal{V}_i. \tag{14}$$

We claim that $\mathcal{P}_{i+1} := (\mathcal{P}_i, X_{i+1})$ is a path decomposition of G_{i+1} of width at most k and of length $\mathrm{len}(\mathcal{P}_{i+1}) \le \ell_{i+1} = \ell_i + 1$. We will verify the conditions **(PD1)**, **(PD2)**, and **(PD3)**.

(i) Since (v_i, v_{i+1}) is a step edge, we have $\mathcal{V}_{i+1} \setminus \mathcal{V}_i \subseteq X_{i+1}$. Moreover, by (12), $X_{i+1} \subseteq \mathcal{V}_{i+1}$. Since, by the construction of the edge set of $\mathcal{G}_k$, $\mathcal{V}_i \subseteq \mathcal{V}_{i+1}$, it follows that $\mathcal{V}_{i+1} = \mathcal{V}_i \cup X_{i+1}$. Thus, by the induction hypothesis, $\mathrm{span}(\mathcal{P}_{i+1}) = \mathcal{V}_i \cup X_{i+1} = \mathcal{V}_{i+1}$. Hence, **(PD1)** follows.

(ii) Let $\{x, y\} \in E(G_{i+1})$. Note that $x, y \in V(G_{i+1}) = \mathcal{V}_{i+1} = \mathcal{V}_i \cup X_{i+1}$. We claim that some bag of $\mathcal{P}_{i+1}$ contains both x and y. If x and y are both in $\mathcal{V}_i$, then this follows from the induction hypothesis. If x and y are both in $\mathcal{V}_{i+1} \setminus \mathcal{V}_i$, then $\{x, y\} \subseteq X_{i+1}$, as required. So we may assume that $x \in \mathcal{V}_{i+1} \setminus \mathcal{V}_i$ and $y \in \mathcal{V}_i$. Thus, y has a neighbor, i.e., x, in G that is not in $\mathcal{V}_i$, and it follows that $y \in \delta_G(G_i)$. Hence, by (14), $y \in X_{i+1}$. Thus, since $\mathcal{V}_{i+1} \setminus \mathcal{V}_i \subseteq X_{i+1}$ (by the definition of a step edge), we have $\{x, y\} \subseteq X_{i+1}$, as required. This settles condition **(PD2)**.

(iii) By (14), for all $j \le i$ we have $X_j \cap X_{i+1} \subseteq X_{i+1} \cap \mathcal{V}_i = \delta_G(G_i)$. Thus, by Observation 3.20, $X_j \cap X_{i+1} \subseteq X_i$ which implies $X_j \cap X_{i+1} \subseteq X_j \cap X_i$ for $j \le i$. Therefore, since $\mathcal{P}_i$ is a path decomposition of G_i, by the induction hypothesis we have $X_j \cap X_{i+1} \subseteq X_j \cap X_i \subseteq X_p$ for each $j \le p \le i$. This proves that **(PD3)** holds for $\mathcal{P}_{i+1}$.

Finally, it follows from the induction hypothesis and from the fact that $|X_{i+1}| \le k+1$ that every bag of $\mathcal{P}_{i+1}$ has size at most $k+1$. Moreover, $w(v_i, v_{i+1}) = 1$. Thus, $\ell_{i+1} = \ell_i + 1$. On the other hand, $\mathrm{len}(\mathcal{P}_{i+1}) = \mathrm{len}(\mathcal{P}_i) + 1$. However, by the induction hypothesis, $\mathrm{len}(\mathcal{P}_i) = \ell_i$. Thus, $\mathrm{len}(\mathcal{P}_{i+1}) = \ell_{i+1}$. Therefore, $\mathcal{P}_{i+1}$ is a path decomposition of G_{i+1} with $\mathrm{width}(\mathcal{P}_{i+1}) \le k$ and $\mathrm{len}(\mathcal{P}_{i+1}) = \ell_{i+1}$. Moreover, by (14), $\delta_G(G_{i+1}) \subseteq X_{i+1}$, which completes the proof of Case 1.

Case 2: (v_i, v_{i+1}) is a jump edge. Let S, Y_S and $\bar{\mathcal{P}} = (\bar{X}_1, \ldots, \bar{X}_{\bar{t}})$ be as in **(G2)**. Note that, by the definition of $\bar{\mathcal{P}}$ in **(G2d)**, $\mathrm{span}(\bar{\mathcal{P}}) = Y_S$. Let $\bar{S}$ be defined as in **(G2b)**, i.e., $\bar{S} = X_i \cap X_{i+1}$. We claim that

$$\mathcal{P}_{i+1} := (\mathcal{P}_i, \bar{S} \cup \bar{X}_1, \bar{S} \cup \bar{X}_2, \ldots, \bar{S} \cup \bar{X}_{\bar{t}}) \tag{15}$$

is a path decomposition of G_{i+1} of width at most k and of length $\mathrm{len}(\mathcal{P}_{i+1}) = \ell_{i+1} = \ell_i + \bar{t}$. Since $\bar{S} \subseteq X_i \subseteq \mathcal{V}_v$, **(G2a)** and **(G2d)** imply that $\bar{S} \cap \bar{X}_j = \emptyset$ for all $j = 1, \ldots, \bar{t}$. Thus $(\bar{S} \cup \bar{X}_1, \ldots, \bar{S} \cup \bar{X}_{\bar{t}})$ is a path decomposition of the subgraph $G[Y_S \cup \bar{S}]$. We now verify that $\mathcal{P}_{i+1}$ satisfies the conditions **(PD1)**, **(PD2)**, and **(PD3)**.

(i) Note that, by **(G2c)**, $\bar{S} = X_{i+1}$. By (15), **(G2b)**, **(G2d)** and by the induction hypothesis,

$$\mathrm{span}(\mathcal{P}_{i+1}) = \mathrm{span}(\mathcal{P}_i) \cup Y_S \cup \bar{S} = \mathcal{V}_i \cup Y_S \cup X_{i+1} = \mathcal{V}_i \cup Y_S = \mathcal{V}_{i+1}. \tag{16}$$

Thus, $\mathcal{P}_{i+1}$ satisfies **(PD1)**.

(ii) Let $\{x, y\} \in E(G_{i+1})$. We claim that some bag of $\mathcal{P}_{i+1}$ contains both x and y. By (16), $x \in \mathcal{V}_i$ or $x \in Y_S$, and $y \in \mathcal{V}_i$ or $y \in Y_S$. Let $x \in \mathcal{V}_i$ first. If $y \in \mathcal{V}_i$, then the claim follows from the induction hypothesis. If $y \in Y_S$, then $y \in \bar{X}_j$ for some $j \in \{1, \ldots, \bar{t}\}$, because, by **(G2d)**, $\bar{\mathcal{P}}$ is a path decomposition of $G[Y_S]$. Moreover, if $y \in Y_S$, then, by **(G2a)** $x \in \delta_G(G_i)$, and, by **(G2b)**, $x \in \bar{S}$. Hence, by (15), the $(\ell_i + j)$-th bag of $\mathcal{P}_{i+1}$ contains both x and y. Let now $x \in Y_S$. The case when $y \in \mathcal{V}_i$ follows by the symmetry. Hence, $y \in Y_S$ and the claim follows from the fact that $\bar{\mathcal{P}}$ is a path decomposition of $G[Y_S]$. This proves condition **(PD2)** for $\mathcal{P}_{i+1}$.

(iii) To prove **(PD3)** for $\mathcal{P}_{i+1}$, we prove the equivalent **(PD3′)** instead. Let $x \in \mathcal{V}_{i+1}$. By (16), $x \in Y_S$ or $x \in \mathcal{V}_i$. If $x \in Y_S = \mathrm{span}(\bar{\mathcal{P}})$, then **(PD3′)** follows from (15), the fact that $\bar{\mathcal{P}}$ is a path decomposition of $G[Y_S]$ and Y_S is disjoint from $\mathcal{V}_i \cup \bar{S}$. The latter is due to **(G2a)**, **(G2b)** and **(G2c)**.
Thus, let $x \in \mathcal{V}_i$. If $x \in \mathcal{V}_i \setminus \bar{S}$, then by (15), **(PD3′)** follows from the induction hypothesis for $\mathcal{P}_i$. It remains to consider $x \in \bar{S}$. Then, by **(G2b)**, $\bar{S} \subseteq X_i$. Thus, $x \in X_i$. Now, let j be the smallest index in $\{1, \ldots, \mathrm{len}(\mathcal{P}_i)\}$ such that x appears in the j-th bag of $\mathcal{P}_i$. Then, due to the induction hypothesis, x appears in each bag $X_{j'}$, $j' = j, \ldots, \mathrm{len}(\mathcal{P}_i)$. Moreover, by (15), $\bar{S}$ is included in each bag $X_{j'}$, $j' = \mathrm{len}(\mathcal{P}_i) + 1, \ldots, \mathrm{len}(\mathcal{P}_i) + \bar{t}$, thus so is $x \in \bar{S}$. Therefore, x appears in each bag $X_{j'}$, $j' = j, \ldots, \mathrm{len}(\mathcal{P}_i) + \bar{t}$ which proves **(PD3′)**.

To complete the proof we need to show that each $x \in \delta_G(G_{i+1})$ belongs to the last bag of $\mathcal{P}_{i+1}$. To that end we observe that by (15) and by $\mathcal{V}_i \subseteq \mathcal{V}_{i+1}$,

$$\delta_G(G_{i+1}) \subseteq \delta_G(G_i) \cup Y_S \cup \bar{S}. \tag{17}$$

By **(G2b)**, $\delta_G(G_i) = \bar{S}$. We now argue that $x \notin Y_S$. Suppose for a contradiction that $x \in Y_S$. The set Y_S contains only the vertices of S-components. Therefore, any neighbor of x belongs either to Y_S or to S, and thus belongs to $\mathcal{V}_{i+1}$ (observe that by **(G2)** $S \subseteq \bar{S}$). Hence, $x \notin \delta_G(G_{i+1})$ which leads to a contradiction. This proves that $x \in \delta_G(G_{i+1}) \subseteq \bar{S}$ and thus x appears in the last bag of $\mathcal{P}_{i+1}$.

MOST WIEDZY Downloaded from mostwiedzy.pl

Finally, it follows from the induction hypothesis and condition **(G2d)** that every bag of $\mathcal{P}_{i+1}$ has size at most $k+1$. Moreover, $w(v_i, v_{i+1}) = \bar{t}$. Thus, $\ell_{i+1} = \ell_i + \bar{t}$. On the other hand, by (15), $\text{len}(\mathcal{P}_{i+1}) = \text{len}(\mathcal{P}_i) + \bar{t}$. By the induction hypothesis, $\text{len}(\mathcal{P}_i) = \ell_i$. Thus, $\text{len}(\mathcal{P}_{i+1}) = \ell_{i+1}$, which completes the proof of Case 2. □

Lemma 3.22. *Let $k \in \{1, 2, 3\}$. Let G be a connected graph and let $\ell > 0$. If there exists a path decomposition of width at most k and length ℓ for G, then there exists an s–t path P of weighted length at most ℓ in $\mathcal{G}_k$.*

Proof. Let $\mathcal{P} = (X_1, \ldots, X_\ell)$ be a path decomposition of G with $\text{width}(\mathcal{P}) \le k$. By Observation 3.8, we may assume that $\mathcal{P}$ is a clean path decomposition. Let $\mathcal{P}_i = (X_1, \ldots, X_i)$ for each $i \in \{1, \ldots, \ell\}$ and let for brevity $\mathcal{P}_0$ be an empty list. For every $S \in \mathcal{S}$, let $\mathbf{G}_S$ be the set of green S-branches. Note that all vertices of each S-branch in $\mathbf{G}_S$ are in $X_{t_1(S)} \cup \cdots \cup X_{t_2(S)}$. For $S \subseteq V(G)$ such that $|S| \le 4$ and $S \notin \mathcal{S}$, set $\mathbf{G}_S := \emptyset$. Furthermore, for a non-empty $S \subseteq X_i$, let l_S^i be the number of S-leaves in $\text{span}(\mathcal{P}_i)$, $i = 1, \ldots, \ell$. Finally, for every non-empty $S \subseteq X_i$, $i = 1, \ldots, \ell$, and for each $H \in \mathcal{C}_G^2(S)$ define the function f_S^i as follows:

$$f_S^i(H) = \begin{cases} 1 & \text{if } V(H) \subseteq \text{span}(\mathcal{P}_i); \\ 0 & \text{otherwise.} \end{cases}$$

Now, let $\prec$ be the partial order on $\mathcal{S}$ defined by $S \prec S'$ if and only if $I(S) \subseteq I(S')$, where $S, S' \in \mathcal{S}$. Let $S_1, \ldots, S_p$ be all maximal elements of $\prec$. By Lemma 3.18, the sets $I(S_q)$ for $q = 1, \ldots, p$ are pairwise disjoint. Thus, without loss of generality, we assume that $1 \le t_1(S_1) \le t_2(S_1) < t_1(S_2) \le t_2(S_2) < \ldots < t_1(S_p) \le t_2(S_p) \le \ell$.

Define

$$I = \{0, \ldots, \ell\} \setminus \bigcup_{i=1,\ldots,p} \{t_1(S_i), \ldots, t_2(S_i) - 1\}$$

and denote $I = \{s_1, \ldots, s_r\}$, where $s_1 < s_2 < \cdots < s_r$. Observe that, by Lemma 3.18, for each $i \in I$ and for each $S \subseteq X_i$, $S \ne \emptyset$, the function f_S^i satisfies the condition in (10). Thus, the following is a sequence of vertices of $\mathcal{G}_k$: $P = s\text{–}(X_{s_1}, R_{s_1})\text{–}(X_{s_2}, R_{s_2})\text{–}\ldots\text{–}(X_{s_r}, R_{s_r})\text{–}t$, where f_S^i and l_S^i are used in each R_i, $i \in I$, i.e., $R_i(S) = (\mathbf{G}_S, f_S^i, l_S^i)$ for each non-empty $S \subseteq X_i$. Denote $s = (\emptyset, \emptyset) = (X_0, R_0)$. In the reminder of the proof, we show how to obtain an s–t path of length ℓ in $\mathcal{G}_k$ from the sequence P.

