Intervalizing Sandwich Graphs[*]

Babette de Fluiter Hans L. Bodlaender
Department of Computer Science, Utrecht University
P.O. Box 80.089, 3508 TB Utrecht, the Netherlands
e-mail: {babette,hansb}@cs.ruu.nl

Abstract

In this report, we consider the following problem: given two graphs $G_1 = (V, E_1)$ and $G_2 = (G, E_2)$ such that $E_1 \subseteq E_2$, is there an interval graph $G' = (V, E')$ with maximum clique size at most three such that $E_1 \subseteq E' \subseteq E_2$?. We give an $O(n^2)$ algorithm for this problem.

1 Introduction

In this report we consider a graph problem which models a problem arising in molecular biology, namely INTERVALIZING SANDWICH GRAPHS or ISG. This problem is defined as follows. Given are a positive integer k and two graphs $G_1 = (V, E_1)$ and $G_2 = (V, E_2)$ with the same vertex set, such that $E_1 \subseteq E_2$. The question is whether there is an interval graph $G = (V, E)$ such that $E_1 \subseteq E \subseteq E_2$, and the maximum clique size of G is at most k. It has been shown that ISG is NP-complete [Golumbic, Kaplan, and Shamir, 1994; Fellows, Hallett, and Wareham, 1993]. From the biological application it appears that the case in which k is some fixed constant is also of interest. For these cases, we denote the problem by k-ISG. Bodlaender and de Fluiter [1996] have shown that k-ISG is also NP-complete if $k \geq 4$. However, for 2-ISG there is a simple linear time algorithm. In this report, we consider 3-ISG: we give a quadratic algorithm for this problem.

In Bodlaender and de Fluiter [1996, 1995], a restricted version of 3-ISG is discussed, namely 3-ICG, or THREE-INTERVALIZING COLORED GRAPHS. In 3-ICG, we are given a graph $G_1 = (V, E_1)$ and a three-coloring $c : V \to \{1, 2, 3\}$ of G_1, and the question is whether there is an interval graph $G = (V, E)$ with $E_1 \subseteq E$, such that G is properly colored by c. It can be seen that 3-ICG is a restricted version of 3-ISG: if you have a graph $G_1 = (V, E_1)$ and a three-coloring c of G, then this three-coloring can also be represented by the graph $G_2 = (V, E_2)$ with

$$E_2 = \{\{u, v\} \mid u, v \in V \wedge c(u) \neq c(v)\}$$

[*]This research was partially supported by the Foundation for Computer Science (S.I.O.N) of the Netherlands Organisation for Scientific Research (N.W.O.) and by ESPRIT Long Term Research Project 20244 (project ALCOM IT: *Algorithms and Complexity in Information Technology*).

It is then easy to see that G_1 and G_2 form a yes-instance for 3-ISG if and only if G_1 and c form a yes-instance for 3-ICG (note that a graph which is three-colorable has no cliques with more that three vertices). The algorithm for 3-ICG that is presented in Bodlaender and de Fluiter [1995] uses quadratic time. In this report, we generalize this algorithm for 3-ISG (although this report can be read independently of Bodlaender and de Fluiter [1995]).

The report acts as a completion of Chapter 4 of de Fluiter [1997]: that chapter discusses the algorithm for 3-ISG for the case that the input graph G_1 is biconnected. In this report, we give the complete algorithm. Therefore, this report does not contain much background information, references or preliminary results: these can all be found in de Fluiter [1997], especially in Section 2.3.1 and Chapters 3 and 4. In this report we frequently refer to results presented in de Fluiter [1997].

If an instance G_1, G_2 of 3-ISG has a solution, then the graph G_1 must have pathwidth at most two. This result is used in the algorithm for 3-ISG: the algorithm first checks whether G_1 has pathwidth at most two. If not, then false is returned. Otherwise, the structure of G_1 is used to solve 3-ISG. For this, we use the characterization of graphs of pathwidth at most two as it is presented in Chapter 3 of de Fluiter [1997]. This characterization is split into three parts: the characterization of biconnected graphs of pathwidth at most two, the characterization of trees of pathwidth at most two, and the characterization of general graphs of pathwidth at most two. This latter characterization shows how a graph of pathwidth at most two is built up from biconnected graphs of pathwidth at most two and trees of pathwidth at most two. In the algorithm, we follow this division.

This report is organized as follows. In Section 2, we give some preliminary results, and we recall some results from Chapter 2 of de Fluiter [1997] about the structure of graphs of pathwidth at most two. In Sections 3 – 6, we give an algorithm that solves 3-ISG in $O(n^2)$ time. We first give the algorithm for biconnected graphs in Section 3. In Section 4 we extend this algorithm to graphs which consist of a block with isolated vertices connected to it. After that, this algorithm is used as a building block for the algorithm for 3-ISG on input graphs which are trees that is presented in Section 5. Finally, in Section 6 we shortly discuss how this algorithm can be extended for general graphs. We do not give the complete algorithm for general graphs: this algorithm is a straightforward extension of the algorithm for trees, but it takes a lot of space.

2 Preliminaries

The graphs we consider are simple and contain no self-loops.

Definition 2.1. A sandwich graph S is a triple (V, E_1, E_2) in which (V, E_1) and (V, E_2) are simple graphs, and $E_1 \subseteq E_2$.

Definition 2.2 (Interval Graph). A graph $G = (V, E)$ is an *interval graph* if there is a function ϕ which maps each vertex of V to an interval of the real line, such that for each $u, v \in V$ with $v \neq u$,

$$\phi(u) \cap \phi(v) \neq \emptyset \Leftrightarrow \{u, v\} \in E.$$

The function ϕ is called an *interval realization* for G.

Definition 2.3 (Intervalization). Let $S = (V, E_1, E_2)$ be a sandwich graph. An *intervalization* of S is an interval graph G with $V(G) = V$ and $E_1 \subseteq E(G) \subseteq E_2$. Let $k \geq 1$. An intervalization G of S is called a k-intervalization if the maximum clique size of G is k.

In this report, the following problem is discussed [Golumbic, Kaplan, and Shamir, 1994].

INTERVALIZING SANDWICH GRAPHS (ISG)
Instance: A sandwich graph $S = (V, E_1, E_2)$, an integer $k \geq 1$
Question: Is there a k-intervalization of S?

It has been shown that ISG is NP-complete [Golumbic et al., 1994; Fellows et al., 1993]. However, from the application it appears that the cases where k is some small given constant are of interest. For fixed k, we denote the problem by k-ISG.

Bodlaender and de Fluiter [1996] have shown that k-ISG is NP-complete for $k \geq 4$ (see also de Fluiter [1997]). In this report, we resolve the complexity of k-ISG for $k \leq 3$. We observe that the case $k = 2$ is easy to resolve in $O(n)$ time. Then, we give an $O(n^2)$ algorithm that solves 3-ISG. We also show how the algorithm can be made constructive.

Definition 2.4. Let $G = (V, E)$ be a graph. A *path decomposition* of G is a sequence $PD = (V_1, V_2, \ldots, V_t)$ ($t \geq 1$) such that $V_i \subseteq V$ for each i ($1 \leq i \leq t$), and furthermore, the following holds:

1. $\bigcup_{i=1}^{t} V_i = V$,
2. for each $e \in E$, there is a node i with $e \subseteq V_i$, and
3. for each $i \leq j \leq l$, $V_i \cap V_l \subseteq V_j$.

The *width* of a path decomposition is $\max_{1 \leq i \leq t} |V_i| \Leftrightarrow 1$. The *pathwidth* of a graph G is the minimum width of any path decomposition of G.

Let $S = (V, E_1, E_2)$ be a sandwich graph. For $i = 1, 2$, the graph (V, E_i) is denoted by $G_i(S)$. We call $G_1(S)$ the *underlying graph* of S. The set of vertices of S is also denoted by $V(S)$, the first edge set by $E_1(S)$ and the second edge set by $E_2(S)$. Let $W \subseteq V$. By $S[W]$ we denote the sub-sandwich graph of S induced by W, defined as follows:

$$\begin{aligned} V(S[W]) &= W \\ E_1(S[W]) &= E_1 \cap \{\{v, w\} \mid v, w \in W\} \\ E_2(S[W]) &= E_2 \cap \{\{v, w\} \mid v, w \in W\}. \end{aligned}$$

Definition 2.5. Let $S = (V, E_1, E_2)$ be a sandwich graph. A *path decomposition* of S is a path decomposition $PD = (V_1, \ldots, V_t)$ of $G_1(S)$, such for each $v, v' \in V$, if there is a node V_i, $1 \leq i \leq t$, with $v, v' \in V_i$, then $\{v, v'\} \in E_2$. The pathwidth of S is the minimum width of any path decomposition of S.

A sandwich graph is called biconnected if its underlying graph is biconnected. A biconnected sandwich graph is also called a *sandwich block*. The blocks of a sandwich graph are the blocks of its underlying graph. A sandwich graph of which the underlying graph is a tree is called a *sandwich tree*.

The problem of k-intervalizing sandwich graphs is closely related to the pathwidth problem.

The following lemma corresponds to Lemma 2.3.3 in de Fluiter [1997].

Lemma 2.1 [Möhring, 1990]. *Let $G = (V, E)$ be a graph and let $ci(G)$ denote the least maximum clique size of any interval graph which is a supergraph of G. Then* $\mathrm{pw}(G) = ci(G) \Leftrightarrow 1$.

The following lemma corresponds to Lemma 4.2.1 in de Fluiter [1997] and is a generalization of Lemma 2.1.

Lemma 2.2. *Let $S = (V, E_1, E_2)$ be a sandwich graph and let $k \geq 1$. Sandwich graph S has pathwidth at most $k \Leftrightarrow 1$ if and only if S has a k-intervalization.*

Thus, the following problem is equivalent to ISG.

SANDWICH PATHWIDTH
Instance: A sandwich graph $S = (V, E_1, E_2)$, an integer $k \geq 1$
Question: Does S have pathwidth at most $k \Leftrightarrow 1$?

The proof of Lemma 2.2 [de Fluiter, 1997] also gives an easy way to transform a solution for one problem into a solution for the other problem. Furthermore, it implies the following result.

Corollary 2.1. *Let $k \geq 1$ and let S be a sandwich graph. If there is a k-intervalization of S then the underlying graph of S has pathwidth at most $k \Leftrightarrow 1$.*

For the case $k = 2$, the question whether there is a path decomposition of a sandwich graph S is equal to the question whether the underlying graph of S is a partial one-path (see also Fellows et al. [1993]). This is because each path decomposition of width one of $G_1(S)$ can be transformed into a path decomposition of width one of S by simply deleting all nodes which contain no edge, and then adding a node at the right side of the path decomposition for each isolated vertex containing this vertex only. Checking whether a graph has pathwidth one can be done in linear time (Chapter 3 of de Fluiter [1997]).

Theorem 2.1. 2*-ISG can be solved in linear time.*

Let G be a graph, and $PD = (V_1, \ldots, V_t)$ a path decomposition of G. Let G' be a subgraph of G. The *occurrence* of G' in PD is the subsequence $(V_j, \ldots, V_{j'})$ of PD in which V_j and $V_{j'}$ contain an edge of G', and no node V_i, with $i < j$ or $i > j'$ contains an edge of G', i.e. $(V_j, \ldots, V_{j'})$ is the shortest subsequence of PD that contains all nodes of PD which contain an edge of G'. We say that G' *occurs* in $(V_j, \ldots, V_{j'})$. The vertices of G' are said to occur in $(V_l, \ldots, V_{l'})$ if this sequence is the shortest subsequence of PD containing all vertices of G'.

Let G be a graph and $PD = (V_1, \ldots, V_t)$ a path decomposition of G. Let $1 \leq j \leq t$. We say that a node V_i is on the *left side* of V_j if $i < j$, and on the *right side* of V_j if $i > j$. Let G' be a connected subgraph of G, suppose G' occurs in $(V_l,\dots,V_{l'})$. We say that G' occurs on the left side of V_j if $l' < j$, and on the right side of V_j if $l > j$. In the same way, we speak about the left and right sides of a sequence $(V_j,\dots,V_{j'})$, i.e. a node is on the left side of $(V_j,\dots,V_{j'})$ if it is on the left side of V_j, and a node is on the right side of $(V_j,\dots,V_{j'})$ if it is on the right side of $V_{j'}$.

The following definition only makes sense if the graph G has pathwidth at most two. An edge e (or vertex v) is an *end edge* (or *end vertex*) of G' if in each path decomposition of width two of G, e (or v) occurs in the leftmost or rightmost end node of the occurrence of G'. An edge e (or vertex v) is a *double end edge* (or *double end vertex*) of G' if in each path decomposition of width two of G, e (or v) occurs in both end nodes of the occurrence of G'.

Let G be a graph, let $PD = (V_1,\dots,V_t)$ be a path decomposition of G, and let $V' \subseteq V$. Suppose $G[V']$ occurs in $(V_j,\dots,V_{j'})$, $1 \le j \le j' \le t$. The path decomposition of $G[V']$ induced by PD is denoted by $PD[V']$ and is obtained from the sequence $(V_j \cap V',\dots,V_{j'} \cap V')$ by deleting all empty nodes and all nodes $V_i \cap V'$, $j \le i < j'$, for which $V_i \cap V' = V_{i+1} \cap V'$.

Let G be a graph, and let G_1 and G_2 be subgraphs of G such that the union of G_1 and G_2 equals G. Let $PD_1 = (V_1,\dots,V_t)$ and $PD_2 = (W_1,\dots,W_{t'})$ be path decompositions of G_1 and G_2. The *concatenation* of PD_1 and PD_2 is denoted by $PD_1 + PD_2$ and is defined as follows.

$$PD_1 + PD_2 = (V_1,\dots,V_t,W_1,\dots,W_{t'})$$

Note that $PD_1 + PD_2$ is a path decomposition of G if and only if the vertices of $V(G_1) \cap V(G_2)$ occur in V_t and in W_1.

The following lemma corresponds to Lemma 3.1.1 in de Fluiter [1997]

Lemma 2.3. *Let $G = (V,E)$ be a connected partial two-path and let $V' \subseteq V$. Let $PD = (V_1,\dots,V_t)$ be a path decomposition of width two of G such that the vertices of V' occur in $(V_j,\dots,V_{j'})$. On each side of $(V_j,\dots,V_{j'})$, edges of at most two components of $G[V \Leftrightarrow V']$ occur.*

2.1 The Structure of Biconnected Partial Two-Paths

We only consider non-trivial biconnected graphs in this section.

Definition 2.6 [Bodlaender and Kloks, 1993]. Given a biconnected graph $G = (V,E)$, the *cell completion* $\overline{G}$ of G is the graph which is obtained from G by adding an edge $\{u,v\}$ for all pairs u,v of vertices in V, $u \neq v$, for which $\{u,v\} \notin E(G)$ and $G[V(G) \Leftrightarrow \{u,v\}]$ has at least three connected components.

The following lemma corresponds to Lemma 3.2.2 of de Fluiter [1997].

Lemma 2.4 [Bodlaender and Kloks, 1993]. *Let G be a biconnected partial two-path. Each path decomposition of width two of G is a path decomposition (of width two) of the cell completion $\overline{G}$ of G.*

Bodlaender and Kloks [1993] have shown that the cell completion of a biconnected partial two-tree is a ‘tree of cycles’. We show that the cell completion of a biconnected partial two-path is a ‘path of cycles’.

Definition 2.7 [Bodlaender and Kloks, 1993]. The class of *trees of cycles* is the class of graphs recursively defined as follows.

- Each cycle is a tree of cycles.
- For each tree of cycles G and each cycle C, the graph obtained from G and C by taking the disjoint union and then identifying an edge and its end vertices in G with an edge and its end vertices in C, is a tree of cycles.

Note that two different chordless cycles in a tree of cycles have at most one edge in common.

Definition 2.8. A *path of cycles* is a tree of cycles G for which the following holds.

1. Each chordless cycle of G has at most two edges which are contained in other chordless cycles of G.
2. If an edge $e \in E(G)$ is contained in $m \geq 3$ chordless cycles of G, then at least $m \Leftrightarrow 2$ of these cycles have no other edges in common with other chordless cycles, and consist of three vertices.

With each path of cycles G, we can associate a sequence $(C_1, \dots, C_p)$ of all chordless cycles of G and a sequence $(e_1, \dots, e_{p-1})$ of edges of G, such that for each i, $1 \leq i < p$, cycles C_i and C_{i+1} have edge e_i in common, and furthermore, if $i < p \Leftrightarrow 1$ and $e_i = e_{i+1}$, then C_{i+1} has three vertices.

Definition 2.9 (Cycle Path). Let G be path of cycles, let $\mathcal{C} = (C_1, \dots, C_p)$ be a sequence of chordless cycles as defined above, and let $\mathcal{E} = (e_1, \dots, e_{p-1})$ be the corresponding set of common edges. The pair $(\mathcal{C}, \mathcal{E})$ is called a *cycle path* for G.

The following theorem corresponds to Theorem 3.2.1 in de Fluiter [1997].

Theorem 2.2. *Let G be a biconnected graph. G is a partial two-path if and only if $\overline{G}$ is a path of cycles.*

Theorem 2.3. *There is an $O(n)$ time algorithm which, given a biconnected graph G, checks if the cell completion $\overline{G}$ of G is a path of cycles and constructs a cycle path for $\overline{G}$.*

The algorithm is given in Section 3.5.1 of de Fluiter [1997].

2.2 The Structure of Trees of Pathwidth Two

The following result, describing the structure of trees of pathwidth k, is similar to a result of Ellis, Sudborough, and Turner [1994]. It corresponds to Lemma 3.3.1 in de Fluiter [1997].

Lemma 2.5. *Let H be a tree and let $k \geq 1$. H is a tree of pathwidth at most k if and only if there is a path $P = (v_1, \dots, v_s)$ in H such that $H[V \Leftrightarrow V(P)]$ has pathwidth at most $k \Leftrightarrow 1$, i.e. if and only if H consists of a path with trees of pathwidth at most $k \Leftrightarrow 1$ connected to it.*

A graph has pathwidth zero if and only if it consists of a set of isolated vertices. Because graphs of pathwidth one do not contain cycles, each component of a graph of pathwidth one is a tree which consists of a path with ‘sticks’, which are vertices of degree one adjacent only to a vertex on the path (‘caterpillars with hair length one’).

The next lemmas correspond to Lemmas 3.3.2, 3.3.3 and 3.3.4 in de Fluiter [1997], respectively.

Lemma 2.6. *Let H be a tree of pathwidth k, $k \geq 1$, and suppose there is no vertex $v \in V(H)$ such that $H[V \Leftrightarrow \{v\}]$ has pathwidth $k \Leftrightarrow 1$ or less. Then there is a unique shortest path P in H such that $H[V \Leftrightarrow V(P)]$ has pathwidth $k \Leftrightarrow 1$ or less. Furthermore, P is a subpath of each path P' in H for which $H[V \Leftrightarrow V(P')]$ has pathwidth at most $k \Leftrightarrow 1$.*

Lemma 2.7. *Let H be a tree of pathwidth one, let $W \subseteq V(H)$ consist of all vertices $v \in V(H)$ for which $H[V \Leftrightarrow \{v\}]$ has pathwidth zero, and suppose that $|W| \geq 1$. Then $|W| \leq 2$, and if $|V(H)| > 2$, then $|W| = 1$.*

Lemma 2.8. *Let H be a tree of pathwidth two and let $W \subseteq V(H)$ consist of all vertices $v \in V(H)$ for which $H[V \Leftrightarrow \{v\}]$ has pathwidth at most one. Suppose $|W| \geq 1$. The following holds.*

1. *$H[W]$ is a connected graph.*
2. *If there is a $v \in W$ such that $H[V \Leftrightarrow \{v\}]$ has four or more components of pathwidth one, then $|W| = 1$.*
3. *There is a vertex $v \in W$ such that $H[V \Leftrightarrow \{v\}]$ has two or more components of pathwidth one.*
4. *$|W| \leq 7$.*

Definition 2.10. Let H be a tree and let $k \geq 1$. $\mathcal{P}_k(H)$ denotes the set of all paths P in H for which $H[V \Leftrightarrow V(P)]$ is a partial $(k \Leftrightarrow 1)$-path, and there is no strict subpath P' of P for which $H[V \Leftrightarrow V(P')]$ is a partial $(k \Leftrightarrow 1)$-path. If $|\mathcal{P}_k(H)| = 1$, then $P_k(H)$ denotes the unique element of $\mathcal{P}_k(H)$.