We first prove that $G_{\mathcal{P}_i} = G[\mathcal{V}_{v_i}]$ and $\delta(\mathcal{P}_i) = \delta_G(G[\mathcal{V}_{v_i}])$, where $v_i = (X_i, R_i)$, for each $i \in I$. By definition, $G[\mathcal{V}_{v_\ell}] = G_{\mathcal{P}} = G$ and $V(G[\mathcal{V}_s]) = \emptyset = V(G_{\mathcal{P}_0})$ and hence let $0 < i < \ell$ in the following. We have $\delta(\mathcal{P}_i) \subseteq X_i$ by **(PD2)** and **(PD3)**. Moreover, $\delta(\mathcal{P}_i) \ne \emptyset$ since G is connected and $i < l$. Set $S = \delta(\mathcal{P}_i)$. Then, for each connected component H in $G - S$, either $V(H) \subseteq \text{span}(\mathcal{P}_i)$ or $V(H) \cap \text{span}(\mathcal{P}_i) = \emptyset$. Otherwise, $V(H) \cap \delta(\mathcal{P}_i) \ne \emptyset$, which is in contradiction with H being a component in $G - S$. Then, if H is a connected component in $G - S$ but not an S-component such that $V(H) \subseteq \text{span}(\mathcal{P}_i)$, then there exists a non-empty $S' \subsetneq S$ such that H is an S'-component. Again, for this S'-component either $V(H) \subseteq \text{span}(\mathcal{P}_i)$ or $V(H) \cap \text{span}(\mathcal{P}_i) = \emptyset$. This proves that $G_{\mathcal{P}_i} = G[\mathcal{V}_{v_i}]$ which implies $\delta(\mathcal{P}_i) = \delta_G(G[\mathcal{V}_{v_i}])$ for each $i \in I$.

Let $i \in I$ be selected in such a way that $i + 1 \in I$. Now we argue that $((X_i, R_i), (X_{i+1}, R_{i+1})) \in E(\mathcal{G}_k)$. Clearly, $\text{span}(\mathcal{P}_i) \subseteq \text{span}(\mathcal{P}_{i+1})$ and $\text{span}(\mathcal{P}_{i+1}) \setminus \text{span}(\mathcal{P}_i) \subseteq X_{i+1}$. Hence, $\mathcal{V}_{v_i} \subseteq \mathcal{V}_{v_{i+1}}$ and $\mathcal{V}_{v_{i+1}} \setminus \mathcal{V}_{v_i} \subseteq X_{i+1}$. By **(PD2)** and **(PD3)**, $\delta(\mathcal{P}_i) \subseteq X_{i+1} \cap \text{span}(\mathcal{P}_i)$. Thus, since $\mathcal{P}$ is a clean path decomposition, $\delta(\mathcal{P}_i) = X_{i+1} \cap \text{span}(\mathcal{P}_i)$. This implies that $\delta_G(G[\mathcal{V}_{v_i}]) = \delta(\mathcal{P}_i) = X_{i+1} \cap \mathcal{V}_{v_i}$. Therefore, we just proved that there is a step edge from (X_i, R_i) to (X_{i+1}, R_{i+1}) in $\mathcal{G}_k$.

We now consider $i \in I$ and $j \in I$ such that $j > i + 1$ and $\{i + 1, \ldots, j - 1\} \cap I = \emptyset$. Hence, there exists $q \in \{1, \ldots, p\}$ such that $t_1(S_q) = i + 1$ and $t_2(S_q) = j$. By Lemma 3.17,

$$X_{i+1} \cup \cdots \cup X_j = (X_i \cap X_j) \cup V(\mathbf{G}_{S_q}) \cup \Lambda, \tag{18}$$

where Λ is the set of all S_q-leaves in $X_{i+1}, \ldots, X_j$. Let $Y = V(\mathbf{G}_{S_q}) \cup \Lambda$ for convenience. We claim that there is a jump edge between (X_i, R_i) and $(X_j \setminus Y, R_j)$ in $\mathcal{G}_k$. Note that $S_q \subseteq X_i \cap X_j$. By (18), $X_j \setminus Y = X_i \cap X_j$ thus **(G2c)** is met. By deleting the nodes other than those in Y from each bag $X_{i+1}, \ldots, X_j$ we obtain a path decomposition of length $j - i$ for the union of S_q-branches in $\mathbf{G}_{S_q}$ and S_q-leaves in Λ. The width of this path decomposition is at most $k - |X_i \cap X_j|$ since $X_i \cap X_j$ belongs to each bag X_t, for $t = i + 1, \ldots, j$, and no vertex in $X_i \cap X_j$ belongs to Y. This implies condition **(G2d)**.

By Lemma 3.17, condition **(G2a)** is met. As shown earlier, $\delta(\mathcal{P}_i) = X_{i+1} \cap \text{span}(\mathcal{P}_i)$. By **(PD3)**, $X_i \cap X_j \subseteq X_{i+1}$, and by (18), $X_{i+1} \setminus Y \subseteq X_i \cap X_j$. Thus, $\delta(\mathcal{P}_i) = X_i \cap X_j \cap \text{span}(\mathcal{P}_i) = X_i \cap X_j = X_j \setminus Y = \delta_G(G[\mathcal{V}_{v_i}])$ and **(G2b)** is met.

Finally, clearly $\mathcal{V}_{v_j} \setminus \mathcal{V}_{v_i} \not\subseteq X_j \setminus Y$, because $Y \ne \emptyset$. Moreover, by **(G2b)**, $X_j \setminus Y = \delta_G(G[\mathcal{V}_{v_i}])$, which implies $((X_j \setminus Y) \setminus \delta_G(G[\mathcal{V}_{v_i}])) \cap \mathcal{V}_{v_i} = \emptyset$. Thus, we just proved that there is a jump edge from $(X_{t_1(S_q)-1}, R_{t_1(S_q)-1})$ to $(X_j \setminus Y, R_{t_2(S_q)})$ with $\bar{t} \le j - i$ in $\mathcal{G}_k$.

By definition of $\mathcal{G}_k$ and since $\ell > 0$ there is a directed edge (v_ℓ, t) in $\mathcal{G}_k$, and its weight is zero. Therefore, we have shown how to build an s–t path of weighted length at most ℓ in $\mathcal{G}_k$ from the sequence P. □

4. MLPD(k)-CONSTR for graphs with one big component, $k \le 3$

In this section and in Section 5, we apply the results from Section 3 to develop an algorithm for general graphs. We will do it in two steps. This section adapts the algorithm from Section 3 so that it can handle graphs that consist of one

MOST WIEDZY Downloaded from mostwiedzy.pl

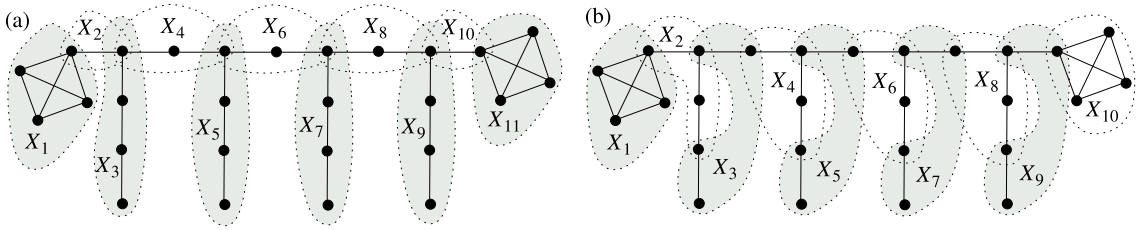

Fig. 7. (a) A path decomposition of length 11 of G; (b) minimum-length path decomposition of G.

component with more than two vertices and perhaps a number of isolated vertices and isolated edges. We call such a graph a *chunk graph*. Section 5 uses the algorithm for chunk graphs to obtain an algorithm for general graphs.

To be precise, let G be a graph. A connected component of G is called *big* if it has at least three vertices, and it is called *small* otherwise. Clearly, small components are either isolated vertices or isolated edges; we refer to them as K_1-components (isolated vertices) and K_2-components (isolated edges). A *chunk graph* is a graph that has exactly one big component.

Our polynomial-time algorithm for chunk graphs needs to meet additional restrictions on the sizes of the first and the last bags of the resulting path decompositions. Thus, we need to extend the path decomposition definition as follows. Let e_1, e_2 be integers. We call a path decomposition $\mathcal{P} = (X_1, \ldots, X_l)$ a (e_1, e_2)*-path decomposition* if $|X_1| \le e_1$ and $|X_l| \le e_2$. The main result of this section is:

Theorem 4.1. *Let $k \in \{1, 2, 3\}$.*

If: *for each $k' \in \{0, \ldots, k-1\}$, there exists a polynomial-time algorithm that, for any graph G either constructs a minimum-length path decomposition of width k' of G, or concludes that no such path decomposition exists,*

then: *there exists a polynomial-time algorithm that, for any given* ***chunk graph*** *C and for any $e_1, e_2 \in \{2, \ldots, k+1\}$, either constructs a minimum-length (e_1, e_2)-path decomposition of width at most k of C, or concludes that no such path decomposition exists.*

Here, by a minimum-length (e_1, e_2)-path decomposition of width at most k of C, we mean a path decomposition that is shortest among all (e_1, e_2)-path decompositions of width at most k of C.

We extend the notion of clean path decompositions to (e_1, e_2)-path decompositions. We say that a (e_1, e_2)-path decomposition $\mathcal{P}$ of G is *clean* if, among all (e_1, e_2)-path decompositions of G, it satisfies conditions **(C1)** and **(C2)** in Definition 3.7.

Before continuing with the algorithm, we want to point out that, in general, a minimum-length path decomposition of a chunk graph cannot be obtained by simply constructing a minimum-length path decomposition of its big component, filling up non-full bags (if any), and adding new bags containing the vertices of the small components. The following example illustrates this fact.

Example. Consider the following graph C, which is the disjoint union of the graph G in Fig. 7, two isolated edges, and three isolated vertices. Fig. 7(b) shows a minimum-length path decomposition of G (its length is 10). Since all bags are of size 4 in this decomposition, the small components must use two additional bags, resulting in a path decomposition of C of length 12. The verification that each bag is of size 4 in any minimum-length path decomposition of G is left for the reader. Now, consider a path decomposition of G of length 11 in Fig. 7(a). There are two bags of size 2 and three of size 3 in this decomposition which readily accommodates all small components resulting in a path decomposition of C of length 11. In other words, the addition of three K_1-components and two K_2-components does not increase the length of the path decomposition in Fig. 7(b) if those components are inserted properly, but it increases the length of the one in Fig. 7(b) by two.

4.1. An algorithm for chunk graphs

Let $1 \le k \le 3$ be an integer, let $2 \le e_1, e_2 \le k+1$ and let C be a chunk graph with a big connected component G, K_1-components $K_1^1, \ldots, K_1^{q_1}$ and K_2-components $K_2^1, \ldots, K_2^{q_2}$. Let $\mathcal{G}_k$ be the auxiliary graph for G defined in Section 3.5. For every $v \in V(\mathcal{G}_k) \setminus \{t\}$, define $W(v) = \{(v, i, j) \mid i \in \{0, \ldots, q_1\}, j \in \{0, \ldots, q_2\}\}$. Now define the auxiliary graph $\mathcal{H}_k = \mathcal{H}_k(e_1, e_2)$ corresponding to C as follows. The vertex set of $\mathcal{H}_k$ is given by

$$V(\mathcal{H}_k) = \{t'\} \cup \bigcup_{v \in V(\mathcal{G}_k) \setminus \{t\}} W(v).$$

Thus, $V(\mathcal{H}_k)$ is constructed from $\mathcal{G}_k$ by expanding every vertex v of $\mathcal{G}_k$ (except for the sink t) to a set of vertices $W(v)$ that includes all possible additions of small components to v. Let for brevity $s' = (s, 0, 0)$. Notice that $s' \in V(\mathcal{H}_k)$. For any $x = (v, i, j) \in W(v)$, where $v = (X, R) \in V(\mathcal{G}_k) \setminus \{t\}$, define

MOST WIEDZY Downloaded from mostwiedzy.pl

$$\mathcal{V}'_x = \mathcal{V}_v \cup \bigcup_{1 \le p \le i} V(K_1^p) \cup \bigcup_{1 \le p \le j} V(K_2^p).$$

The interpretation of this set is as follows. Every path P in $\mathcal{H}_k$ from s' to x represents partial path decompositions of C. These partial decompositions cover i isolated vertices, j isolated edges and the vertices in $\mathcal{V}_v$, and they can be extracted from the consecutive vertices and edges of P. The construction of $\mathcal{H}_k$ will guarantee that each of these partial path decompositions covers exactly those vertices of C that are in $\mathcal{V}'_x$.

Now, let us define the edge set of $\mathcal{H}_k$. First, for any $x \in V(\mathcal{H}_k)$, define $\theta(x) = e_1$ if $x = s'$, and $\theta(x) = k+1$ otherwise. Similarly, for any $x \in V(\mathcal{H}_k)$, let $\eta(x) = e_2$ if $\mathcal{V}'_x = V(C)$, and $\eta(x) = k+1$ otherwise. There are four types of directed edges in $\mathcal{H}_k$:

(H1) Edges from a vertex in $W(v)$ to a vertex in $W(u)$ where (u, v) is a step edge in $\mathcal{G}_k$.
Let $u, v \in V(\mathcal{G}_k) \setminus \{t\}$ such that (u, v) is a step edge in $\mathcal{G}_k$, and let $x \in W(u)$ and $y \in W(v)$. Let $a = |\delta_G(G[\mathcal{V}_u]) \cup (\mathcal{V}'_y \setminus \mathcal{V}'_x)|$. Then, $(x, y) \in E(\mathcal{H}_k)$ if and only if $\mathcal{V}'_x \subseteq \mathcal{V}'_y$, $a \le \theta(x)$ and $a \le \eta(y)$. The weight of each edge of this type is $w'(x, y) = 1$.

(H2) Edges from a vertex in $W(v)$ to a vertex in $W(u)$ where (u, v) is a jump edge in $\mathcal{G}_k$.
Let $u, v \in V(\mathcal{G}_k) \setminus \{t\}$ such that (u, v) is a jump edge in $\mathcal{G}_k$, and let $x = (u, i_u, j_u) \in W(u)$ and $y = (v, i_v, j_v) \in W(v)$ be such that $\mathcal{V}'_x \subseteq \mathcal{V}'_y$. Denote $u = (X_u, R_u)$ and $v = (X_v, R_v)$. Let $\mathcal{Q}'_{xy} = (\bar{Y}_1, \ldots, \bar{Y}_{\bar{z}})$ be a path decomposition of width $\mathrm{width}(\mathcal{Q}'_{xy}) \le k - |X_u \cap X_v|$ of the graph $\bar{C} = \bar{G} \cup \bigcup_{i_u < p \le i_v} K_1^p \cup \bigcup_{j_u < p \le j_v} K_2^p$, where $\bar{G}$ is defined in **(G2d)**. Then, $(x, y) \in E(\mathcal{H}_k)$ and the weight of this edge is $w'(x, y) = \bar{z}$, where $\bar{z}$ is taken as small as possible, i.e., $\mathcal{Q}'_{xy}$ is of minimum length. We refer to any path decomposition $(\bar{Y}_1 \cup (X_u \cap X_v), \ldots, \bar{Y}_{\bar{z}} \cup (X_u \cap X_v))$ as a *witness* of the jump edge.

(H3) Edges inside $W(v)$.
Let $x, y \in W(v)$ for some $v \in V(\mathcal{G}_k) \setminus \{t\}$, and let $a = |\delta_G(G[\mathcal{V}_v]) \cup (\mathcal{V}'_y \setminus \mathcal{V}'_x)|$. Then, $(x, y) \in E(\mathcal{H}_k)$ if and only if $\mathcal{V}'_x \subsetneq \mathcal{V}'_y$, $a \le \theta(x)$ and $a \le \eta(y)$. The weight of each edge of this type is $w'(x, y) = 1$.

(H4) Edges from a vertex in $W(v)$ to t'.
For $x \in V(\mathcal{H}_k) \setminus \{t'\}$, let $(x, t') \in E(\mathcal{H}_k)$ if and only if $\mathcal{V}'_x = V(C)$. The weight of such edge is $w'(x, t') = 0$.

The parameter $\theta(x)$ guarantees that the first bag of the path decomposition that corresponds to an s'–t' in $\mathcal{H}_k$ path has the size at most e_1. The restriction imposed by $\theta(x)$ is vacuous for $x \ne s'$. Similarly, the parameter $\eta(y)$ guarantees that the last bag of the path decomposition that corresponds to an s'–t' path has the size at most e_2. This restriction imposed by $\eta(y)$ is vacuous for $\mathcal{V}'_y \ne V(C)$.

We also remark that our construction is fairly general, in the sense that one may argue that certain vertices and certain edges of $\mathcal{H}_k$ can never be a part of a shortest s'–t' path in $\mathcal{H}_k$. However, we proceed with this construction to avoid tedious analysis of special cases.

Before we continue with a formal analysis we give an intuition on the edge set of $\mathcal{H}_k$. We use the step edges (u, v) of $\mathcal{G}_k$ by adding, whenever possible, some vertices of small components to the bag that corresponds to v. For a jump edge of $\mathcal{G}_k$, we recalculate the path decomposition $\bar{\mathcal{P}}$ used in **(G2b)** in such a way that the new path decomposition $\mathcal{Q}'_{xy}$ in **(H2)** covers the vertices in $\mathrm{span}(\bar{\mathcal{P}})$ and the vertices of some small components. (If no small component vertices are added, then one may take $\mathcal{Q}'_{xy} = \bar{\mathcal{P}}$.) Then, **(H3)** allows the vertices of K_1- and K_2-components only to fill in a bag that corresponds to a vertex of $\mathcal{H}_k$. In particular, the edges with $v = s$ introduce bags prior to $\alpha(G)$ and those with $\mathcal{V}_v = V(G)$ introduce bags after $\beta(G)$ — see proof of Lemma 4.3.

Before we prove the main result of this section, we start with a useful observation:

Observation 4.2. *Let G' be a graph and let $\mathcal{P} = (X_1, \ldots, X_l)$ be a clean path decomposition of G'. Then, for every small component H of G', there exists a unique $i \in \{1, \ldots, l\}$ such that $X_i \cap V(H) \ne \emptyset$.* □

Lemma 4.3. *Let $k \in \{1, 2, 3\}$. Let $e_1, e_2 \in \{2, \ldots, k+1\}$ and let C be a chunk graph. There exists an s'–t' path in $\mathcal{H}_k$ of weighted length at most l' if and only if there exists a (e_1, e_2)-path decomposition of C of width at most k and length at most l'.*

Proof. Let G be the unique big component of C. First assume that there exists an s'–t' path $P' = x_0$–x_1–$\ldots$–$x_{m'}$–t' of weighted length l' in $\mathcal{H}_k$. Denote $x_i = (v_i, a_i, a'_i)$, $i = 0, \ldots, m'$, and $v_i = (X_i, R_i)$. Let $\mathcal{G}_k$ be the auxiliary graph for G defined in Section 3.5. Notice that $v_0 = s$, the source of $\mathcal{G}_k$. Let for brevity $v_{m'+1} = t$, the sink of $\mathcal{G}_k$. From definition of $E(\mathcal{H}_k)$ it follows that by replacing any maximal subsequence v_r–$\ldots$–$v_{r'}$ such that $v_r = \cdots = v_{r'}$ in v_0–$\ldots$–$v_{m'}$–$v_{m'+1}$ (all $x_r, x_{r+1}, \cdots, x_{r'}$ belong to $W(v_r)$) by the single vertex v_r, we obtain an s–t path $P = u_0$–$\ldots$–u_m–u_{m+1}, where $u_0 = s$ and $u_{m+1} = t$, in $\mathcal{G}_k$.

Now, construct $\mathcal{P} = (\mathcal{R}_1, \ldots, \mathcal{R}_m)$ and $\mathcal{P}' = (\mathcal{R}'_1, \ldots, \mathcal{R}'_{m'})$ for the paths P' and P as follows. For each $i = 1, \ldots, m'$:

(1) If $(v_{i-1}, v_i) = (u_{j-1}, u_j)$ is a step edge in $\mathcal{G}_k$, then set $\mathcal{R}'_i = (X_i)$, and $\mathcal{R}_j = (X_i)$.

MOST WIEDZY Downloaded from mostwiedzy.pl

Algorithm 1 Finding a minimum-length (e_1, e_2)-path decomposition of a chunk graph C.

Input: A chunk graph C, $k \in \{1, 2, 3\}$ and $e_1, e_2 \in \{2, 3, 4\}$.
Output: A minimum-length (e_1, e_2)-path decomposition of C of width k or 'failure' if $\mathrm{pw}(C) > k$.
Let G be the big component of C.
if $\mathrm{pw}(G) > k$ (use the algorithm in [6] to calculate $\mathrm{pw}(G)$) **then**
  return 'failure'.
Construct the auxiliary graph $\mathcal{H}_k$.
Find a shortest s'–t' path P in $\mathcal{H}_k$.
Use P to construct the corresponding path decomposition $\mathcal{P}$ of C of the same length as P.
return $\mathcal{P}$.

(2) If $(v_{i-1}, v_i) = (u_{j-1}, u_j)$ is a jump edge in $\mathcal{G}_k$, then let $(Y'_1, \ldots, Y'_{\bar{z}})$ be a witness of (v_{i-1}, v_i) with $w(v_{i-1}, v_i) = \bar{z}$. Set $\mathcal{R}'_i = (Y'_1, \ldots, Y'_{\bar{z}})$, and $\mathcal{R}_j = (Y'_1 \cap V(G), \ldots, Y'_{\bar{z}} \cap V(G))$. Observe that $\mathcal{R}_j$ is a witness of (u_{j-1}, u_j), however, not necessarily with minimum length.
(3) If $v_{i-1} = v_i = u_j$, then set $\mathcal{R}'_i = \big(\delta_G(\mathcal{V}_{v_{j-1}}) \cup (\mathcal{V}'_{x_i} \setminus \mathcal{V}'_{x_{i-1}})\big)$.

It follows from the proof of Lemma 3.21 that $\mathcal{P}$ is a path decomposition of G of width at most k. Thus, by the construction of $\mathcal{P}'$ in (1-3) and the fact that P' is an s'–t' path in $\mathcal{H}_k$ it follows that $\mathcal{P}'$ is a path decomposition of C of width at most k and of length l'. Moreover, by definition of $\mathcal{H}_k$, the edges (x_0, x_1) and $(x_{m'-1}, x_{m'})$ are not jump edges in $\mathcal{H}_k$. Therefore, the functions θ and η in the definition of $\mathcal{H}_k$ guarantee that the first bag and the last bag of $\mathcal{P}'$ have sizes at most e_1 and e_2, respectively. Thus, $\mathcal{P}'$ is a (e_1, e_2)-path decomposition of C.

For the converse, assume that $\mathcal{P}' = (X'_1, \ldots, X'_{l'})$ is a (e_1, e_2)-path decomposition of C with $\mathrm{width}(\mathcal{P}') \le k$. We may assume without loss of generality that $\mathcal{P}'$ is clean. For $i = 1, \ldots, l$, let a_i and b_i the numbers of vertices of K_1- and K_2-components respectively in $\bigcup_{j=1}^{i} X'_j$. By Observation 4.2, all vertices of a small component are either in $\bigcup_{j=1}^{i} X'_j$ or in $\bigcup_{j=i+1}^{l'} X'_j$. Consider $\mathcal{P} = (X'_1 \cap V(G), \ldots, X'_{l'} \cap V(G))$. Let $Q = \{(p, q), 1 \le p < q \le l'\}$ be the set of all pairs such that the sequence $X'_p \cap V(G) = \cdots = X'_q \cap V(G)$ is maximal. Delete all $X'_{p+1} \cap V(G), \ldots, X'_q \cap V(G)$ from $\mathcal{P}$ for each pair $(p, q) \in Q$. The resulting path decomposition $\mathcal{P}'' = (Y_1, \ldots, Y_l)$ of G, $l \le l'$, is clean for $\mathcal{P}'$ is clean. Let $h(i)$, $i = 1, \ldots, l$, be such that $Y_i = X'_{h(i)} \cap V(G)$. By the proof of Lemma 3.22, there exists an s–t path $P = s$–v_1–$\cdots$–v_r–t of weighted length l in the auxiliary graph $\mathcal{G}_k$ for G, and there are integers $1 \le s_1 < \cdots < s_r \le l$ such that if $s_{i+1} - s_i = 1$, then (v_i, v_{i+1}) is a step edge in $\mathcal{G}_k$, and if $s_{i+1} - s_i > 1$, then (v_i, v_{i+1}) is a jump edge in $\mathcal{G}_k$ with $(Y_{s_i+1}, \ldots, Y_{s_{i+1}})$ being its witness. Define $w_i = (v_i, a_{h(i)}, b_{h(i)})$ for $i = 1, \ldots, r$. For each $(p, q) \in Q$, let v_i be such that $h(i) = p$. Replace $w_i = (v_i, a_p, b_p)$ by $w_i, w_i^1 = (v_i, a_{p+1}, b_{p+1}), \ldots, w_i^{q-p} = (v_i, a_q, b_q)$ in P. All these new vertices are in $W(v_i)$ and the edges between them in $E(\mathcal{H}_k)$ by **(H3)**. Let u_1–$\cdots$–$u_{l''}$ be the resulting sequence, where $u_i = (v'_i, a_i, b_i)$ for each $i \in \{1, \ldots, l''\}$. Clearly, if $v'_i \ne v'_{i+1}$, then (v'_i, v'_{i+1}) is either a step edge or a jump edge in $\mathcal{G}_k$. Moreover, if (v'_i, v'_{i+1}) is a jump edge, then $(v'_i, v'_{i+1}) = (v_j, v_{j+1})$ for some j and $(X'_{s_j+1}, \ldots, X'_{s_{j+1}})$ is a witness of (v_j, v_{j+1}). Thus, by definition of $V(\mathcal{H}_k)$, **(H1)**, and **(H2)** the edges (u_i, u_{i+1}) belong to $E(\mathcal{H}_k)$. Therefore, $P' = s'$–u_1–$\cdots$–$u_{l''}$–t' is an s'–t' path in $\mathcal{H}_k$. The path is not longer than l' since there is one to one correspondence between the jump edges in P and in P' and their witnesses are of the same length $s_{i+1} - s_i$. □

We conclude this section with a formal statement of the algorithm for computing a minimum-length (e_1, e_2)-path decomposition of a chunk graph C — see Algorithm 1. Note that $\mathrm{pw}(C) > 0$, because C is assumed to contain a big component G.

Note that $\mathrm{pw}(G) \le k$ if and only if $\mathrm{pw}(C) \le k$. We have $|W(v)| \le (1+q_1)(1+q_2)$ for each $v \in V(\mathcal{G}_k)$ and hence $|V(\mathcal{H}_k)| = O(q_1 q_2 |V(\mathcal{G}_k)|)$. Finally, Theorem 3.1 and Lemma 4.3 imply Theorem 4.1.