Let H be a tree and let $k \geq 1$. Note that if H has pathwidth more than k, then $\mathcal{P}_k(H) = \phi$. If H has pathwidth less than k, then $|\mathcal{P}_k(H)| = 1$ and $P_k(H) = ()$. If H has pathwidth exactly k then $|\mathcal{P}_k(H)| \geq 1$ and all paths in $\mathcal{P}_k(H)$ contain at least one vertex. If $\mathcal{P}_k(H)$ contains more than one element, then its elements are all paths consisting of one vertex.

For a tree of pathwidth one, all path decompositions of width one are essentially the same. The following lemma corresponds to Corollary 3.3.1 in de Fluiter [1997].

Lemma 2.9. *Let $k \geq 1$, let H be a tree of pathwidth k, and let $PD = (V_1, \ldots, V_t)$ be a path decomposition of width k of H. Let $v \in V_1$ and $v' \in V_t$. Then the path P from v to v' contains one of the paths in $\mathcal{P}_k(H)$ as a subpath.*

Theorem 2.4. *There is an $O(n)$ time algorithm which, given a tree G, checks if G has pathwidth zero, one or two, and computes $\mathcal{P}_1(H)$ if the pathwidth is one, or $\mathcal{P}_2(H)$ if the pathwidth is two.*

The algorithm is given in Section 3.5.2 of de Fluiter [1997].

3 Three-Intervalizing Sandwich Blocks

By Corollary 2.1, a sandwich graph has a three-intervalization only if the underlying graph of S has pathwidth at most two. Therefore, our algorithm for finding a three-intervalization of a sandwich graph makes use of the structure of partial two-paths as described in Chapter 3 of de Fluiter [1997] (and briefly in Section 2.1 of this report). The algorithm first checks if the underlying graph $G_1(S)$ is a partial two-path and if so, finds its structure. Then this structure is used to find a three-intervalization of S.

In this section we give the algorithm for the case that the input sandwich graph is a block. The main algorithm has the following form: first, the cell completion $\overline{G_1}(S)$ of the underlying graph of S is computed. Then, a cycle path for $\overline{G_1}(S)$ is constructed if it exists. After that, this cycle path is used to check whether there is a path decomposition of S of width at most two.

Lemma 2.4 states that each path decomposition of width two of a partial two-path G is also a path decomposition of width two of its cell completion $\overline{G}$. With respect to intervalizations, the lemma states that each three-intervalization of a sandwich graph S is a supergraph of the cell completion $\overline{G_1}(S)$ of the underlying graph $G_1(S)$ of S.

The following lemma follows directly from the results in Section 3.2 of de Fluiter [1997].

Lemma 3.1. *Let S be a sandwich block. Suppose that $G_1(S)$ is a partial two-path, $\overline{G_1}(S)$ is sandwiched in S, and (C,E) is a cycle path for $\overline{G_1}(S)$ with $C = (C_1,\dots,C_p)$ and $E = (e_1,\dots,e_{p-1})$. There is a path decomposition of S if and only if the following conditions hold:*

1. *there is a path decomposition of width two of $S[V(C_1)]$ with edge e_1 in the rightmost node (if $p > 1$),*
2. *there is a path decomposition of width two of $S[V(C_p)]$ with edge e_{p-1} in the leftmost node (if $p > 1$), and*
3. *for all i, $1 < i < p$, there is a path decomposition of width two of $S[V(C_i)]$ with edge e_{i-1} in the leftmost node and edge e_i in the rightmost node.*

Hence to check whether there is a path decomposition of width two of S with cycle path (C,E), the algorithm checks for each cycle C_i, $1 \leq i \leq p$, whether there is a path decomposition of $S[V(C_i)]$ with the appropriate edges in the leftmost and the rightmost node. The path decompositions of the sub-sandwich graphs induced by the cycles are then concatenated in the order in which they occur in C, and this gives a path decomposition of width two of S.

3.1 Cycles

We concentrate now on checking whether there exists a path decomposition of width two of a sandwich graph whose underlying graph is a cycle. Let S be such a sandwich graph and let $C = G_1(S)$.We denote the vertices and edges of C by $V(C) = \{v_0,v_1,\dots,v_{n-1}\}$, and $E(C) = \{\{v_i,v_{i+1}\} \mid 0 \leq i < n\}$ (for each i, let v_i denote $v_{i \bmod n}$). For each j and l, $1 \leq l < n$, let $I(j,l)$ denote the set of vertices of $V(C)$ between v_j and v_{j+l}, when going from v_j to v_{j+l} in positive direction, i.e.,

$$I(j,l) = \{v_i \mid j \leq i \leq j+l\}.$$

Furthermore, let $C(j,l)$ denote the cycle with

$$\begin{aligned} V(C(j,l)) &= I(j,l) \\ E(C(j,l)) &= \{\{v_j,v_{j+l}\}\} \cup \{\,\{v_i,v_{i+1}\} \mid v_i \in I(j,l) \Leftrightarrow \{v_{j+l}\}\,\} \end{aligned}$$

Note that $C(j,n \Leftrightarrow 1) = C$ for all j. For an example, consider Figure 1.

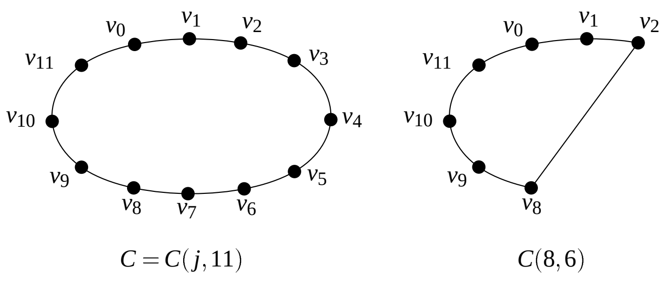

Figure 1: A cycle C with 12 vertices, and the cycle $C(8,6)$ derived from C.

The following lemma is used to obtain a dynamic programming algorithm for our problem.

Lemma 3.2. *Let $S = (V,E_1,E_2)$ be a sandwich graph whose underlying graph is a cycle C with n vertices. Let i, j and l be integers, $2 \le l < n$, and suppose $j \le i < j+l$. There is a path decomposition $PD = (V_1,\dots,V_t)$ of width two of $C(j,l)$ such that $\{v_i,v_{i+1}\} \subseteq V_1$ and $\{v_j,v_{j+l}\} \subseteq V_t$ if and only if $\{v_j,v_{j+l}\} \in E_2$ and either one of the following conditions holds:*

1. $|V(C)| = 3$,
2. *there is a path decomposition $PD' = (V'_1,\dots,V'_r)$ of width two of $S[I(j,l \Leftrightarrow 1)]$ such that $\{v_i,v_{i+1}\} \subseteq V'_1$ and $\{v_j,v_{j+l-1}\} \subseteq V'_r$, or*
3. *there is a path decomposition $PD'' = (V''_1,\dots,V''_s)$ of width two of $S[I(j+1,l \Leftrightarrow 1)]$ such that $\{v_i,v_{i+1}\} \subseteq V''_1$ and $\{v_{j+1},v_{j+l}\} \subseteq V''_s$.*

Proof. For the 'if' part, suppose $\{v_j,v_{j+l}\} \in E_2$. If $|V(C)| = 3$, then $C(j,l) = C$, and hence $(V(C))$ is a path decomposition of width two of S. Suppose there is a path decomposition $PD' = (V'_1,\dots,V'_r)$ of width two of $S[I(j,l \Leftrightarrow 1)]$ with $\{v_i,v_{i+1}\} \subseteq V'_1$ and $\{v_j,v_{j+l-1}\} \subseteq V'_r$. Then $PD = PD' + (\{v_j,v_{j+l-1},v_{j+l}\})$ is a path decomposition of width two of $S[I(j,l)]$ which satisfies the appropriate conditions. The other case is similar.

For the 'only if' part, suppose there is a path decomposition $PD = (V_1,\dots,V_t)$ of width two of $S[I(j,l)]$ such that $\{v_i,v_{i+1}\} \subseteq V_1$ and $\{v_j,v_{j+l}\} \subseteq V_t$. Clearly, $\{v_j,v_{j+l}\} \in E_2$, since $v_j,v_{j+l} \in V_t$. Suppose $|V(C)| > 3$. If $\{v_i,v_{i+1}\} = \{v_j,v_{j+l}\}$, then $l = n \Leftrightarrow 1$, hence $C(j,l) = C$ and $|I(j,l)| > 3$. Lemma 3.2.4 of de Fluiter [1997] shows that the leftmost and the rightmost node of PD can not contain the same edge, contradiction. So $\{v_i,v_{i+1}\} \neq \{v_j,v_{j+l}\}$. Let V_m and $V_{m'}$, $1 \le m,m' \le t$, be the rightmost nodes containing edge $\{v_{j+1},v_j\}$ and $\{v_{j+l-1},v_{j+l}\}$, respectively.

First suppose $m' < m$. Then $V_m = \{v_{j+1},v_j,v_{j+l}\}$, and for each k, $m < k \le t$, $v_j,v_{j+l} \in V_k$. We claim that the path decomposition obtained from $(V_1,\dots,V_m)$ by deleting v_j from each node is a path decomposition of width two of $S[I(j+1, l \Leftrightarrow 1)]$ with edge $\{v_{j+1}, v_{j+l}\}$ in the rightmost node and edge $\{v_i, v_{i+1}\}$ in the leftmost node.

Suppose there is a vertex $v \in V(C) \Leftrightarrow \{v_j, v_{j+1}\}$ which occurs on the right side of V_m. Vertex v has an edge to some vertex in $V(C) \Leftrightarrow \{v_j, v_{j+1}\}$, hence $v \in V_m$. But then $v = v_{j+l-1}$, which gives a contradiction. Hence all edges of $S[I(j+1, l \Leftrightarrow 1)]$ occur in $(V_1, \ldots, V_m)$. Furthermore, $\{v_{j+1}, v_{j+l-1}\}$ occurs in V_m. We only have to show $j \neq i$ and $j \neq i+1$. Node $V_{m'}$ contains v_{j+l}, v_{j+l-1}, and a vertex of the path from v_{j+1} to v_{i+1} which avoids v_j. Hence $v_j \notin V_{m'}$ and thus $v_j \notin V_1$. This proves the claim.