5. MLPD(k)-CONSTR for general graphs, $k \le 3$

In Section 4 we developed an algorithm that finds a minimum-length (e_1, e_2)-path decomposition for a chunk graph. We use this algorithm as a subroutine to obtain a polynomial-time algorithm for general graphs in this section. The key idea of the algorithm for a general graph G with $\mathrm{pw}(G) \le 3$ is as follows. We look at a graph G as a disjoint union of chunk graphs $C^1, \ldots, C^c$. Each C^i consists of a big component G_i and possibly some K_1- and K_2-components. We show in Lemma 5.3 that we can limit ourselves to minimum-length path decompositions of width $k \le 3$ for G where big components are never 'processed in parallel' (see Section 5.1 for formal definitions). It is therefore natural to construct a minimum-length path decomposition of G by first constructing minimum-length path decompositions $\mathcal{Q}_1, \ldots, \mathcal{Q}_c$ of the chunk graphs $C^1, \ldots, C^c$, respectively, and then by sequencing them one after another in some order, and finally by concatenating consecutive path decompositions. Two crucial issues need to be resolved however by this approach. The first consists in how many K_1- and K_2-components to add to G_i to make up a chunk graph C^i, $i = 1, \ldots, c$. This issue is resolved by a dynamic program given in Subsection 5.4. The second issue consists in how to sequence and concatenate $\mathcal{Q}_1, \ldots, \mathcal{Q}_c$ for given chunk graphs $C^1, \ldots, C^c$, so that the resulting decomposition of G has minimum length for the given $\mathcal{Q}_1, \ldots, \mathcal{Q}_c$. The latter is illustrated for a given order of $\mathcal{Q}_1, \ldots, \mathcal{Q}_c$ as follows. For a graph G which breaks up into two chunk graphs C^1 and C^2 we can concatenate $\mathcal{Q}_1$ and $\mathcal{Q}_2$ by taking first the bags of $\mathcal{Q}_1$ and then the bags of $\mathcal{Q}_2$. However, if the last bag of $\mathcal{Q}_1$ and the first bag of $\mathcal{Q}_2$ both have size two, then we can save one bag by replacing the last bag of $\mathcal{Q}_1$ and the first bag of $\mathcal{Q}_2$ by their union. In general,

MOST WIEDZY Downloaded from mostwiedzy.pl

we can save some bags by placing $\mathcal{Q}_1, \ldots, \mathcal{Q}_c$ in an appropriate order and by applying an appropriate concatenation. The order is dealt with in Subsection 5.3, and the concatenation in Subsection 5.2.

The main result of this section is the following theorem that essentially reduces MLPD(k)-CONSTR for general graphs to MLPD(k)-CONSTR for chunk graphs, $k = 1, 2, 3$.

Theorem 5.1. *Let $k \in \{1, 2, 3\}$.*

If: there exists a polynomial-time algorithm that, for any chunk graph C, and for any $e_1, e_2 \in \{2, \ldots, k+1\}$, either constructs a minimum-length (e_1, e_2)-path decomposition of width k of C, or concludes that no such path decomposition exists,

*then: there exists a polynomial-time algorithm that, for **any** graph G, either constructs a minimum-length path decomposition of width k of G, or concludes that no such path decomposition exists.*

Since it is straightforward to obtain a minimum-length path decomposition for graphs with no big components, we assume that the input graph has at least one big component.

5.1. Avoiding parallel processing of big components

We start with a definition of parallel processing of big components.

Definition 5.2. Let G be a graph and let $\mathcal{P} = (X_1, \ldots, X_l)$ be a path decomposition of G. We say that two big connected components G_1 and G_2 of G are *processed in parallel* if

$$|\{\alpha_{\mathcal{P}}(G_1), \ldots, \beta_{\mathcal{P}}(G_1)\} \cap \{\alpha_{\mathcal{P}}(G_2), \ldots, \beta_{\mathcal{P}}(G_2)\}| \geq 2.$$

The main result of this subsection, Lemma 5.3, shows that when constructing a minimum-length path decomposition of width k, $k \leq 3$, we may limit ourselves to path decompositions with no two big components processed in parallel. Notice that the NP-completeness proof of Section 2 shows that a minimum-length path decomposition of width 4 may require parallel processing of big components. Thus, the parallel processing of big components is one of the main features distinguishing (in terms of the computational complexity) between the problems MLPD(k)-CONSTR, $k \leq 3$, and the problems MLPD(k)-CONSTR, $k \geq 4$.

Lemma 5.3. *Let G be a graph and let $k \in \{1, 2, 3\}$. If $\mathrm{pw}(G) \leq k$, then there exists a minimum-length path decomposition of width k of G such that no two big connected components of G are processed in parallel.*

Proof. For any path decomposition $\mathcal{P} = (X_1, \ldots, X_l)$ of G, let $\zeta(\mathcal{P})$ be the smallest $t \in \{1, \ldots, l\}$ such that two big components are processed in parallel in step t, i.e., both components have a non-empty intersection with X_t and with X_{t+1} (set $\zeta(\mathcal{P}) = \infty$ if no such t exists). Assume without loss of generality that $\mathcal{P}$ is a minimum-length path decomposition of width k of G with maximum $\zeta(\mathcal{P})$. We will show that no two big components are processed in parallel in $\mathcal{P}$, by showing that if there are two such components, i.e., if $\zeta(\mathcal{P}) \neq \infty$, then we can increase ζ without increasing the length of the path decomposition, contrary to our choice of $\mathcal{P}$. This process is done in two stages. First, we construct a longer path decomposition $\mathcal{P}'$ from $\mathcal{P}$. Next, we show how $\mathcal{P}'$ can be shortened to a path decomposition $\mathcal{P}''$ of length exactly $\mathrm{len}(\mathcal{P})$, but with a larger value of ζ.

So suppose that there are two big connected components, say G_1 and G_2, that are processed in parallel in $\mathcal{P}$. Let $\{t_1, \ldots, t_2\} = \{\alpha(G_1), \ldots, \beta(G_1)\} \cap \{\alpha(G_2), \ldots, \beta(G_2)\}$. We may assume that G_1 and G_2 are chosen so that $t_1 = \zeta(\mathcal{P})$ and $\beta(G_1) \leq \beta(G_2)$. Let $q = t_2 - t_1 + 1$. By Definition 5.2, $q \geq 2$. For notational convenience, define for $i = 1, \ldots, q$, $Y_i = X_{t_1+i-1} \cap V(G_1)$ and $Z_i = X_{t_1+i-1} \setminus V(G_1)$. Observe that each of the sets Y_i and Z_i is non-empty, and $|Y_i| + |Z_i| = |Y_i \cup Z_i| \leq k+1$. Now define a sequence (of subsets of $V(G)$) $\mathcal{P}'$ and then we prove that $\mathcal{P}'$ is a path decomposition of G of width k and of length $l + q$. We also introduce the two following partial path decompositions $\mathcal{Q}_1$ and $\mathcal{Q}_2$:

$$\mathcal{P}' = \begin{cases} (\underbrace{Y_1, \ldots, Y_q}_{:=\mathcal{Q}_1}, \quad \underbrace{X_1, \ldots, X_{t_1-1}, \quad Z_1, \ldots, Z_q, \quad X_{t_2+1}, \ldots, X_l}_{:=\mathcal{Q}_2}) & \text{if } \alpha(G_1) \geq \alpha(G_2); \\ (\underbrace{X_1, \ldots, X_{t_1-1}, \quad Y_1, \ldots, Y_q}_{:=\mathcal{Q}_1}, \quad \underbrace{Z_1, \ldots, Z_q, \quad X_{t_2+1}, \ldots, X_l}_{:=\mathcal{Q}_2}) & \text{if } \alpha(G_1) < \alpha(G_2). \end{cases}$$

We claim that $\mathcal{P}'$ is a path decomposition of G. First suppose that $\alpha(G_1) \geq \alpha(G_2)$. Then $t_1 = \alpha(G_1)$ and $t_2 = \beta(G_1)$. Therefore, by Observation 1.2, $V(G_1) \cap X_t \neq \emptyset$ if and only if $t_1 \leq t \leq t_2$. Thus, $\mathcal{Q}_1$ is a path decomposition of G_1, and $\mathcal{Q}_2$ is a path

MOST WIEDZY Downloaded from mostwiedzy.pl

decomposition of $G - G_1$. Therefore, $\mathcal{P}$ is a path decomposition of G. Next, suppose that $\alpha(G_1) < \alpha(G_2)$. Clearly, $\mathcal{P}'$ satisfies **(PD1)**. Also, since there is no edge between $x \in Y_i$ and $y \in Z_i$ in G, and $\mathcal{P}$ is a path decomposition of G, it follows that **(PD2)** is met by $\mathcal{P}'$. Finally, to show that $\mathcal{P}'$ satisfies **(PD3)**, we argue that $A = (X_1 \cup \cdots \cup X_{t_1-1}) \cap (Z_1 \cup \cdots \cup Z_q \cup X_{t_2+1} \cup \cdots \cup X_l) = \emptyset$.

Suppose for a contradiction that $A \neq \emptyset$ and let $x \in A$ be selected arbitrarily. Since x appears in at least two bags of $\mathcal{P}$, by Observation 4.2, x belongs to a big component G'. However, $G_2 \neq G'$ since $\alpha(G_2) = t_1$, and $G_1 \neq G'$ since $\beta(G_1) = t_2$. Therefore, G_1 and G' are two big components that are processed in parallel starting at $t' < t_1$, contrary to our choice of G_1 and G_2. Hence, $A = \emptyset$. Moreover, we observe that since $\beta(G_1) = t_2$, it follows that there is no x such that $x \in Y_1 \cup \cdots \cup Y_q$ and $x \in X_{t_2+1} \cup \cdots \cup X_l$. Thus, **(PD3)** is met by $\mathcal{P}'$. Therefore, $\mathcal{P}'$ is a path decomposition of G.

We now describe an algorithm that takes the path decomposition $\mathcal{P}'$ as an input and returns a path decomposition $\mathcal{P}''$ of length exactly $\mathrm{len}(\mathcal{P})$. We consider the case of $k = 3$, as the other cases are analogous. The algorithm uses the following two length decreasing operations that preserve the property that $\mathcal{P}'$ is a path decomposition:

- If $\mathcal{P}'$ has a bag Z_i or Y_i of cardinality one, then, due to Lemma 3.9, we may delete it.
- If $\mathcal{P}'$ has $s \geq 2$ consecutive non-empty bags $X_{t+1}, \ldots, X_{t+s}$ such that $|X_{t+1} \cup \ldots \cup X_{t+s}| \leq 4$, then we may replace $X_{t+1}, \ldots, X_{t+s}$ by one bag containing $X_{t+1} \cup \ldots \cup X_{t+s}$ (i.e., we merge the bags $X_{t+1}, \ldots, X_{t+s}$).

We define three types of subintervals of $\{1, \ldots, q\}$. For a subinterval $J = \{a, \ldots, b\}$, we say that

- J is of Type A if $|J| \geq 2$ and $|Y_j| = 2$ for all $j \in J$;
- J is of Type B if $|J| \geq 3$, $|Y_a| = |Y_b| = 2$, and $|Y_j| \in \{1, 3\}$ for all $j \in J \setminus \{a, b\}$;
- J is of Type C if $|Y_j| = 2$ for at most one index $j \in J$.

We now partition the interval $\{1, \ldots, q\}$ in the following way. Let $\mathcal{A}$ be the collection of all maximal subintervals J of $\{1, \ldots, q\}$ of type A. Next, let $\mathcal{B}$ be the collection of all maximal subintervals of $\{1, \ldots, q\} \setminus \bigcup \mathcal{A}$ of type B. Finally, let $\mathcal{C}$ be the collection of all maximal subintervals of $\{1, \ldots, q\} \setminus \bigcup(\mathcal{A} \cup \mathcal{B})$. By construction, the intervals in $\mathcal{C}$ are of type C. Moreover, the intervals in $\mathcal{A} \cup \mathcal{B} \cup \mathcal{C}$ form a partition of $\{1, \ldots, q\}$.

The length-reduction subroutine consists of using three different subroutines, one for each type A, B, C. These subroutines all work as follows. Given an interval $\{a, \ldots, b\}$, they take two partial path decompositions $(Y_a, \ldots, Y_b)$ and $(Z_a, \ldots, Z_b)$, and return two new partial path decompositions $(Y'_1, \ldots, Y'_{a'})$ and $(Z'_1, \ldots, Z'_{b'})$ that satisfy: $a' + b' = b - a + 1$, $\bigcup_{i=1}^{a'} Y'_i = \bigcup_{i=a}^{b} Y_i$, $\bigcup_{i=1}^{b'} Z'_i = \bigcup_{i=a}^{b} Z_i$, $Y_a \subseteq Y'_1$, $Y_b \subseteq Y'_{a'}$, $Z_a \subseteq Z'_1$, and $Z_b \subseteq Z'_{b'}$. Thus, we may replace the partial path decompositions $(Y_a, \ldots, Y_b)$ and $(Z_a, \ldots, Z_b)$ in $\mathcal{P}'$ by $(Y'_1, \ldots, Y'_{a'})$ and $(Z'_1, \ldots, Z'_{b'})$, respectively. After running the appropriate subroutine for each of the intervals and performing these replacements, we end up with a path decomposition of G of length exactly $\mathrm{len}(\mathcal{P})$. Let $J = \{a, \ldots, b\} \in \mathcal{A} \cup \mathcal{B} \cup \mathcal{C}$. We consider the following cases:

Case 1: J is of Type A. We use the following subroutine which decreases the length of $\mathcal{P}$ by $|J|$. Notice that, since $|Y_j| = 2$ for all $j \in J$, it follows that $|Z_j| \leq 2$ for all $j \in J$.

```
Set 𝒴 = 𝒵 = ∅.
if |J| is odd then
    Append Y_a ∪ Y_{a+1} ∪ Y_{a+2} to 𝒴.          (Merge Y_a, Y_{a+1}, and Y_{a+2})
    Append Z_a ∪ Z_{a+1} and Z_{a+2} to 𝒵.        (Merge Z_a and Z_{a+1})
    a' = a + 3.
else
    a' = a.
if a' < b then
    for j = a', a' + 2, ..., b − 1 do
        Append Y_j ∪ Y_{j+1} to 𝒴.                (Merge Y_j and Y_{j+1})
        Append Z_j ∪ Z_{j+1} to 𝒵.                (Merge Z_j and Z_{j+1})
```

To see that this subroutine is correct, observe that if Y_a, Y_{a+1}, Y_{a+2} all have cardinality 2, then, since G_1 is connected, by Lemma 3.9, $Y_a \cap Y_{a+1} \neq \emptyset$ and $Y_{a+1} \cap Y_{a+2} \neq \emptyset$. Hence $|Y_a \cup Y_{a+1} \cup Y_{a+2}| \leq 4$.

Case 2: J is of Type B. For each $i \in \{a+1, \ldots, b-1\}$, if $|Y_i| = 1$, then set $Y_i := \emptyset$; if $|Y_i| = 3$, then set $Z_i := \emptyset$. Next, let $p > a$ be the smallest index such that $Y_p \neq \emptyset$. Such a p exists because $Y_b \neq \emptyset$. Set $Y_a := Y_a \cup Y_p$, and $Y_p := \emptyset$. Since G_1 is connected, by Lemma 3.9, $Y_a \cap Y_p \neq \emptyset$, thus $|Y_a \cup Y_p| \leq 4$. Finally, let $r > a$ be the smallest index such that $Z_r \neq \emptyset$. Such r exists because $Z_b \neq \emptyset$. Set $Z_a := Z_a \cup Z_r$, and $Z_r := \emptyset$. By Lemma 3.9, $|Z_a \cup Z_r| \leq 4$. The removal of empty bags decreases the length of $\mathcal{P}'$ by exactly $|J|$.

Case 3: J is of Type C. Then, we use the following subroutine to decrease the length of $\mathcal{P}$ by $|J|$. We may assume without loss of generality that $|Y_b| \neq 2$ because otherwise we can run the subroutine backwards.

MOST WIEDZY Downloaded from mostwiedzy.pl

Set $i = a$ and $\mathcal{Y} = \mathcal{Z} = \emptyset$.
while $i \le b$ **do**
  if $|Y_i| = 1$ **then**
    Append Z_i to $\mathcal{Z}$ and set $i := i + 1$. (Remove Y_i)
  else if $|Y_i| = 3$ **then**
    Append Y_i to $\mathcal{Y}$ and set $i := i + 1$. (Remove Z_i)
  else if $|Y_i| = 2$ **then**
    if $|Y_{i+1}| = 1$ **then**
      Append Y_i to $\mathcal{Y}$. (Remove Y_{i+1})
      Append $Z_i \cup Z_{i+1}$ to $\mathcal{Z}$. (Merge Z_i and Z_{i+1})
    else if $|Y_{i+1}| = 3$ **then**
      Append $Y_i \cup Y_{i+1}$ to $\mathcal{Y}$. (Merge Y_i and Y_{i+1})
      Append Z_i to $\mathcal{Z}$. (Remove Z_{i+1})
    Set $i := i + 2$.

To see that this subroutine is correct, notice that if $|Y_i| = 2$, then $i < b$ by the assumption that $|Y_b| \neq 2$. Thus, Y_{i+1} and Z_{i+1} are well-defined. Moreover, because J is an interval of Type C, $|Y_{i+1}| \in \{1, 3\}$. Also notice that if $|Y_i| = 2$ and $|Y_{i+1}| = 1$, then $|Z_i| \le 2$ and $|Z_{i+1}| \le 3$. Since, due to Lemma 3.9, both Z_i and Z_{i+1} contain vertices from the connected component G_2, it follows that $|Z_i \cup Z_{i+1}| \le 4$.

From cases 1–3 we obtain that $\mathrm{len}(\mathcal{P}'') = \mathrm{len}(\mathcal{P}') - q$ as required. To see that $\zeta(\mathcal{P}'') > \zeta(\mathcal{P})$, observe that $\zeta(\mathcal{P}) = t_1$ and $\zeta(\mathcal{P}'') \ge t_1 + 1$. □

We remark that we obtain a stronger analogue of Lemma 5.3 for $k \in \{1, 2\}$.

Observation 5.4. *Let G be a graph and let $k \in \{1, 2\}$. If $\mathrm{pw}(G) \le k$, then there exists a minimum-length path decomposition $\mathcal{P}$ of width k of G such that for any two big components G_1 and G_2 of G it holds $\{\alpha_{\mathcal{P}}(G_1), \ldots, \beta_{\mathcal{P}}(G_1)\} \cap \{\alpha_{\mathcal{P}}(G_2), \ldots, \beta_{\mathcal{P}}(G_2)\} = \emptyset$.* □

We end with the following corollary that follows immediately from Lemma 5.3 and Lemma 3.9.

Corollary 5.5. *Let G be a graph and let $k \in \{1, 2, 3\}$. If $\mathrm{pw}(G) \le k$, then there exists a minimum-length path decomposition $\mathcal{P} = (X_1, \ldots, X_l)$ of width k of G such that for every big component G' of G, $|X_i \cap V(G')| \ge 2$ for $i = \alpha(G'), \ldots, \beta(G')$.* □

5.2. Type-optimal path decompositions of chunk graphs

We now deal with two issues alluded to earlier in Section 5. One is the appropriate concatenation of minimum-length path decompositions of given chunk graphs. The other is the appropriate selection of a minimum-length path decomposition for a given chunk graph. Note that the sizes of the first and the last bag of a minimum-length path decomposition of a chunk graph may impact the length of its subsequent concatenation with minimum-length path decompositions of other chunk graphs. By Observation 5.4, we obtain the following.

Observation 5.6. *Let G be a graph with $\mathrm{pw}(G) \le k$, $k \in \{1, 2\}$, and let c be the number of big connected components of G. There exist chunk graphs $C^1, \ldots, C^c$ such that $G = C^1 \cup \cdots \cup C^c$ and $\mathcal{P} = (\mathcal{Q}_1, \ldots, \mathcal{Q}_c)$ is a minimum-length path decomposition of width at most k of G, where $\mathcal{Q}_i$ is any minimum-length path decomposition of C^i, $i = 1, \ldots, c$.* □

Hence, we assume $k = 3$ for the reminder of this subsection. We first distinguish four types of path decompositions of chunk graphs. We say that a path decomposition $\mathcal{Q} = (X_1, \ldots, X_l)$ of a chunk graph C is of

- Type $\mathbb{A}$, if $|X_1| \le 2$ and $|X_l| \le 2$;
- Type $\mathbb{B}_1$, if $|X_1| \le 2$ and $|X_l| > 2$;
- Type $\mathbb{B}_2$, if $|X_1| > 2$ and $|X_l| \le 2$; and
- Type $\mathbb{C}$, if $|X_1| > 2$ and $|X_l| > 2$.

We say that $\mathcal{Q}$ is of Type $\mathbb{B}$ if $\mathcal{P}$ is either of Type $\mathbb{B}_1$ or of Type $\mathbb{B}_2$. Notice that if $\mathcal{Q}$ is of Type $\mathbb{B}_1$, then $(X_l, \ldots, X_1)$ is of Type $\mathbb{B}_2$, and vice versa. We say that a path decomposition $\mathcal{Q}$ of a chunk graph C is *type-optimal* if $\mathcal{Q}$ has minimum length and

- $\mathcal{Q}$ is of Type $\mathbb{C}$ if no minimum-length path decomposition of Type $\mathbb{A}$ or $\mathbb{B}$ of C exists, or
- $\mathcal{Q}$ is of Type $\mathbb{B}$ if no minimum-length path decomposition of Type $\mathbb{A}$ of C exists, or
- $\mathcal{Q}$ is of Type $\mathbb{A}$ otherwise.

MOST WIEDZY Downloaded from mostwiedzy.pl

Let $\mathcal{P}_1 = (X_1^1, \ldots, X_{l_1}^1)$ and $\mathcal{P}_2 = (X_1^2, \ldots, X_{l_2}^2)$ be two path decompositions of disjoint graphs G^1 and G^2, respectively. We define the *concatenation* of $\mathcal{P}_1$ and $\mathcal{P}_2$, denoted $\mathcal{P}_1 \oplus \mathcal{P}_2$, as follows

$$\mathcal{P}_1 \oplus \mathcal{P}_2 = \begin{cases} (X_1^1, \ldots, X_{l_1-1}^1, X_{l_1}^1 \cup X_1^2, X_2^2, \ldots, X_{l_2}^2), & \text{if } |X_{l_1}| \le 2 \text{ and } |X_1^2| \le 2; \\ (X_1^1, \ldots, X_{l_1}^1, X_1^2, \ldots, X_{l_2}^2), & \text{otherwise.} \end{cases}$$

Clearly, if $\mathrm{width}(\mathcal{P}_1) \le 3$ and $\mathrm{width}(\mathcal{P}_2) \le 3$, then $\mathcal{P}_1 \oplus \mathcal{P}_2$ is a path decomposition of $G^1 \cup G^2$ of width at most 3. Observe that $\mathrm{len}(\mathcal{P}_1) + \mathrm{len}(\mathcal{P}_2) - 1 \le \mathrm{len}(\mathcal{P}_1 \oplus \mathcal{P}_2) \le \mathrm{len}(\mathcal{P}_1) + \mathrm{len}(\mathcal{P}_2)$. Then, let

$$\mathcal{P}_1 \oplus \mathcal{P}_2 \oplus \cdots \oplus \mathcal{P}_c = (\cdots((\mathcal{P}_1 \oplus \mathcal{P}_2) \oplus \mathcal{P}_3)\cdots) \oplus \mathcal{P}_c.$$

We now show that any minimum-length path decomposition of width $k = 3$ of any graph G can be expressed as the concatenation of type-optimal path decompositions of chunk graphs whose union is G.

Lemma 5.7. *Let G be a graph with $\mathrm{pw}(G) \le 3$ and let c be the number of big connected components of G. There exist chunk graphs $C^1, \ldots, C^c$ such that $G = C^1 \cup \cdots \cup C^c$ and for each $i = 1, \ldots, c$ there exists a type-optimal path decomposition $\mathcal{Q}_i$ of C^i such that $\mathcal{P} = \mathcal{Q}_1 \oplus \cdots \oplus \mathcal{Q}_c$ is a minimum-length path decomposition of width at most 3 of G.*

Proof. By Lemma 5.3, there exists a minimum-length path decomposition $\mathcal{P} = (X_1, \ldots, X_l)$ of G, $\mathrm{width}(\mathcal{P}) \le 3$, in which no two big components of G are processed in parallel. Let $G_1, \ldots, G_c$ be all big components of G, and let $L = V(G) \setminus \bigcup_{i=1}^c V(G_i)$. We claim that $\alpha(G_i) \ne \alpha(G_j)$ for all distinct $i, j \in \{1, \ldots, c\}$. Suppose $\alpha(G_i) = \alpha(G_j) = t^*$ for some $i \ne j$. By Corollary 5.5 and by the fact that $|X_{t^*}| \le 4$, we obtain that X_{t^*} contains exactly 2 vertices of each of G_i, G_j. Since G_i and G_j are big components, it follows that X_{t^*+1} also contains at least one vertex from each of G_i, G_j. But this implies that G_i and G_j are processed in parallel, a contradiction. Thus, we may assume without loss of generality that $\alpha(G_i) < \alpha(G_{i+1})$ for each $i = 1, \ldots, c-1$. It follows from Lemma 5.3 that $\beta(G_i) \le \alpha(G_{i+1})$.

Now, let us define the chunk graphs $C^1, \ldots, C^c$ and the corresponding path decompositions $\mathcal{Q}_1, \ldots, \mathcal{Q}_c$. Define $\alpha_1 = 1$, and let $\alpha_i = \alpha(G_i)$ for $i = 2, \ldots, c$. Define $\omega: L \to \{1, \ldots, c\}$ by $\omega(v) = \max\{i: \alpha(v) \ge \alpha_i, 1 \le i \le c\}$. For $i \in \{1, \ldots, c\}$, let $C^i = G\left[V(G_i) \cup \omega^{-1}(i)\right]$ and let

$$\mathcal{Q}_i = \left(X_{\alpha(C^i)} \cap V(C^i), \ldots, X_{\beta(C^i)} \cap V(C^i)\right).$$

By this construction, we have $\alpha(G_i) = \alpha(C^i)$, and, moreover, if $\beta(G_{i-1}) < \beta(C^{i-1})$, then $\beta(C^{i-1}) < \alpha(C^i)$ for $i = 2, \ldots, c$. Thus, if $\beta(C^{i-1}) = \alpha(C^i)$, then $\beta(G_{i-1}) = \alpha(G_i)$. Therefore, by Corollary 5.5, $|X_{\beta(C^{i-1})}| = 2$ and $|X_{\alpha(C^i)}| = 2$ in such case. Finally, if $\beta(C^{i-1}) < \alpha(C^i)$ for any $i \in \{2, \ldots, c\}$, then $|X_{\beta(C^{i-1})} \cap V(C^{i-1})| \ge 3$ or $|X_{\alpha(C^i)} \cap V(C^i)| \ge 3$. Otherwise, $|X_{\beta(C^{i-1})} \cup X_{\alpha(C^i)}| \le 4$ and $\mathcal{P}$ would not be a minimum-length path decomposition, thus contradiction. Therefore, we just proved that $\mathcal{P} = \mathcal{Q}_1 \oplus \cdots \oplus \mathcal{Q}_c$. Possibly by choosing $\mathcal{P}$ differently, we may assume without loss of generality that $\mathcal{P}$ has the maximum number of indices $i \in \{1, \ldots, c\}$ such that $\mathcal{Q}_i$ is type-optimal for C^i.