For the case that $m < m'$, a path decomposition of width two of $S[I(j, l \Leftrightarrow 1)]$ with $\{v_i, v_{i+1}\}$ in the leftmost node and $\{v_j, v_{j+l-1}\}$ in the rightmost node can be constructed in the same way.

If $m = m'$, then $v_{j+1} = v_{j+l-1}$, hence $|I(j,l)| = 3$. Since $\{v_i, v_{i+1}\} \neq \{v_j, v_{j+l}\}$, this means that $\{v_i, v_{i+1}\} = \{v_j, v_{j+1}\}$ or $\{v_i, v_{i+1}\} = \{v_{j+l-1}, v_{j+l}\}$. In the first case, $(\{v_i, v_{i+1}\})$ is a path decomposition of width two of $S[I(j, l \Leftrightarrow 1)]$ with edge $\{v_i, v_{i+1}\}$ in the leftmost node and edge $\{v_j, v_{j+l-1}\}$ in the rightmost node. In the latter case, $(\{v_i, v_{i+1}\})$ is a path decomposition of width two of $S[I(j+1, l \Leftrightarrow 1)]$ with edge $\{v_i, v_{i+1}\}$ in the leftmost node and edge $\{v_{j+1}, v_{j+l}\}$ in the rightmost node. □

Let S be a sandwich graph whose underlying graph is a cycle C. A *starting point* or *ending point* of S is an element of $E(C) \cup \{\mathsf{nil}\}$. Let $PD = (V_1, \ldots, V_t)$ be a path decomposition of S. We say that a starting point sp of S is in the leftmost node if either $sp \in E(C)$ and $sp \subseteq V_1$, or $sp = \mathsf{nil}$. We also denote this by $sp \in V_1$. Similarly, an ending point ep of S is in the rightmost node of PD, or $ep \in V_t$, if either $ep \in E(C)$ and $ep \subseteq V_t$, or $ep = \mathsf{nil}$.

We define $PW2$ as follows.

Definition 3.1. Let S be a sandwich graph of which the underlying graph is a cycle C with n vertices. Let sp be a starting point of S, and let j and l be integers, $1 \leq l < n$ and $0 \leq j < n$.

$$PW2(S, sp, j, l) = \begin{cases} \mathsf{true} & \text{if there is a path decomposition } PD = (V_1, \ldots, V_t) \\ & \text{of width two of } S[I(j,l)] \text{ with } v_j, v_{j+l} \in V_t \text{ and } sp \in V_1 \\ \mathsf{false} & \text{otherwise} \end{cases}$$

Let sp and ep be starting and ending points of a sandwich graph S of which the underlying graph is a cycle. There is a path decomposition of width two of S with sp in the leftmost node and ep in the rightmost node if and only if there is a j with $0 \leq j < n$ such that $PW2(S, sp, j, n \Leftrightarrow 1)$ holds and either $ep = \mathsf{nil}$ or $ep = \{v_{j-1}, v_j\}$.

If $n = 3$, then for any starting point sp and ending point ep, $(V(S))$ is a path decomposition of width two of S with sp in the leftmost node and ep in the rightmost node.

Suppose $n > 3$. It can be seen from the definition of $PW2$ that for all starting points sp of S, and all j, $0 \leq j < n$, $PW2(S, sp, j, 1)$ holds if and only if $sp = \mathsf{nil}$ or $sp = \{v_j, v_{j+1}\}$. We use this fact and Lemma 3.2 to describe $PW2$ recursively. Let sp be a starting point of S, and let j and l be integers with $1 \leq l < n$ and $0 \leq j < n$.

$$PW2(S, sp, j, l) = \begin{cases} sp = \mathsf{nil} \vee sp = \{v_j, v_{j+l}\} & \text{if } l = 1 \\ \{v_j, v_{j+l}\} \in E_2(S) \wedge & \\ (PW2(S, sp, j+1, l \Leftrightarrow 1) \vee PW2(S, sp, j, l \Leftrightarrow 1)) & \text{if } l > 1 \end{cases}$$

(Notice that $j+1$ denotes $(j+1)$ mod n.)

We can now use dynamic programming to compute whether there is a path decomposition of width two of S with the appropriate starting and ending points as follows.

Algorithm 3-ISG_Cycle(S, sp, ep)
Input: Sandwich graph S with $G_1(S)$ a cycle C with n vertices $v_0, \ldots, v_{n-1}$,
and edges $\{\{v_i, v_{i+1}\} \mid 0 \leq i < n\}$
Starting point sp of S
Ending point ep of S
Output: $(\exists_{0\leq j<n} (ep = \text{nil} \vee ep = \{v_{j-1}, v_j\}) \wedge PW2(S, sp, j, n \Leftrightarrow 1))$

```
1.  if $n = 3$ then return true
2.  if $sp = \text{nil}$
3.     then for $j \leftarrow 0$ to $n \Leftrightarrow 1$
4.              do $P(j,1) \leftarrow$ true
5.     else for $j \leftarrow 0$ to $n \Leftrightarrow 1$
6.              do $P(j,1) \leftarrow$ false
7.           Let $j$ be such that $sp = \{v_j, v_{j+1}\} \in E(C)$
8.           $P(j,1) \leftarrow$ true
9.  (* $\forall_{0\leq j<n} P(j,1) \equiv PW2(S, sp, j, 1)$ *)
10. for $l \leftarrow 2$ to $n \Leftrightarrow 1$
11.    do for $j \leftarrow 0$ to $n \Leftrightarrow 1$
12.             do $P(j,l) \leftarrow (\{v_j, v_{j+l}\} \in E_2(S)) \wedge (P((j+1) \bmod n, l \Leftrightarrow 1) \vee P(j, l \Leftrightarrow 1))$
13. (* $\forall_{0\leq j<n} P(j, n \Leftrightarrow 1) \equiv PW2(S, sp, j, n \Leftrightarrow 1)$ *)
14. if $ep = \text{nil}$ then return true
15. Let $j$ be such that $ep = \{v_{j-1}, v_j\}$
16. return $P(j, n \Leftrightarrow 1)$
```

The algorithm uses $O(n^2)$ time if we first build an adjacency matrix of the graph $G_2(S)$: this is needed in order to do the test in line 12 in constant time.

The algorithm can be made constructive in the sense that if there exists an intervalization, then the algorithm outputs one, as follows. Construct an array PP of pointers, such that for each j and l, $0 \leq j < n$ and $1 \leq l < n$, $PP(j,l)$ contains the nil pointer if $l = 1$ or if $P(j,l)$ is false. If $P(j,l)$ is true and $l > 1$, then $PP(j,l)$ contains a pointer to $PP(j, l \Leftrightarrow 1)$ if $P(j, l \Leftrightarrow 1)$ is true, and to $PP((j+1) \bmod n, l \Leftrightarrow 1)$ otherwise. The computation of PP can be done during the computation of P in 3-ISG_Cycle. Afterwards, if there is a three-intervalization, then one can be constructed as follows. First let G be the underlying graph of the input sandwich graph. If $ep = \text{nil}$, then start with any j, $0 \leq j < n$ for which $P(j, n \Leftrightarrow 1)$ is true, otherwise. start with j for which $ep = \{v_{j-1}, v_j\}$. Then follow the pointers from $PP(j, n \Leftrightarrow 1)$ until the nil pointer is reached, and add edge $\{v_i, v_{i+l}\}$ to G for each i and l for which $PP(i,l)$ is visited. Note that the nil pointer is reached if the previous pointer pointed to $PP(i,1)$ for some i such that either $sp = \{v_i, v_{i+1}\}$ or $sp = \text{nil}$. Hence G is a three-intervalization of the input sandwich graph.

Lemma 3.3. *Algorithm 3-ISG_Cycle solves 3-*ISG *in $O(n^2)$ time and space for sandwich graphs of which the underlying graph is a cycle.*

3.2 Blocks

Let S be a sandwich block, suppose $G_1(S)$ is a partial two-path and $\overline{G_1}(S)$ is sandwiched in S. Let (C, E) be a cycle path for $G_1(S)$ with $C = (C_1, \dots, C_p)$. There is a path decomposition of width two of S if and only if for each i, $1 \leq i \leq p$, there is a path decomposition of width two of $S[V(C_i)]$ with starting point e_{i-1} if $i > 1$, nil otherwise, and ending point e_i if $i < p$, nil otherwise (Lemma 3.1).

For a given sandwich block S, the following algorithm returns true if there is a three-intervalization of G, and false otherwise.

Algorithm 3-ISG_SB(S)
Input: Sandwich block S
Output: true if there is a three-intervalization of S, false otherwise

```
1.  Check if $\overline{G_1}(S)$ is sandwiched in $S$, and if there is a cycle path for $\overline{G_1}(S)$. If so, construct
    such a path $(C, E)$ with $C = (C_1, \dots, C_p)$ and $E = (e_1, \dots, e_{p-1})$. If not, **return** false.
2.  **for** $i \leftarrow 1$ **to** $p$
3.      **do** $m \leftarrow |V(C_i)|$
4.         **if** $i > 1$ **then** $sp \leftarrow e_{i-1}$ **else** $sp \leftarrow$ nil
5.         **if** $i < p$ **then** $ep \leftarrow e_i$ **else** $ep \leftarrow$ nil
6.         **if** $\neg$3-ISG_Cycle($S[V(C_i)]$,$sp$,$ep$) **then return** false
7.  **return** true
```

For Step 1, we can use the algorithm from Section 3.5.1 of de Fluiter [1997], which takes $O(n)$ time. The loop in lines 2 – 6 runs in $O(n^2)$ time ($n = |V(G)|$) if we first make an adjacency matrix for $G_2(S)$, and then use procedure 3-ISG_Cycle.

Algorithm 3-ISG_SB can again be made constructive. To this end, the constructive version of algorithm 3-ISG_Cycle is used in line 6. After the loop has ended, the union of the graphs that are constructed by the calls to 3-ISG_Cycle form a three-intervalization of the input sandwich graph. Hence, we have proved the main result of this section.