We conclude the proof by showing that, for each $i \in \{1, \ldots, c\}$, $\mathcal{Q}_i$ is a type-optimal path decomposition of C^i. Suppose for a contradiction that this claim does not hold for some $i \in \{1, \ldots, c\}$. Let $\mathcal{Q}_i'$ be a type-optimal path decomposition of C^i.

If $\mathrm{len}(\mathcal{Q}_i) = \mathrm{len}(\mathcal{Q}_i')$, then by the definition of the types of decompositions,

$$\mathrm{len}(\mathcal{Q}_1 \oplus \cdots \oplus \mathcal{Q}_{i-1} \oplus \mathcal{Q}_i' \oplus \mathcal{Q}_{i+1} \oplus \cdots \oplus \mathcal{Q}_c) \le \mathrm{len}(\mathcal{P}),$$

which contradicts our choice of $\mathcal{P}$. Hence, $\mathrm{len}(\mathcal{Q}_i) > \mathrm{len}(\mathcal{Q}_i')$. If $\mathcal{Q}_i$ is not of Type $\mathbb{A}$ or $\mathcal{Q}_i'$ is not of Type $\mathbb{C}$, then

$$\mathrm{len}(\mathcal{Q}_1 \oplus \cdots \oplus \mathcal{Q}_{i-1} \oplus \mathcal{Q}_i' \oplus \mathcal{Q}_{i+1} \oplus \cdots \oplus \mathcal{Q}_c) \le \mathrm{len}(\mathcal{P}),$$

and if $\mathcal{Q}_i$ is of Type $\mathbb{A}$ and $\mathcal{Q}_i'$ is of Type $\mathbb{C}$, then

$$\mathrm{len}\left(\mathcal{Q}_1 \oplus \cdots \oplus \mathcal{Q}_{i-1} \oplus \mathcal{Q}_{i+1} \oplus \cdots \oplus \mathcal{Q}_c \oplus \mathcal{Q}_i'\right) \le \mathrm{len}(\mathcal{P}),$$

which again contradicts our choice of $\mathcal{P}$. This completes the proof of this case. □

Lemma 5.7 implies that, we may construct a minimum-length path decomposition of width at most k of graph G by constructing a type-optimal path decomposition of each chunk graph of G separately, and then concatenating the resulting type-optimal path decompositions. The only two caveats here are: the optimal number of K_1- and K_2-components in each chunk graph and the optimal ordering of the type-optimal path decompositions. We deal with the latter in the next subsection.

MOST WIEDZY Downloaded from mostwiedzy.pl

5.3. Optimal ordering of path decompositions of chunk graphs

Let us now assume that path decompositions $\mathcal{Q}_1, \ldots, \mathcal{Q}_c$ of chunk graphs $C^1, \ldots, C^c$, respectively, are given and $G = C^1 \cup \ldots \cup C^c$. In this subsection, our goal is to find an optimal order, i.e., the one that minimizes the length of the resulting path decomposition of G, in which to concatenate $\mathcal{Q}_i$'s. Note that, by Observation 5.6, the $\mathcal{Q}_i$'s can be concatenated in any order when $k < 3$. Thus, we assume that $k = 3$ in the remainder of this section. To obtain the order we determine the permutation of $\mathcal{Q}_i$'s and, if a particular $\mathcal{Q}_i$ is of type $\mathbb{B}$, then determine whether $\mathcal{Q}_i$ should be of Type $\mathbb{B}_1$ or of Type $\mathbb{B}_2$ in the concatenation (hence, such a $\mathcal{Q}_i$ may be reversed before producing the concatenation).

Let a, b_1, b_2 denote the number of $\mathcal{Q}_i$'s of Type $\mathbb{A}$, $\mathbb{B}_1$, $\mathbb{B}_2$, respectively, $i = 1, \ldots, c$, and let $b = b_1 + b_2$. We say that the sequence $\mathcal{Q}_1, \ldots, \mathcal{Q}_c$ is *in normal form* if $b_1 = \lfloor b/2 \rfloor$, $b_2 = \lceil b/2 \rceil$, and they are ordered as follows:

(i) If $b = 0$, then $\mathcal{Q}_1, \ldots, \mathcal{Q}_a$ are of Type $\mathbb{A}$ and $\mathcal{Q}_{a+1}, \ldots, \mathcal{Q}_c$ are of Type $\mathbb{C}$. (Informally, using the Kleene star notation, the pattern is non-empty and belongs to $\mathbb{A}^*\mathbb{C}^*$.)
(ii) If $b = 1$, then $\mathcal{Q}_1, \ldots, \mathcal{Q}_{c-a-1}$ are of type $\mathbb{C}$; $\mathcal{Q}_{c-a}$ is of Type $\mathbb{B}_2$; and $\mathcal{Q}_{c-a+1}, \ldots, \mathcal{Q}_c$ are of type $\mathbb{A}$. (Informally: the pattern belongs to $\mathbb{C}^*\mathbb{B}_2\mathbb{A}^*$.)
(iii) If $b > 1$, then $\mathcal{Q}_1$ is of Type $\mathbb{B}_2$; $\mathcal{Q}_2, \ldots, \mathcal{Q}_{a+1}$ are Type $\mathbb{A}$; $\mathcal{Q}_{a+2}$ is of Type $\mathbb{B}_1$; $\mathcal{Q}_{a+3}, \ldots, \mathcal{Q}_{a+c+2}$ are of Type $\mathbb{C}$; and $\mathcal{Q}_{a+c+3}, \ldots, \mathcal{Q}_c$ alternate Type $\mathbb{B}_2$ and $\mathbb{B}_1$, starting with Type $\mathbb{B}_2$. (Informally: the pattern belongs to $\mathbb{B}_2\mathbb{A}^*\mathbb{B}_1\mathbb{C}^*(\mathbb{B}_2\mathbb{B}_1)^*$ for an even b, and to $\mathbb{B}_2\mathbb{A}^*\mathbb{B}_1\mathbb{C}^*\mathbb{B}_2(\mathbb{B}_1\mathbb{B}_2)^*$) for an odd b.)

The following lemma implies that the normal form is an optimal way of ordering and reversing the path decompositions $\mathcal{Q}_1, \ldots, \mathcal{Q}_c$ that results in a minimum-length path decomposition of G. For convenience, define

$$\mu(a, b) = \begin{cases} \max(0, a-1) & \text{if } b = 0, \\ a + \lfloor b/2 \rfloor & \text{if } b > 0. \end{cases}$$

Lemma 5.8. *Let G be a graph with $\mathrm{pw}(G) = 3$. Let $C^1, \ldots, C^c$ be any chunk graphs such that $G = C^1 \cup \ldots \cup C^c$. Let $\mathcal{Q}_i$ be a path decomposition of width at most 3 of C^i, $i = 1, \ldots, c$. Let a and b be the numbers of path decompositions among $\mathcal{Q}_1, \ldots, \mathcal{Q}_c$ of Type $\mathbb{A}$ and of Type $\mathbb{B}$, respectively. Then,*

$$\min_{\pi, \mathbf{f} \in \{0,1\}^c} \left\{ \mathrm{len}\left(\mathcal{Q}^{f_1}_{\pi(1)} \oplus \ldots \oplus \mathcal{Q}^{f_c}_{\pi(c)} \right) \right\} = \sum_{i=1}^{c} \mathrm{len}(\mathcal{Q}_i) - \mu(a, b), \tag{19}$$

where the minimization is over all permutations $\pi : \{1, \ldots, c\} \to \{1, \ldots, c\}$, $\mathcal{Q}^0_i = \mathcal{Q}_i$, $\mathcal{Q}^1_i$ is the reverse of $\mathcal{Q}_i$ and $\mathbf{f} = (f_1, \ldots, f_c)$.

Proof. Consider an arbitrary permutation $\pi : \{1, \ldots, c\} \to \{1, \ldots, c\}$ and a vector $\mathbf{f} = (f_1, \ldots, f_c) \in \{0, 1\}^c$. Let $\nu = \nu(\pi, \mathbf{f})$ denote the number of pairs $(i, i+1)$, $i \in \{1, \ldots, c-1\}$, such that $\mathrm{len}\left(\mathcal{Q}^{f_i}_{\pi(i)} \oplus \mathcal{Q}^{f_{i+1}}_{\pi(i+1)} \right) = \mathrm{len}\left(\mathcal{Q}^{f_i}_{\pi(i)} \right) + \mathrm{len}\left(\mathcal{Q}^{f_{i+1}}_{\pi(i+1)} \right) - 1$. We refer to these pairs as *matchups*. Clearly, since each chunk graph C^i has a big connected component,

$$\mathrm{len}\left(\mathcal{Q}^{f_1}_{\pi(1)} \oplus \cdots \oplus \mathcal{Q}^{f_c}_{\pi(c)} \right) = \sum_{i=1}^{c} \mathrm{len}\left(\mathcal{Q}_i\right) - \nu(\pi, \mathbf{f}). \tag{20}$$

By Corollary 5.5, and by definition of $\oplus$, each matchup $(i, i+1)$ requires that $\mathcal{Q}^{f_i}_{\pi(i)}$ is of Type $\mathbb{A}$ or of Type $\mathbb{B}_2$, and $\mathcal{Q}^{f_{i+1}}_{\pi(i+1)}$ is of Type $\mathbb{A}$ or of Type $\mathbb{B}_1$. We therefore have that if $b > 0$, then $\nu(\pi, \mathbf{f}) \le a + \min\{b_1, b_2\} \le \mu(a, b)$. Moreover, if $b = 0$, then $\nu(\pi, \mathbf{f}) \le \max\{0, a-1\} = \mu(a, b)$. Since π and $\mathbf{f}$ were chosen arbitrarily, it follows that $\nu(\pi, \mathbf{f}) \le \mu(a, b)$ for every permutation π and $\mathbf{f} \in \{0, 1\}^c$, which, by (20), proves the "$\ge$" direction of (19).

It remains to prove the "$\le$" direction of (19). Clearly, there are π and $\mathbf{f}$ such that the sequence $\mathcal{Q}^{f_1}_{\pi(1)} \oplus \cdots \oplus \mathcal{Q}^{f_c}_{\pi(c)}$ is in normal form. Now, it is straightforward to check that the number of matchups for $\mathcal{Q}_1, \ldots, \mathcal{Q}_c$ is exactly $\mu(a, b)$, thus proving the lemma. □

Note that the proof of Lemma 5.8 together with Lemma 5.7 immediately give an algorithm for graphs G with no K_1- and K_2-components: find a type-optimal path decomposition for each connected component G_i separately, order the resulting path decompositions so that the sequence is in normal form, and concatenate them.

Therefore, it remains do show how many K_1- and K_2-components need to be added to each G_i to make up a chunk graph C^i. Section 5.4 deals with this question.

5.4. A dynamic programming algorithm for general graphs

We assume that G is a graph with big connected components $G_1, \ldots, G_c$, $c \ge 1$, K_1-components $K^1_1, \ldots, K^{q_1}_1$, and K_2-components $K^1_2, \ldots, K^{q_2}_2$.

MOST WIEDZY Downloaded from mostwiedzy.pl

Clearly we could find all possible chunk graphs for G by enumerating all distributions of K_1- and K_2-components of G among the big components of G. However, the running time of such a procedure is not, in general, polynomial in the size of G. We overcome this problem by designing a dynamic programming procedure that eliminates unnecessary distributions leaving only those that can possibly lead to minimal length path decomposition of G. The procedure calls Algorithm 1 to determine type-optimal path decomposition for each distribution (i.e., for given chunk graphs) it considers worth trying. We now give details of the procedure.

For any vector $\mathbf{s} = (s_1, s_2) \in \mathbb{Z}_+^2$ and for any integer $i \in \{1, \ldots, c\}$, construct $H_i(\mathbf{s})$ by taking G_i, s_1 isolated vertices and s_2 isolated edges, and let $\mathcal{Q}_i(\mathbf{s})$ be a type-optimal path decomposition of $H_i(\mathbf{s})$. Let $\tau(\mathcal{Q}_i(\mathbf{s}))$ be the type of $\mathcal{Q}_i(\mathbf{s})$. For $m \in \{1, \ldots, c\}$, $r_1 \in \{0, \ldots, q_1\}$ and $r_2 \in \{0, \ldots, q_2\}$, define

$$\mathcal{D}_m(r_1, r_2) = \left\{ \mathbf{d}\colon \{1, \ldots, m\} \to \mathbb{Z}_+^2 \;\middle|\; \sum_{i=1}^{m} \mathbf{d}_1(i) = r_1 \text{ and } \sum_{i=1}^{m} \mathbf{d}_2(i) = r_2 \right\},$$

where $\mathbf{d}_1(i)$ and $\mathbf{d}_2(i)$ are the first and second entry of the vector $\mathbf{d}(i)$, respectively. The set $\mathcal{D}_m(r_1, r_2)$ represents all assignments of r_1 K_1-components and r_2 K_2-components to the first m big components $G_1, \ldots, G_m$. Thus, $\mathbf{d}_1(i)$ and $\mathbf{d}_2(i)$ are the quantities of K_1- and K_2-components, respectively, that are assigned to the big component G_i, $i = 1, \ldots, m$. Thus, once $\mathbf{d}$ is fixed, the chunk graphs are fixed. And with those fixed, Lemma 5.7 and Lemma 5.8 show that by concatenating their type-optimal path decompositions in the normal form we obtain a minimum-length path decomposition of G.