Theorem 3.1. *There exists an $O(n^2)$ time algorithm that solves the constructive version* 3-ISG *for sandwich blocks.*

4 Three-Intervalizing Sandwich Blocks with Sticks

The algorithm to decide 3-ISG for sandwich blocks with sticks is derived from the algorithm to decide 3-ISG for sandwich blocks. Therefore, we first consider sandwich graphs of which the underlying graph is a cycle with sticks.

4.1 Cycles with Sticks

Let $S = (V, E_1, E_2)$ be a sandwich graph such that $G_1(S)$ is a cycle C with sticks W. As is shown in Chapter 3 of de Fluiter [1997], $G_1(S)$ has pathwidth two. The following lemmas show necessary and sufficient conditions for S to have pathwidth two.

Lemma 4.1. *Let $S = (V,E_1,E_2)$ be a sandwich graph such that $G_1(S)$ is a cycle C with sticks. Let $e = \{x,y\} \in E(C)$ and $e' = \{x',y'\} \in E(C)$. Suppose there is path from x to x' which does not contain y or y', and let P_1 denote this path. Let P_2 denote the path from y to y' which does not contain x or x'.*

There is a path decomposition $PD = (V_1,\dots,V_t)$ of width two of S such that $e \subseteq V_1$ and $e' \subseteq V_t$ if and only if there is a path decomposition $PD' = (V'_1,\dots,V'_r)$ of width two of C such that

1. *$e \subseteq V'_1$ and $e' \subseteq V'_r$,*
2. *for each i, each $v,v' \in V'_i$, if $v \neq v'$, then $\{v,v'\} \in E_2$, and*
3. *for each $j \in \{1,2\}$, each $v \in V(P_j)$, each stick w of v, there is a vertex $v' \in V(P_{3-j})$ and a node V'_l such that $v,v' \in V'_l$.*

Proof. For the 'if' part, suppose $PD' = (V'_1,\dots,V'_r)$ is a path decomposition of width two of C satisfying conditions 1 – 3. We transform PD' into a path decomposition of width two of S with e in the leftmost node, and e' in the rightmost node. Because of condition 2, PD' is a path decomposition of width two of $S[V(C)]$.

First, we compute a set F of edges between vertices of C as follows. For $i = 1,2$, for each vertex $v \in V(P_i)$, and each stick w of v, let $v' \in V(P_{3-i})$ such that $\{v',w\} \in E_2$ and there is a node V'_l containing both v' and w. Add edge $\{v,v'\}$ to F. Note that $F \subseteq E_2$. Let $G = (V(C),E_1(C) \cup F)$. Clearly, PD' is a path decomposition of G. Hence G is a path of cycles. Let $(\mathcal{C},\mathcal{E})$ be a cycle path of G, with $\mathcal{C} = (C_1,\dots,C_p)$, $\mathcal{E} = (e_1,\dots,e_{p-1})$, such that $e \subseteq E(C_1)$ and $e' \subseteq E(C_p)$. Note that $F = \{e_i \mid 1 \leq i < p\}$. As is shown in Section 3.2 of de Fluiter [1997], for each i, $1 \leq i \leq p$, there is a path decomposition PD_i of width two of C_i, such that e_{i-1} is in the leftmost node of PD_i (if $i > 1$) and e_i is in the rightmost node of PD_i (if $i < p$), and furthermore $PD' = PD_1 +\!\!+ PD_2 +\!\!+ \cdots +\!\!+ PD_p$.

Let $e_0 = e$ and $e_p = e'$. Now for each vertex v, each stick w of v, do the following. Let i, $0 \leq i \leq p$, be such that $v \in e_i$ and there is a $v' \in V$ such that $e_i = \{v,v'\}$ and $\{v',w\} \in E_2$. Add a node $\{v,v',w\}$ between PD_{i-1} and PD_i (if $i = 0$, then add this node before PD_1, and if $i = p$, then add it after PD_p). The resulting path decomposition is a path decomposition of width two of S with e in the leftmost node and e' in the rightmost node.

For the 'only if' part, suppose $PD = (V_1,\dots,V_t)$ is a path decomposition of width two of S with $e \subseteq V_1$, $e' \subseteq V_t$. We show that $PD' = PD[V(C)] = (V'_1,\dots,V'_r)$ is a path decomposition of width two of C which satisfies conditions 1 – 3. Clearly, conditions 1 and 2 hold for PD'. Consider condition 3. Each node V_i contains at least one vertex of P_1 and at least one vertex of P_2. Let $v \in V(P_1)$, let w be a stick of v. Then there is a $v' \in V(P_2)$ and a node V_i, $1 \leq i \leq t$, such that $V_i = \{v,v',w\}$. Hence there is a node $V'_{i'}$ in PD' such that $1 \leq i' \leq r$ and $V'_{i'}$ contains v and v'. This completes the proof of the 'only if' part. □

Lemma 4.2. *Let $S = (V,E_1,E_2)$ be a sandwich graph such that $G_1(S)$ is a cycle C with sticks. There is a path decomposition of width two of S if and only if there are vertices $v,v' \in V(C)$ and there is a path decomposition $PD = (V_1,\dots,V_t)$ of $S' = S[V \Leftrightarrow W]$, where W is the set of sticks of v and v' in $G_1(S)$, and $v \in V_1$, $v' \in V_t$ and V_1 and V_t contain an edge of C.*

Proof. For the 'if' part, suppose there are $v, v' \in V(C)$ such that there is a path decomposition PD of width two of $S'[V \Leftrightarrow W]$, where W is the set of sticks adjacent to v and v', such that v is in the leftmost node and v' is in the rightmost node, and the leftmost and rightmost node contain an edge of C. Then we can transform PD into a path decomposition of width two of S as follows. For each stick w adjacent to v, add a node $\{v,w\}$ before of the leftmost node. If $v' \neq v$, do the same for v' after the rightmost node.

For the 'only if' part, suppose there is a path decomposition $PD = (V_1, \ldots, V_t)$ of width two of S. Suppose C occurs in $(V_j, \ldots, V_{j'})$, $1 \leq j \leq j' \leq t$. We transform PD in such a way that there is at most one $v \in V(C)$ which has a stick w such that $\{v,w\}$ occurs on the left side of V_j, and similar for the right side of $V_{j'}$. First consider the left side of PD. If no vertex of C occurs on the left side of V_j, then $j = 1$: let $v \in V(C) \cap V_j$, let W_v be the sticks of v, remove all sticks in W from PD. The new PD is a path decomposition of width two of $S[V \Leftrightarrow W_v]$ with vertex v in the leftmost node.

If only one vertex v occurs on the left side of V_j, then remove nodes $(V_1, \ldots, V_{j-1})$ from PD and remove all sticks W of v from PD. Again, the new PD is a path decomposition of width two of $S[V \Leftrightarrow W_v]$ with vertex v in the leftmost node.

If there are two vertices $u, v \in V(C)$ which occur on the left side of V_j, then $\{u,v\} \notin E(C)$, but there is a vertex $w \in V(C)$ such that $\{u,w\} \in E(C)$, $\{v,w\} \in E(C)$, and $V_j = \{u,v,w\}$ (Lemma 3.4.5 of de Fluiter [1997]). Furthermore, w has no sticks (follows from the proof of Lemma 3.4.6 of de Fluiter [1997]). Let l, $1 \leq l < j$, be the smallest integer for which V_l contains u and v. Suppose u does not occur on the left side of V_l. Now remove $V_1, \ldots, V_{l-1}$ from PD, remove all sticks W_v of v from PD, remove all occurrences of w from PD, and add a new node $\{u,v,w\}$ in front of PD. Again, the new PD is a path decomposition of width two of $S[V \Leftrightarrow W_v]$ with vertex v in the leftmost node.

Repeating the symmetrical version of this procedure on the right hand side of PD completes the proof of the 'only if' part. □

Let $S = (V, E_1, E_2)$ be a sandwich graph, such that $G_1(S)$ is a cycle C with sticks W. Let $V(C) = \{v_0, v_1, \ldots, v_{n-1}\}$ such that $E(C) = \{\{v_i, v_{i+1}\} \mid 0 \leq i < n\}$ (for each i, let v_i denote $v_{i \bmod n}$). For each i, $0 \leq i < n$, let W_i denote the set of sticks of vertex v_i. Let j, l be integers, $1 \leq l < n$. Recall from Section 3 that $I(j,l) = \{v_j, v_{j+1}, \ldots, v_{j+l}\}$, and $C(j,l)$ is the cycle with

$$\begin{aligned} V(C(j,l)) &= I(j,l) \\ E(C(j,l)) &= \{\{v_i, v_{i+1}\} \mid j \leq i < j+l\} \cup \{\{v_j, v_{j+l}\}\}. \end{aligned}$$

Let $S(j,l) = (V(j,l), E_1(j,l), E_2(j,l))$ be the sandwich graph defined as follows.

$$\begin{aligned} V(j,l) &= I(j,l) \cup \bigcup_{i=j+1}^{j+l-1} W_i \\ E_1(j,l) &= \{\{v,v'\} \in E_1 \mid v, v' \in V(j,l)\} \\ E_2(j,l) &= \{\{v,v'\} \in E_2 \mid v, v' \in V(j,l)\} \end{aligned}$$

Additionally, let $G_i(j,l) = G_i(S(j,l))$, for $i = 1, 2$. Note that the sticks of v_j and v_{j+l} are not included in $S(j,l)$. Figure 2 gives an example of $S(10,8)$ for the case that the underlying cycle

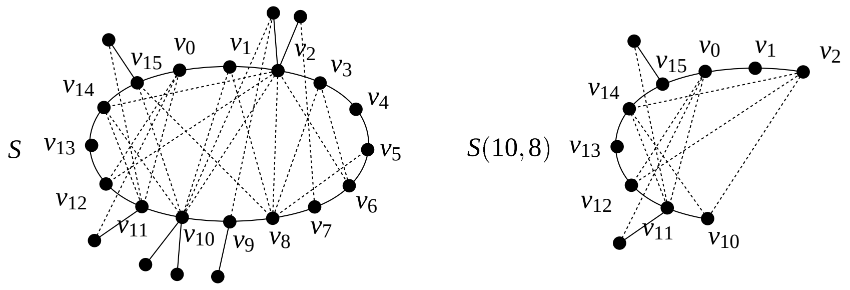

Figure 2: A sandwich graph S for which $G_1(S)$ is a cycle C with sticks ($|V(C)| = 16$), and the graph $S(10,8)$.

of S has 16 vertices (the solid lines depict the edges of $E_1(10,8)$, and the dashed lines depict the edges of $E_2(10,8)$ which are not in $E_1(10,8)$).