Let $\mathbf{d} \in \mathcal{D}_m(r_1, r_2)$. Let $\#\mathbb{A}(\mathbf{d})$ and $\#\mathbb{B}(\mathbf{d})$ be the numbers of path decompositions of Type $\mathbb{A}$ and of Type $\mathbb{B}$, respectively, among the type-optimal path decompositions $\mathcal{Q}_1(\mathbf{d}(1)), \ldots, \mathcal{Q}_m(\mathbf{d}(m))$. Define

$$\operatorname{len}(\mathbf{d}) = \sum_{i=1}^{m} \operatorname{len}(\mathcal{Q}_i(\mathbf{d}(i))) - \mu(\mathbf{d}). \tag{21}$$

Moreover, let

$$\omega(\mathbf{d}) = \begin{cases} 1, & \text{if } \#\mathbb{A}(\mathbf{d}) = \#\mathbb{B}(\mathbf{d}) = 0; \\ 2, & \text{if } \#\mathbb{A}(\mathbf{d}) > 0 \text{ and } \#\mathbb{B}(\mathbf{d}) = 0; \\ 3, & \text{if } \#\mathbb{B}(\mathbf{d}) > 0 \text{ and } \#\mathbb{B}(\mathbf{d}) \text{ is odd}; \\ 4, & \text{if } \#\mathbb{B}(\mathbf{d}) > 0 \text{ and } \#\mathbb{B}(\mathbf{d}) \text{ is even}. \end{cases}$$

We will call $\omega(\mathbf{d})$ the *configuration* of $\mathbf{d}$. It divides all possible normal-form path decompositions for $\mathcal{Q}_1(\mathbf{d}(1))$, …, $\mathcal{Q}_m(\mathbf{d}(m))$ into four different categories, depending on the number of path decompositions of Type $\mathbb{A}$ and $\mathbb{B}$ among $\mathcal{Q}_1(\mathbf{d}(1)), \ldots, \mathcal{Q}_m(\mathbf{d}(m))$. Our dynamic programming algorithm works as follows. It builds a 4-dimensional array ϕ_m^*, $m \in \{1, \ldots, c\}$, with the entries $\phi_m^*(r_1, r_2, t)$ where $r_1 \in \{0, \ldots, q_1\}$, $r_2 \in \{0, \ldots, q_2\}$ and $t \in \{1, 2, 3, 4\}$, which satisfy

$$\phi_m^*(r_1, r_2, t) = \min\left\{ \operatorname{len}(\mathbf{d}) \;\middle|\; \mathbf{d} \in \mathcal{D}_m(r_1, r_2) \text{ and } \omega(\mathbf{d}) = t \right\}. \tag{22}$$

We set $\phi_m^*(r_1, r_2, t) = \infty$ for $\left\{ \mathbf{d} \;\middle|\; \mathbf{d} \in \mathcal{D}_m(r_1, r_2) \text{ and } \omega(\mathbf{d}) = t \right\} = \emptyset$. It follows from Lemma 5.7 and Lemma 5.8 that for each $t \in \{1, 2, 3, 4\}$, $\phi_c^*(q_1, q_2, t) < \infty$ is the minimum integer such that there exists a path decomposition $\mathcal{P}$ of width 3 of G with $\operatorname{len}(\mathcal{P}) = \phi_c^*(q_1, q_2, t)$ and with the corresponding vector $\mathbf{d}$ in configuration t, $\omega(\mathbf{d}) = t$. Thus, $l = \min_{t \in \{1,2,3,4\}}\{\phi_c^*(q_1, q_2, t)\}$ is the minimum integer such that there exists a path decomposition of width 3 and length l of G. Clearly, $\phi_c^*(q_1, q_2, t) < \infty$ for some $t \in \{1, 2, 3, 4\}$. It remains to show that the values of $\phi_m^*(r_1, r_2, t)$ can be recursively calculated in polynomial time, which is what the next two lemmas show. We start with the following crucial observation:

Lemma 5.9. *Let $m \in \{2, \ldots, c\}$. Let $\bar{\mathbf{d}} \in \mathcal{D}_m(r_1, r_2)$ and let $\mathbf{d} \in \mathcal{D}_{m-1}\left(\sum_{i=1}^{m-1} \bar{\mathbf{d}}_1(i), \sum_{i=1}^{m-1} \bar{\mathbf{d}}_2(i)\right)$ be such that $\mathbf{d}(i) = \bar{\mathbf{d}}(i)$ for each $i = 1, \ldots, m-1$. If $\tau(\mathcal{Q}_m(\bar{\mathbf{d}}(m))) = \mathbb{C}$, then $\mu(\#\mathbb{A}(\bar{\mathbf{d}}), \#\mathbb{B}(\bar{\mathbf{d}})) = \mu(\#\mathbb{A}(\mathbf{d}), \#\mathbb{B}(\mathbf{d}))$ and $\delta(\mathbf{d}, \bar{\mathbf{d}}) = 0$. Otherwise,*

$$\delta(\mathbf{d}, \bar{\mathbf{d}}) = \mu(\#\mathbb{A}(\bar{\mathbf{d}}), \#\mathbb{B}(\bar{\mathbf{d}})) - \mu(\#\mathbb{A}(\mathbf{d}), \#\mathbb{B}(\mathbf{d})) = \begin{cases} 0, & \textit{if } (\omega(\mathbf{d}), \omega(\bar{\mathbf{d}})) \in \{(1,2), (1,3), (4,3), (2,3)\}; \\ 1, & \textit{otherwise.} \end{cases}$$

In particular, $\delta(\mathbf{d}, \bar{\mathbf{d}})$ only depends on $\omega(\mathbf{d})$, $\omega(\bar{\mathbf{d}})$, and $\tau(\mathcal{Q}_m(\bar{\mathbf{d}}(m)))$.

Proof. If $\mathcal{Q}_m(\bar{\mathbf{d}}(m))$ is of Type $\mathbb{C}$, the result follow from the fact that $\mathcal{Q}_i(\mathbf{d}(i))$ and $\mathcal{Q}_i(\bar{\mathbf{d}}(i))$ are of the same type for each $i = 1, \ldots, m-1$. So we may assume that $\mathcal{Q}_m(\bar{\mathbf{d}}(m))$ is not of Type $\mathbb{C}$. The possible transitions from $\omega(\mathbf{d})$ to $\omega(\bar{\mathbf{d}})$ when $\mathcal{Q}_m(\bar{\mathbf{d}}(m))$ is of Type $\mathbb{A}$ or $\mathbb{B}$ are graphically shown in Fig. 8. The nodes represent configurations $\omega \in \{1, 2, 3, 4\}$ and the edges represent pairs $(\omega(\mathbf{d}), \omega(\bar{\mathbf{d}}))$. Each edge has two labels, one that indicates the value of $\delta(\mathbf{d}, \bar{\mathbf{d}})$ and the other that indicates the type of $\mathcal{Q}_m(\bar{\mathbf{d}}(m))$. Notice that $\#\mathbb{A}(\bar{\mathbf{d}}) - \#\mathbb{A}(\mathbf{d}) \in \{0, 1\}$ and $\#\mathbb{B}(\bar{\mathbf{d}}) - \#\mathbb{B}(\mathbf{d}) \in \{0, 1\}$. It is now straightforward to check that the edges depicted in the figure are the only possible pairs $(\omega(\mathbf{d}), \omega(\bar{\mathbf{d}}))$ and that the value of $\delta(\mathbf{d}, \bar{\mathbf{d}})$ is exactly as given in the figure. □

Let $h\colon \{1, 2, 3, 4\} \times \{\mathbb{A}, \mathbb{B}, \mathbb{C}\} \to \{1, 2, 3, 4\}$ be the transition function and $g\colon \{1, 2, 3, 4\} \times \{\mathbb{A}, \mathbb{B}, \mathbb{C}\} \to \{0, 1\}$ be transition weights δ from Fig. 8. We assume $h(t, \mathbb{C}) = t$ and $g(t, \mathbb{C}) = 0$ for any configuration t.

MOST WIEDZY Downloaded from mostwiedzy.pl

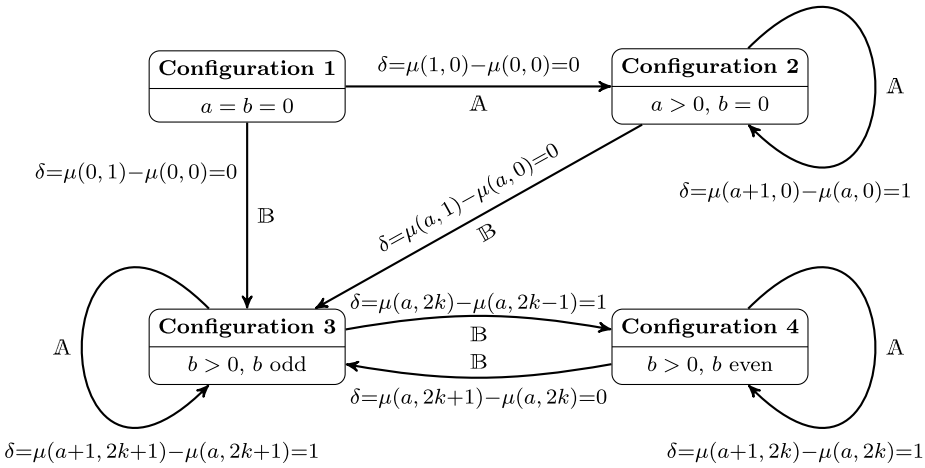

Fig. 8. Transitions between different configurations.

Lemma 5.10. *Let $m \in \{2,\dots,c\}$. If there exists a polynomial-time algorithm that, for any chunk graph C, and for any $e_1, e_2 \in \{2,\dots,k+1\}$, either constructs a minimum-length (e_1,e_2)-path decomposition of width k of C, or concludes that no such path decomposition exists, and if the array ϕ^*_{m-1} is given, then $\phi^*_m(r_1,r_2,t)$ can be computed in polynomial time for each $r_1 \in \{0,\dots,q_1\}$, $r_2 \in \{0,\dots,q_2\}$, $t \in \{1,2,3,4\}$.*

Proof. Fix r_1, r_2, t, and m. In order to prove the lemma, we will show that the following recursive dynamic programming relation holds:

$$\phi^*_m(r_1,r_2,t) = \min_{\substack{0\le i\le r_1\\ 0\le j\le r_2\\ 1\le t'\le 4}} \left[\phi^*_{m-1}(r_1-i,r_2-j,t') + \mathrm{len}(\mathcal{Q}_m(i,j)) - g\big(t',\tau(\mathcal{Q}_m(i,j))\big) \;\middle|\; h(t',\tau(\mathcal{Q}_m(i,j)) = t\right], \tag{23}$$

where the right hand side of (23) equals ∞ if $\phi^*_{m-1}(r_1-i,r_2-j,t') = \infty$ or $h(t',\tau(\mathcal{Q}_m(i,j)) \neq t$ for each $i \in \{0,\dots,r_1\}$, $j \in \{0,\dots,r_2\}$ and $t' \in \{1,2,3,4\}$.

This suffices because the right hand side of (23) can be calculated in polynomial time. Indeed, the array ϕ^*_{m-1} is given by assumption; $\mathcal{Q}_m(i,j)$, its length, $\mathrm{len}(\mathcal{Q}_m(i,j))$, and its type, $\tau(\mathcal{Q}_m(i,j))$, can be calculated in polynomial time by assumption; thus, since the initial configuration t' is given for each entry ϕ^*_{m-1}, both $h(t',\tau(\mathcal{Q}_m(i,j)))$ and $g(t',\tau(\mathcal{Q}_m(i,j)))$ can be readily calculated.

We now prove (23). First, we observe that if for each $i \in \{0,\dots,r_1\}$, $j \in \{0,\dots,r_2\}$ and $t' \in \{1,2,3,4\}$, $\phi^*_{m-1}(r_1-i,r_2-j,t') = \infty$ or $h(t',\tau(\mathcal{Q}_m(i,j)) \neq t$, then the right hand side of (23) equals ∞. On the other hand, the definition of ϕ^*_{m-1} then implies that the set $\{\mathbf{d} \mid \mathbf{d} \in \mathcal{D}_m(r_1,r_2) \text{ and } \omega(\mathbf{d}) = t\}$ is empty and thus $\phi^*_m(r_1,r_2,t) = \infty$ and (23) holds as required. Therefore, we assume now that there are $i \in \{0,\dots,r_1\}$, $j \in \{0,\dots,r_2\}$ and $t' \in \{1,2,3,4\}$ such that $\phi^*_{m-1}(r_1-i,r_2-j,t') < \infty$ and $h(t',\tau(\mathcal{Q}_m(i,j)) = t$. We prove (23) by proving that the inequality holds in both directions.

"$\le$": Let $\mathbf{d}^{i,j,t'} \in \mathcal{D}_{m-1}(r_1-i,r_2-j)$ be a vector such that $\mathrm{len}(\mathbf{d}^{i,j,t'}) = \phi^*_{m-1}(r_1-i,r_2-j,t')$ and $\omega(\mathbf{d}^{i,j,t'}) = t'$. Let $\bar{\mathbf{d}}^{i,j,t'} \in \mathcal{D}_m(r_1,r_2)$ be the extension of $\mathbf{d}^{i,j,t'}$ to $\{1,\dots,m\}$ that satisfies $\bar{\mathbf{d}}^{i,j,t'}_1(m) = i$, $\bar{\mathbf{d}}^{i,j,t'}_2(m) = j$, and $\omega(\bar{\mathbf{d}}^{i,j,t'}) = t$. By Lemma 5.8,

$$\begin{aligned}
\phi^*_m(r_1,r_2,t) \le \mathrm{len}(\bar{\mathbf{d}}^{i,j,t'}) &= \sum_{l=1}^{m} \mathrm{len}\left(\mathcal{Q}_l(\bar{\mathbf{d}}^{i,j,t'}(l))\right) - \mu(\#\mathbb{A}(\bar{\mathbf{d}}^{i,j,t'}), \#\mathbb{B}(\bar{\mathbf{d}}^{i,j,t'}))\\
&= \left[\sum_{l=1}^{m-1} \mathrm{len}\left(\mathcal{Q}_l(\mathbf{d}^{i,j,t'}(l))\right) - \mu(\#\mathbb{A}(\mathbf{d}^{i,j,t'}), \#\mathbb{B}(\mathbf{d}^{i,j,t'}))\right] + \mathrm{len}(\mathcal{Q}_m(i,j)) - \delta(\mathbf{d}^{i,j,t'}, \bar{\mathbf{d}}^{i,j,t'})\\
&= \phi^*_{m-1}(r_1-i,r_2-j,t') + \mathrm{len}(\mathcal{Q}_m(i,j)) - \delta(\mathbf{d}^{i,j,t'}, \bar{\mathbf{d}}^{i,j,t'})\\
&= \phi^*_{m-1}(r_1-i,r_2-j,t') + \mathrm{len}(\mathcal{Q}_m(i,j)) - g(t',\tau(\mathcal{Q}_m(i,j))),
\end{aligned}$$

where $\delta(\mathbf{d}^{i,j,t'}\bar{\mathbf{d}}^{i,j,t'}) = g(t',\tau(\mathcal{Q}_m(i,j)))$ in the last equality follows from Lemma 5.9 and $\omega(\mathbf{d}^{i,j,t'}) = t'$. Finally, $\omega(\bar{\mathbf{d}}^{i,j,t'}) = h(t',\tau(\mathcal{Q}_m(i,j)) = t$ as required.