A *starting point* or *ending point* of S is an element of $V(C) \cup E(C) \cup \{\mathsf{nil}\}$. Let $PD = (V_1,\ldots,V_t)$ be a path decomposition of S. We say that a starting point sp of S is in the leftmost node, or $sp \in V_1$, if either $sp \in E(C)$ and $sp \subseteq V_1$, $sp \in V(C)$ and $sp \in V_1$, or $sp = \mathsf{nil}$. Similarly, an ending point ep of S is in the rightmost node of PD, or $ep \in V_t$, if either $ep \in E(C)$ and $ep \subseteq V_t$, $ep \in V(C)$ and $ep \in V_t$, or $ep = \mathsf{nil}$.

We use a dynamic programming method for solving 3-ISG on sandwich graphs of which the underlying graph is a cycle with sticks which is similar to the method that is given in Section 3 for the case that the underlying graph is a cycle. Therefore, we modify the definition of $PW2$ given in Definition 3.1 as follows.

Definition 4.1. Let $S = (V,E_1,E_2)$ be a sandwich graph such that $G_1(S)$ is a cycle C with sticks W, let sp be a starting point of S. Then for each j,l, $1 \le l < n$, $PW2(S,sp,j,l)$ is a record with fields *ft* and *lt*. Both *ft* and *lt* have two fields: *ok*, which is a boolean, and *st*, which is a set of vertices (sticks). They are defined as follows.

$PW2(S,sp,j,l).ft.ok =$

$$\begin{cases} \mathsf{true} & \text{if } \exists_{PD=(V_1,\ldots,V_t)}\ PD \text{ is a path decomposition of width two} \\ & \text{of } S[V(j,l) \cup W_j]\ \wedge v_j, v_{j+l} \in V_t \wedge sp \in V_1 \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$PW2(S,sp,j,l).ft.st = W'_{j+l}$, where $W'_{j+l} = \emptyset$ if $PW2(S,sp,j,l).ft.ok$ is false, otherwise, W'_{j+l} is a maximal subset of W_{j+l} for which there is a path decomposition $PD = (V_1,\ldots,V_t)$ of $S[V(j,l) \cup W'_{j+l} \cup W_j]$, such that $v_j, v_{j+l} \in V_t$ and $sp \in V_1$.

$PW2(S,sp,j,l).lt.ok =$

$$\begin{cases} \mathsf{true} & \text{if } \exists_{PD=(V_1,\ldots,V_t)}\ PD \text{ is a path decomposition of width two} \\ & \text{of } S[(j,l) \cup W_{j+l}] \wedge v_j, v_{j+l} \in V_t \wedge sp \in V_1 \\ \mathsf{false} & \text{otherwise} \end{cases}$$

$PW2(S,sp,j,l).lt.st = W'_j$, where $W'_j = \emptyset$ if $PW2(S,sp,j,l).lt.ok$ is false, otherwise, W'_j is a maximal subset of W_j for which there is a path decomposition $PD = (V_1,\dots,V_t)$ of $S[V(j,l) \cup W'_j \cup W_{j+l}]$, such that $v_j, v_{j+l} \in V_t$ and $sp \in V_1$.

We say that, for given j and l, the sticks of v_j are *processed* if $PW2(G,sp,j,l).ft.ok$ is true, and the sticks of v_l are processed if $PW2(G,sp,j,l).lt.ok$ is true.

Given a starting point *sp* and an ending point *ep*, there is a path decomposition of width two of S with *sp* in the leftmost node and *ep* in the rightmost node if and only if one of the following three conditions holds.

1. $ep =$ nil and there is a j, $0 \leq j < n$, such that $PW2(G,sp,j,n \Leftrightarrow 1).lt.ok$ or $PW2(G,sp,j+1,n \Leftrightarrow 1).ft.ok$ holds.
2. there is a j, $0 \leq j < n$, such that $ep = v_j$ and $PW2(G,sp,j,n \Leftrightarrow 1).lt.ok$ or $PW2(G,sp,j+1,n \Leftrightarrow 1).ft.ok$ holds.
3. There is a j, $0 \leq j < n$, such that $ep = \{v_j, v_{j+1}\}$, $PW2(S,sp,j+1,n \Leftrightarrow 1).ft.ok$ holds, and $PW2(S,sp,j+1,n \Leftrightarrow 1).ft.st = W_j$.

The definitions of $PW2(S,sp,j,l).ft.st$ and $PW2(S,sp,j,l).lt.st$ may seem strange, because their values do not have to be unique. However, in the following lemma, we show that they are in fact unique, and hence they are not only maximal but even maximum.

Lemma 4.3. *Let $S = (V,E_1,E_2)$ be a sandwich graph, such that $G_1(S)$ is a cycle C with sticks W. Let sp be a starting point. Let j and l be integers, $1 \leq l < n$, let $W'_j \subseteq W_j$ and $W'_{j+l} \subseteq W_{j+l}$. The following holds.*

1. *If there is a path decomposition $PD = (V_1,\dots,V_t)$ of $S[V(j,l) \cup W_j \cup W'_{j+l}]$, such that $sp \in V_1$ and $\{v_j, v_{j+l}\} \in V_t$, then $W'_{j+l} \subseteq PW2(S,sp,j,l).ft.st$.*
2. *If there is a path decomposition $PD = (V_1,\dots,V_t)$ of $S[V(j,l) \cup W'_j \cup W_{j+l}]$, such that $sp \in V_1$ and $\{v_j, v_{j+l}\} \in V_t$, then $W'_j \subseteq PW2(S,sp,j,l).lt.st$.*

Proof. We only show 1. Let $W'_{j+l} \subseteq W_{j+l}$. Suppose $PD = (V_1,\dots,V_t)$ is a path decomposition of $S[V(j,l) \cup W_j \cup W'_{j+l}]$, such that $sp \in V_1$ and $v_j, v_{j+1} \in V_t$. Let i be such that edge $\{v_i, v_{i+1}\}$ is an edge occurring in the leftmost node of the occurrence of C in PD and either $a =$ nil, $sp = \{v_i, v_{i+1}\}$, or $sp \subseteq \{v_i, v_{i+1}\}$ (note that such an i always exists: if *sp* is a vertex, then then the leftmost node of the occurrence of C in PD contains an edge containing *sp*). By definition, $PW2(S,sp,j,l).ft.ok$ holds. Suppose $PD' = (V'_1,\dots,V'_r)$ is a path decomposition of $S[V(j,l) \cup W_j \cup PW2(S,sp,j,l).ft.st]$, such that $ep \in V'_1$, and $v_j, v_{j+l} \in V'_r$. Let i' be such that edge $\{v_{i'}, v_{i'+1}\}$ is an edge occurring in the leftmost node of the occurrence of C in PD' and either $ep =$ nil, $ep =\subseteq \{v_{i'}, v_{i'+1}\}$, or $ep = \{v_{i'}, v_{i'+1}\}$.

Let $P = (v_j, v_{j+1},\dots,v_i)$, and let $P' = (v_j, v_{j+1},\dots,v_{i'})$. For each stick $w \in W'_{j+l}$, there is a node $\{w, v_{j+l}, v_m\}$ in PD, where $j \leq m \leq i$ (according to Lemma 4.1). Similarly for each stick $w' \in PW2(S,sp,j,l).ft.st$, there is a node $\{w', v_{j+l}, v_{m'}\}$ in PD' for some $j \leq m' \leq i'$.

Let m, $j \leq m \leq i$, be the largest integer for which there is node in PD which contains v_{j+l} and v_m, and let m', $j \leq m' \leq i'$, be the largest integer for which there is a node in PD' which contains v_{j+l} and $v_{m'}$. Then in PD, for each i, $j \le i \le m$, there is node containing v_i and v_{j+l}, and in PD', for all vertices $v_{i'}$, $j \le i' \le m'$, there is a node containing $v_{i'}$ and v_{j+l}. Let

$$\begin{aligned} W &= \{\, w \in W_{j+l} \mid \exists_{j \le s \le m} \{v_s, w\} \in E_2 \,\}, \text{ and} \\ W' &= \{\, w \in W_{j+l} \mid \exists_{j \le s \le m'} \{v_s, w\} \in E_2 \,\}. \end{aligned}$$

Clearly, $W'_{l+j} \subseteq W$, and, because $PW2(S,sp,j,l).ft.st$ is maximal, $PW2(S,sp,j,l.ft.st) = W'$. If $m' \le m$, then $W' \subseteq W$, but as $PW2(S,sp,j,l).ft.st$ is maximal, this means that $W' = W$. Hence $W'_{j+l} \subseteq PW2(S,sp,j,l).ft$. If $m \le m'$, then $W \subseteq W'$, and hence $W'_{j+l} \subseteq PW2(S,sp,j,l).ft.st$. □

We now give a recursive definition of $PW2$, called $RPW2$. We first give the definition, and after that, we show equivalence of $PW2$ and $RPW2$.

Definition 4.2. Let $S = (V, E_1, E_2)$ be a sandwich graph such that $G_1(S)$ is a cycle C with sticks W. Suppose $|V(C)| > 3$. Let sp be a starting point of C. Then for each j, l, $j \ne l$, $RPW2(S,sp,j,l)$ is a record with fields ft and lt. Both ft and lt have two fields ok, which is a boolean, and st, which is a set of vertices (sticks). They are defined as follows.

$$(RPW2(S,sp,j,1).ft.ok, RPW2(S,sp,j,1).ft.st) =$$
$$\begin{cases} (\mathsf{true}, W_{j+1}) \\ \qquad \text{if } (sp = \mathsf{nil} \lor sp = v_{j+1}) \land \forall_{w \in W_j} \{w, v_{j+1}\} \in E_2 \\ (\mathsf{true}, \{w \in W_{j+1} \mid \{w, v_j\} \in E_2\}) \\ \qquad \text{if } ((sp = v_j) \lor (sp = \{v_j, v_{j+1}\} \land \forall_{w \in W_j} \{w, v_{j+1}\} \in E_2)) \land \\ \qquad\quad \neg((sp = \mathsf{nil} \lor sp = v_{j+1}) \land \forall_{w \in W_j} \{w, v_{j+1}\} \in E_2) \\ (\mathsf{false}, \emptyset) \\ \qquad \text{otherwise} \end{cases}$$

$$(RPW2(S,sp,j,1).lt.ok, RPW2(S,sp,j,1).lt.st) =$$
$$\begin{cases} (\mathsf{true}, W_j) \\ \qquad \text{if } (sp = \mathsf{nil} \lor sp = v_j) \land \forall_{w \in W_{j+l}} \{w, v_j\} \in E_2 \\ (\mathsf{true}, \{w \in W_j \mid \{w, v_{j+1}\} \in E_2\}) \\ \qquad \text{if } ((sp = v_{j+1}) \lor (sp = \{v_j, v_{j+1}\} \land \forall_{w \in W_{j+1}} \{w, v_j\} \in E_2)) \land \\ \qquad\quad \neg((sp = \mathsf{nil} \lor sp = v_j) \land \forall_{w \in W_{j+1}} \{w, v_j\} \in E_2) \\ (\mathsf{false}, \emptyset) \\ \qquad \text{otherwise} \end{cases}$$

Furthermore, for $l \geq 1$,

$$(RPW2(S,sp,j,l+1).ft.ok, RPW2(S,sp,j,l+1).ft.st) =$$
$$\begin{cases} (\text{true}, \{\, w \in W_{j+l} \mid \{w,v_j\} \in E_2 \,\} \cup RPW2(S,sp,j+1,l).ft.st) \\ \qquad \text{if } \{v_j, v_{j+l+1}\} \in E_2 \wedge RPW2(S,sp,j+1,l).ft.ok \wedge \forall_{w \in W_j}\ \{v_{j+l+1}, w\} \in E_2 \\ (\text{true}, \{\, w \in W_{j+l} \mid \{w,v_j\} \in E_2 \,\}) \\ \qquad \text{if } \{v_j, v_{j+l+1}\} \in E_2 \wedge RPW2(S,sp,j,l).lt.ok \wedge \\ \qquad\quad (\forall_{w \in W_j}\ \{v_{j+l+1}, w\} \in E_2 \vee w \in RPW2(S,sp,j,l).lt.st) \wedge \\ \qquad\quad \neg(RPW2(S,sp,j+1,l).ft.ok \wedge \forall_{w \in W_j}\ \{v_{j+l+1}, w\} \in E_2) \\ (\text{false}, \emptyset) \\ \qquad \text{otherwise} \end{cases}$$

and

$$(RPW2(S,sp,j,l+1).lt.ok, RPW2(S,sp,j,l+1).lt.st) =$$
$$\begin{cases} (\text{true}, \{\, w \in W_j \mid \{w,v_{j+l+1}\} \in E_2 \,\} \cup RPW2(S,sp,j,l).lt.st) \\ \qquad \text{if } \{v_j, v_{j+l+1}\} \in E_2 \wedge RPW2(S,sp,j,l).lt.ok \wedge \forall_{w \in W_{j+l}}\ \{v_j, w\} \in E_2 \\ (\text{true}, \{\, w \in W_j \mid \{w,v_{j+l+1}\} \in E_2 \,\}) \\ \qquad \text{if } \{v_j, v_{j+l+1}\} \in E_2 \wedge RPW2(S,sp,j+1,l).ft.ok \wedge \\ \qquad\quad (\forall_{w \in W_{j+l}}\ \{v_j, w\} \in E_2 \vee w \in RPW2(S,sp,j+1,l).ft.st) \wedge \\ \qquad\quad \neg(RPW2(S,sp,j,l).lt.ok \wedge \forall_{w \in W_{j+l}}\ \{v_j, w\} \in E_2) \\ (\text{false}, \emptyset) \\ \qquad \text{otherwise} \end{cases}$$

We now prove the equivalence of $PW2$ and $RPW2$.

Theorem 4.1. *Let $S = (V, E_1, E_2)$ be a sandwich graph such that $G_1(S)$ is a cycle C with sticks W, and $|V(C)| > 3$. Let sp be a starting point. For each j and l, $1 \leq l < n$, $PW2(S,sp,j,l) = RPW2(S,sp,j,l)$.*

Proof. The proof is similar to the proof of Lemma 3.2, but it contains some additional difficulties. We use induction on l. Clearly, $PW2(S,sp,j,1) = RPW2(S,sp,j,1)$.

Suppose $l \geq 1$, and for all $l' \leq l$, $PW2(S,sp,j,l') = RPW2(S,sp,j,l')$. We only show that $PW2(S,sp,j,l+1).ft = RPW2(S,sp,j,l+1).ft$. For $PW2(S,sp,j,l+1).lt$, the proof is analogous.

We first show that (I) if $RPW2(S,sp,j,l+1).ft.ok$ holds, then $PW2(S,sp,j,l+1).ft.ok$ holds and $RPW2(S,sp,j,l+1).ft.st \subseteq PW2(S,sp,l+1).ft.st$. After that, we show that (II) if $PW2(S,sp,j,l+1).ft.ok$ holds, then $RPW2(S,sp,j,l+1).ft.ok$ holds and $PW2(S,sp,j,l+1).ft.st \subseteq RPW2(S,sp,l+1).ft.st$.

I. Suppose $RPW2(S,sp,j,l+1).ft.ok$ holds. Then $\{v_j, v_{j+l+1}\} \in E_2$ and either

1. $RPW2(S,sp,j+1,l).ft.ok$ holds and $\forall_{w \in W_j}\ \{v_{j+l+1}, w\} \in E_2$, or
2. 1 does not hold, but $RPW2(S,sp,j,l).lt.ok$ and for all $w \in W_j$, $\{v_{j+l+1}, w\} \in E_2$ or $w \in RPW2(S,sp,j,l).lt.st)$.

First suppose 1 holds. By the induction hypothesis, $PW2(S,sp,j+1,l).ft.ok$ holds, and $RPW2(S,sp,j+1,l).ft.st = PW2(S,sp,j+1,l).ft.st$. Let $PD = (V_1,\dots,V_t)$ be a path decomposition of width two of $S[V(j+1,l) \cup W_{j+1} \cup PW2(S,sp,j+1,l).ft.st]$, such that there $sp \in V_1$ and $v_{j+1}, v_{j+l+1} \in V_t$. Let $w_1,\dots,w_m$ denote all vertices of W_j, and let $u_1,\dots,u_p$ denote the set

$$\{u \in W_{j+l+1} \mid \{v_j, u\} \in E_2 \wedge u \notin RPW2(S,sp,j+1,l).ft.st\}.$$

Let

$$\begin{aligned} PD' = PD &+\!\!+ (\{v_{j+1}, v_j, v_{j+l+1}\}) +\!\!+ (\{v_j, v_{j+l+1}, w_1\},\dots,\{v_j, v_{j+l+1}, w_m\}) \\ &+\!\!+ (\{v_j, v_{j+l+1}, u_1\},\dots,\{v_j, v_{j+l+1}, u_p\}). \end{aligned}$$

Then PD' is a path decomposition of width two of $S[V(j,l+1) \cup W_j \cup RPW2(S,sp,j,l+1).ft.st]$ with v_j and v_{j+l+1} in the rightmost node, and sp in the leftmost node. So $PW2(S,sp,j,l+1)ft..ok$ holds, and $RPW2(S,sp,j,l+1).ft.st \subseteq PW2(S,sp,j,l+1).ft.st$, because of Lemma 4.3.

Now suppose 2 holds. By the induction hypothesis, $PW2(S,sp,j,l).lt.ok$ holds, and

$$RPW2(S,sp,j,l).lt.st = PW2(S,sp,j,l).lt.st.$$

Let $PD = (V_1,\dots,V_t)$ be a path decomposition of width two of

$$S[V(j,l) \cup W_{j+l} \cup PW2(S,sp,j,l).lt.st],$$

such that $sp \in V_1$ and $v_j, v_{j+l} \in V_t$. Let

$$\begin{aligned} \{w_1,\dots,w_m\} &= \{w \in W_j \mid w \notin RPW2(S,sp,j,l).lt,st\}, \quad \text{and} \\ \{u_1,\dots,u_p\} &= \{u \in W_{j+l+1} \mid \{v_j, u\} \in E_2\}. \end{aligned}$$

Let

$$\begin{aligned} PD' = PD &+\!\!+ (\{v_j, v_{j+l}, v_{j+l+1}\}) +\!\!+ (\{v_j, v_{j+l+1}, w_1\},\dots,\{v_j, v_{j+l+1}, w_m\}) \\ &+\!\!+ (\{v_j, v_{j+l+1}, u_1\},\dots,\{v_j, v_{j+l+1}, u_p\}). \end{aligned}$$

Then PD' is a path decomposition of width two of $S[V(j,l+1) \cup W_j \cup RPW2(S,sp,j,l+1).ft.st]$ with v_j and v_{j+l+1} in the rightmost node, and sp in the leftmost node, $a \in sp$. So $PW2(S,sp,j,l+1).ft.ok$ holds, and $RPW2(S,sp,j,l+1).ft.st \subseteq PW2(S,sp,j,l+1).ft.st$, because of Lemma 4.3. This completes the proof of part I.

II. Suppose $PW2(S,sp,j,l+1).ft.ok$ holds. We show that $RPW2(S,sp,j,l+1).ft.ok$ holds and $PW2(S,sp,j,l+1).ft.st \subseteq RPW2(S,sp,j,l+1).ft.st$. Let $PD = (V_1,\dots,V_t)$ be a path decomposition of width two of $S[V(j,l+1) \cup W_j \cup PW2(S,sp,j,l+1).ft.st]$ such that $sp \in V_1$, and $\{v_j, v_{j+l+1}\} \subseteq V_t$. Let i be such that $\{v_i, v_{i+1}\}$ occurs in the leftmost node of the occurrence of C and either $sp = \mathsf{nil}$, $sp = \{v_i, v_{i+1}\}$, or $sp = \{v_i, v_{i+1}\}$.