"$\ge$": Let $\bar{\mathbf{d}}^* \in \mathcal{D}_m(r_1,r_2)$ be such that $\mathrm{len}(\bar{\mathbf{d}}^*) = \phi^*_m(r_1,r_2,t)$. Thus, $\omega(\bar{\mathbf{d}}^*) = t$. Let $i = \bar{\mathbf{d}}^*_1(m)$, $j = \bar{\mathbf{d}}^*_2(m)$, let $\mathbf{d}^* \in \mathcal{D}_{m-1}(r_1-i,r_2-j)$ be the restriction of $\bar{\mathbf{d}}^*$ to $\{1,\dots,m-1\}$, and let $t' = \omega(\mathbf{d}^*)$. By Lemma 5.8,

MOST WIEDZY Downloaded from mostwiedzy.pl

$$\begin{aligned}
\phi^*_m(r_1, r_2, t) = \mathrm{len}(\bar{\mathbf{d}}^*) &= \sum_{l=1}^{m} \mathrm{len}\left(\mathcal{Q}_l(\bar{\mathbf{d}}^*(l))\right) - \mu(\#\mathbb{A}(\bar{\mathbf{d}}^*), \#\mathbb{B}(\bar{\mathbf{d}}^*)) \\
&= \left[\sum_{l=1}^{m-1} \mathrm{len}\left(\mathcal{Q}_l(\mathbf{d}^*(l))\right) - \mu(\#\mathbb{A}(\mathbf{d}^*), \#\mathbb{A}(\mathbf{d}^*))\right] + \mathrm{len}(\mathcal{Q}_m(i, j)) - \delta(\mathbf{d}^*, \bar{\mathbf{d}}^*) \\
&\geq \phi^*_{m-1}(r_1 - i, r_2 - j, t') + \mathrm{len}(\mathcal{Q}_m(i, j)) - \delta(\mathbf{d}^*, \bar{\mathbf{d}}^*) \\
&= \phi^*_{m-1}(r_1 - i, r_2 - j, t') + \mathrm{len}(\mathcal{Q}_m(i, j)) - g(t', \tau(\mathcal{Q}_m(i, j)))
\end{aligned}$$

where $\delta(\mathbf{d}^*, \bar{\mathbf{d}}^*) = g(t', \tau(\mathcal{Q}_m(i, j)))$ in the last equality follows from Lemma 5.9 and $\omega(\mathbf{d}^{i,j,t'}) = t'$. Finally, $\omega(\bar{\mathbf{d}}^*) = h(t', \tau(\mathcal{Q}_m(i, j)) = t$ as required. □

We are now ready to prove Theorem 5.1.

Proof of Theorem 5.1. Let $k = 3$. The array ϕ^*_1 can be directly computed using Algorithm 1 for chunk graphs. Thus, induction on m and Lemma 5.10 (given ϕ^*_{m-1}, ϕ^*_m can be computed with the help of Algorithm 1) imply that Theorem 5.1 follows for $k = 3$.

Let $k < 3$. By Observation 5.6, finding a minimum-length path decomposition of G reduces to determining of the assignment of small components to the big components, i.e., it reduces to partitioning G into chunk graphs and then to concatenating (in any order) the minimum-length path decomposition of the chunk graphs. Hence, in the case of $k \in \{1, 2\}$ we use the dynamic programming algorithm in which all (e_1, e_2)-path decompositions of chunk graphs are of the same type. Thus, each minimum-length path decomposition of a chunk graph is type-optimal. The decomposition can be computed by using Algorithm 1. □

We conclude this section by stating the main results of this paper. By the fact that the problem MLPD(0)-CONSTR is trivial, and by Theorem 4.1 and Theorem 5.1 we obtain:

Theorem 5.11. *Given a graph G and an integer $k \leq 3$, there exists a polynomial-time algorithm that computes a minimum-length path decomposition of width at most k of G, or concludes that no such path decomposition exists.* □

6. Conclusions and open problems

In this paper, we have considered a bicriterion generalization of the pathwidth problem, where, for given integers k, l and a graph G, we ask the question whether there exists a path decomposition $\mathcal{P}$ of G such that the width of $\mathcal{P}$ is at most k and the length of $\mathcal{P}$, is at most l. We have shown that a minimum-length path decomposition can be found in polynomial time provided that $k \leq 3$, and that the minimum-length path decomposition problem becomes NP-hard for $k \geq 4$. Also, we have shown that the minimum-width path decomposition problem becomes NP-hard for $l \geq 2$. Though these results provide a complete complexity classification of the bicriterion problem for general graphs, we point out some open problems and interesting directions for further research:

- The most immediate open question is the complexity status of finding a minimum-length path decomposition of width $k = 4$ for connected graphs. Also, given our focus on the structural properties and complexity status of the special cases with fixed pathwidth parameter k in this paper, our algorithms, although polynomial in the size of G for $k \leq 3$, are not very efficient. Hence, an interesting and challenging open question remains: do there exist low-degree polynomial-time algorithms for MLPD(k)-CONSTR for connected and disconnected graphs and $k \leq 3$?
- Another research direction is the study of approximate solutions to MLPD(k)-CONSTR and the trade-offs between the width k and the length l. More precisely, whenever an efficient optimization algorithm for a case of MLPD(k)-CONSTR is unlikely to exist, it is justifiable to design approximation algorithms that find path decompositions whose width and length are within some, preferably provable, bounds from the optima.
- Since the MLPD(k)-CONSTR problem has appeared in a different context as a combinatorial problem motivated by an industrial application [2], one may search for efficient algorithms for special classes of graphs that are particularly relevant for this and other real-life applications.

Acknowledgments

This research has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) Grant OPG0105675, and has been partially supported by Narodowe Centrum Nauki under contract DEC-2011/02/A/ST6/00201. Dariusz Dereniowski has been partially supported by a scholarship for outstanding young researchers founded by the Polish Ministry of Science and Higher Education.

MOST WIEDZY Downloaded from mostwiedzy.pl

References

[1] S. Arnborg, D.G. Corneil, A. Proskurowski, Complexity of finding embeddings in a k-tree, SIAM J. Algebr. Discrete Methods 8 (1987) 277–284.
[2] M. Aschinger, C. Drescher, G. Gottlob, P. Jeavons, E. Thorstensen, Structural decomposition methods and what they are good for, in: Thomas Schwentick, Christoph Dürr (Eds.), 28th International Symposium on Theoretical Aspects of Computer Science, STACS 2011, in: Leibniz International Proceedings in Informatics (LIPIcs), vol. 9, 2011, pp. 12–28.
[3] D. Bienstock, Graph searching, path-width, tree-width and related problems (a survey), DIMACS Ser. Discret. Math. Theor. Comput. Sci. 5 (1991) 33–49.
[4] D. Bienstock, P. Seymour, Monotonicity in graph searching, J. Algorithms 12 (2) (1991) 239–245.
[5] H.L. Bodlaender, NC-algorithms for graphs with small treewidth, in: 14th International Workshop on Graph-Theoretic Concepts in Computer Science, WG 1998, in: Lecture Notes in Computer Science, vol. 344, 1989, pp. 1–10.
[6] H.L. Bodlaender, A linear-time algorithm for finding tree-decompositions of small treewidth, SIAM J. Comput. 25 (6) (1996) 1305–1317.
[7] H.L. Bodlaender, A partial k-arboretum of graphs with bounded treewidth, Theor. Comput. Sci. 209 (1998) 1–45.
[8] H.L. Bodlaender, Treewidth: structure and algorithms, in: 14th International Colloquium on Structural Information and Communication Complexity, SIROCCO 2007, in: Lecture Notes in Computer Science, vol. 4474, 2007, pp. 11–25.
[9] H.L. Bodlaender, Fixed-parameter tractability of treewidth and pathwidth, in: The Multivariate Algorithmic Revolution and Beyond, in: Lecture Notes in Computer Science, vol. 7370, 2012, pp. 196–227.
[10] H.L. Bodlaender, T. Hagerup, Tree decompositions of small diameter, in: The 23rd International Symposium on Mathematical Foundations of Computer Science, MFCS, in: Lecture Notes in Computer Science, vol. 1450, 1998, pp. 702–712.
[11] H.L. Bodlaender, T. Kloks, Efficient and constructive algorithms for the pathwidth and treewidth of graphs, J. Algorithms 21 (2) (1996) 358–402.
[12] H.L. Bodlaender, R.H. Möhring, The pathwidth and treewidth of cographs, SIAM J. Discrete Math. 6 (2) (1993) 181–188.
[13] F.J. Brandenburg, S. Herrmann, Graph searching and search time, in: 32nd Conference on Current Trends in Theory and Practice of Computer Science, SOFSEM 2006, in: Lecture Notes in Computer Science, vol. 3831, 2006, pp. 197–206.
[14] N.D. Dendris, L.M. Kirousis, D.M. Thilikos, Fugitive-search games on graphs and related parameters, Theor. Comput. Sci. 172 (1–2) (1997) 233–254.
[15] D. Dereniowski, From pathwidth to connected pathwidth, SIAM J. Discrete Math. 26 (4) (2012) 1709–1732.
[16] J.A. Ellis, I.H. Sudborough, J.S. Turner, The vertex separation and search number of a graph, Inf. Comput. 113 (1994) 50–79.
[17] F.V. Fomin, Helicopter search problems, bandwidth and pathwidth, Discrete Appl. Math. 85 (1) (1998) 59–70.
[18] F.V. Fomin, P.A. Golovach, Graph searching and interval completion, SIAM J. Discrete Math. 13 (4) (2000) 454–464.
[19] F.V. Fomin, P. Heggernes, J.A. Telle, Graph searching, elimination trees, and a generalization of bandwidth, Algorithmica 41 (2) (2004) 73–87.
[20] F.V. Fomin, D.M. Thilikos, An annotated bibliography on guaranteed graph searching, Theor. Comput. Sci. 399 (3) (2008) 236–245.
[21] M.R. Garey, D.S. Johnson, Computers and Intractability: a Guide to the Theory of NP-Completeness, W. H. Freeman & Co., New York, NY, USA, 1979.
[22] A. Gupta, N. Nishimura, The complexity of subgraph isomorphism for classes of partial k-trees, Theor. Comput. Sci. 164 (1&2) (1996) 287–298.
[23] A. Gupta, N. Nishimura, A. Proskurowski, P. Ragde, Embeddings of k-connected graphs of pathwidth k, Discrete Appl. Math. 145 (2) (2005) 242–265.
[24] A. Gupta, S. Sudarshan, S. Viswanathan, Query scheduling in multi query optimization, in: International Database Engineering and Application Symposium (IDEAS), IEEE, 2001, pp. 11–19.
[25] J. Gustedt, On the pathwidth of chordal graphs, Discrete Appl. Math. 45 (3) (1993) 233–248.
[26] N.G. Kinnersley, The vertex separation number of a graph equals its path-width, Inf. Process. Lett. 42 (6) (1992) 345–350.
[27] L.M. Kirousis, C.H. Papadimitriou, Interval graphs and searching, Discrete Appl. Math. 55 (1985) 181–184.
[28] L.M. Kirousis, C.H. Papadimitriou, Searching and pebbling, Theor. Comput. Sci. 47 (2) (1986) 205–218.
[29] T. Kloks, Treewidth, Computations and Approximations, Lecture Notes in Computer Science, vol. 842, Springer, 1994.
[30] A. Kornai, Z. Tuza, Narrowness, pathwidth, and their application in natural language processing, Discrete Appl. Math. 36 (1) (1992) 87–92.
[31] B. Li, F.Z. Moataz, N. Nisse, K. Suchan, Size-constrained tree decompositions, Research report, INRIA Sophia-Antipolis, October 2014.
[32] J. Manuch, L. Stacho, C. Stoll, Step-wise tile assembly with a constant number of tile types, Nat. Comput. 11 (3) (2012) 535–550.
[33] R. Möhring, Graph problems related to gate matrix layout and PLA folding, in: E. Mayr, H. Noltemeier, M. Syslo (Eds.), Computational Graph Theory, in: Computing Supplementum, vol. 7, Springer, Vienna, 1990, pp. 17–51.
[34] J. Philbin, J. Edler, O.J. Anshus, C.C. Douglas, K. Li, Thread scheduling for cache locality, SIGPLAN Not. 31 (1996) 60–71.
[35] A. Proskurowski, J.A. Telle, Classes of graphs with restricted interval models, Discret. Math. Theor. Comput. Sci. 3 (4) (1999) 167–176.
[36] N. Robertson, P.D. Seymour, Graph minors. I. Excluding a forest, J. Comb. Theory, Ser. B 35 (1) (1983) 39–61.
[37] R. Sethi, Complete register allocation problems, SIAM J. Comput. 4 (3) (1975) 226–248.

MOST WIEDZY Downloaded from mostwiedzy.pl