Clearly, $\{v_j, v_{j+l+1}\} \in E_2$, since $v_j, v_{j+l+1} \in V_t$. If $\{v_i, v_{i+1}\} = \{v_j, v_{j+l+1}\}$, then $|I(j,l+1)| = |V(C)| > 3$, and the leftmost and the rightmost node of the occurrence of C can not contain the same edge, contradiction. So $\{v_i, v_{i+1}\} \neq \{v_j, v_{j+l+1}\}$. Let V_m and $V_{m'}$, $1 \leq m, m' \leq t$, be the rightmost nodes containing edge $\{v_{j+1}, v_j\}$ and $\{v_{j+l}, v_{j+l+1}\}$, respectively.

If $m = m'$, then $v_{j+1} = v_{j+l}$, hence $|I(j,l+1)| = 3$. Since $\{v_i, v_{i+1}\} \neq \{v_j, v_{j+l}\}$, this means that $\{v_i, v_{i+1}\} = \{v_{j+l}, v_{j+l+1}\}$ or $\{v_i, v_{i+1}\} = \{v_j, v_{j+1}\}$. We prove the first case in the same way as the case that $m' < m$, and the latter case in the same way as the case that $m < m'$.

Suppose $m' < m$ or $m' = m$ and $\{v_i, v_{i+1}\} = \{v_{j+l}, v_{j+l+1}\}$. Then $V_m = \{v_{j+1}, v_j, v_{j+l+1}\}$. Furthermore, for each k, $m < k \leq t$, V_k contains v_j, v_{j+l+1}, and possibly a stick of v_j or v_{j+l+1}, since if there is a V_k, $m < k \leq t$, such that $v \in V_k$ for some other vertex of $S(j,l+1)$, then $v \in V_m$, which gives a contradiction (see also the proof of Lemma 3.2).

Note that, if $m' < m$, then $v_j \notin V_{m'}$, since $V_{m'}$ contains v_{j+l}, v_{j+l-1}, and a vertex of the path from v_{j+1} to v_{i+1} which avoids v_j. Hence if $m' < m$, then v_j does not occur in the leftmost node of the occurrence of C, so $v_j \notin \{v_i, v_{i+1}\}$. If $m = m'$, then $v_j = v_{i+2}$, which also means that $v_j \notin \{v_i, v_{i+1}\}$. Furthermore, for all k, if $v_j \in V_k$, then $v_{j+l+1} \in V_k$, so for all sticks $w \in W_j$, $\{v_{j+l+1}, w\} \in E_2$. Also, for each k, $m < k \leq t$, V_k contains only sticks $w \in W_{j+l+1}$ for which $\{v_j, w\} \in E_2$.

Let W'_{j+l} be the set of vertices containing all sticks of v_{j+l} which occur in $(V_1, \ldots, V_m)$. Let PD' be the path decomposition obtained from $(V_1, \ldots, V_m)$ by deleting v_j and its sticks from all nodes containing it. PD' is a path decomposition of width two of $S[V(j+1,l) \cup W_{j+1} \cup W'_{j+l+1}$, and a is contained in the leftmost node, $\{v_{j+1}, v_{j+l+1}\}$ in the rightmost node. Hence $PW2(S,sp,j+1,l).ft.ok$. By the induction hypothesis, this means that $RPW2(S,sp,j+1,l).ft.ok$ holds, and $RPW2(S,sp,j+1,l).ft.st = PW2(S,sp,j+1,l).ft.st$. Since $\{v_{j+l+1}, w\} \in E_2$ for all $w \in W_j$, this means that $RPW2(S,sp,j,l+1).ft.ok$ holds. $W'_{j+l+1} \subseteq PW2(S,sp,j+1,l).ft.st = RPW2(S,sp,j+1,l).ft.st$, hence $PW2(S,sp,j,l+1).ft \subseteq W'_{j+l+1} \cup \{w \in W_{j+l+1} \mid \{v_j, w\} \in E_2\} \subseteq RPW2(S,sp,j,l+1).ft.st$.

Now suppose $m < m'$, or $m = m'$ and $\{v_i, v_{i+1}\} = \{v_j, v_{j+1}\}$. Then, analogously to the other case, $V_{m'} = \{v_j, v_{j+l}, v_{j+l+1}\}$. Furthermore, for each k, $m' < k \leq t$, V_k contains v_j, v_{j+l+1}, and possibly a stick of v_j or v_{j+l+1}, but no other vertices. Also, $v_{j+l+1} \notin \{v_i, v_{i+1}\}$.

Furthermore, for all k, if $v_{j+l+1} \in V_k$, then $v_j \in V_k$, so for all sticks $w \in PW2(S,sp,j,l+1).ft.st$, $\{v_j, w\} \in E_2$. Also, for each k, $m' < k \leq t$, V_k contains only sticks $w \in W_j$ for which $\{v_{j+l+1}, w\} \in E_2$.

Let W'_j be the set of vertices containing all sticks of v_j which occur in $(V_1, \ldots, V_{m'})$. Let PD' be the path decomposition obtained from $(V_1, \ldots, V_{m'})$ by deleting v_{j+l+1} and its sticks from all nodes containing it. Then PD' is a path decomposition of width two of $S[V(j,l) \cup W_{j+l} \cup W'_j]$, and a is contained in the leftmost node, $\{v_j, v_{j+l}\}$ is in the rightmost node. Hence $PW2(S,sp,j,l).lt.ok$. By the induction hypothesis, this means that $RPW2(S,sp,j,l).lt.ok$ holds, and $RPW2(S,sp,j,l).lt.st = PW2(S,sp,j,l).lt.st$. Furthermore, $\{v_{j+l+1}, w\} \in E_2$ for all $w \in W_j$ for which $w \notin PW2(S,sp,j,l).lt.st$. There are two cases.

1. $RPW2(S,sp,j+1,l).ft.ok$ holds and for all $w \in W_j$, $\{v_{j+l+1}, w\} \in E_2$,
2. $RPW2(S,sp,j+1,l).ft.ok$ does not hold or there is a $w \in W_j$ for which $\{v_{j+l+1}, w\} \notin E_2$.

In case 1, $RPW2(S,sp,j,l+1).ft.ok$ holds, and $RPW2(S,sp,j,l+1).ft.st = \{w \in W_{j+l+1} \mid \{w, v_j\} \in E_2\} \cup RPW2(S,sp,j+1,l).ft.st$ by definition, and since $PW2(S,sp,j,l+1).ft.st \subseteq$

$\{ w \in W_{j+l+1} \mid \{v_j, w\} \in E_2 \}$, this means that $PW2(S,sp,j,l+1).ft.st \subseteq RPW2(S,sp,j,l+1).ft.st$.

In case 2, $RPW2(S,sp,j,l).lt.ok$ holds and for all $w \in W_j$, either $w \in RPW2(S,sp,j,l).lt.st$ or $\{v_j, w\} \in E_2$, hence $RPW2(S,sp,j,l+1).ft.ok$ holds and since $PW2(S,sp,j,l+1).ft.st \subseteq \{ w \in W_{j+l+1} \mid \{v_j, w\} \in E_2 \}$, this again means that $PW2(S,sp,j,l+1).ft.st \subseteq RPW2(S,sp,j,l+1).ft.st$.

This completes the proof of part II. □

For a given sandwich graph S for which $G_1(S)$ is a cycle with sticks we can modify algorithm 3-ISG_Cycle that is given in Section 3 in order to compute $PW2$. We call the resulting algorithm 3-ISG_CWS. This algorithm has the following specification.

Algorithm 3-ISG_CWS(S, sp, ep)
Input: Sandwich graph S of which $G_1(S)$ is a cycle C with sticks
Starting point sp of S
Ending point ep of S
Output: true if there is a path decomposition of width two of S with sp in the leftmost node and ep in the rightmost node.

The modifications in the computation of 3-ISG_CWS with respect to 3-ISG_Cycle follow straightforwardly from the definition of $RPW2$, and hence we do not give them here. If we compute an adjacency matrix of the graph $G_2(S)$ before running 3-ISG_CWS, then we can make 3-ISG_CWS to run in $O(n^2)$ time with $O(n^2)$ space. It is again easy to make 3-ISG_CWS also output a three-intervalization if one exists. Hence we have proved the following lemma.

Lemma 4.4. *There exists an $O(n^2)$ time algorithm that solves 3-ISG for sandwich graphs of which the underlying graph is a cycle with sticks.*

For the algorithm for sandwich blocks with sticks, we also need a slightly different result. Therefore, we construct an algorithm 3-ISG_CWS′ with the following specification.

Algorithm 3-ISG_CWS′(S, sp, u, v)
Input: Sandwich graph S of which $G_1(S)$ is a cycle C with sticks
Starting point sp of S
Vertices $u, v \in E(C)$ such that $\{u, v\} \in E(C)$
Output: A pair (ok, st), where
ok is a boolean which is true if and only if there is a path decomposition of width two of $S[V(S) \Leftrightarrow W_v]$, where W_v is the set of sticks of v, with sp in the leftmost node, u and v in the rightmost node, and
$st \subseteq W_v$, such that $st = \emptyset$ if $ok =$ false, and st is the largest subset of W_v for which there is a path decomposition of width two of $S[V(S) \Leftrightarrow W_v \cup st]$ with sp in the leftmost node and u and v in the rightmost node.

Note that the output is well-defined, i.e. it is unique for a given input. From the previous results, it is easy to see that we can construct 3-ISG_CWS′ in such a way that it runs in $O(n^2)$, $(n = |V(S)|)$.

We additionally obtain the following result, which will prove useful for deciding 3-ISG on sandwich trees.

Lemma 4.5. *Let S be a sandwich graph of which the underlying graph is a path P with sticks. Let u_1 and u_2 be the end points of P and let $\{v,w\} \in E(P)$, such that the path from u_1 to v does not contain w. Let P_1 denote the path from u_1 to v and P_2 the path from u_2 to w. For each $x \in V(P_2)$, let P^x denote the path from w to x. Let $V_x = V(P_1) \cup V(P_x)$ and let W_x denote the set of all sticks connected to V_x, except the sticks connected to x. See part I of Figure 3 for an example of $G_1(S)$ and $G_1(S[V_x \cup W_x])$ (the fat lines denote the paths P_1 and P_x).*

In $O(n^2)$ time, we can compute the vertex $y \in V(P_2)$ for which $|V(P_y)|$ is minimal and for which there is a path decomposition of width two of $S[V_y \cup W_y]$ with e in the leftmost node and u_1 and y in the rightmost node.

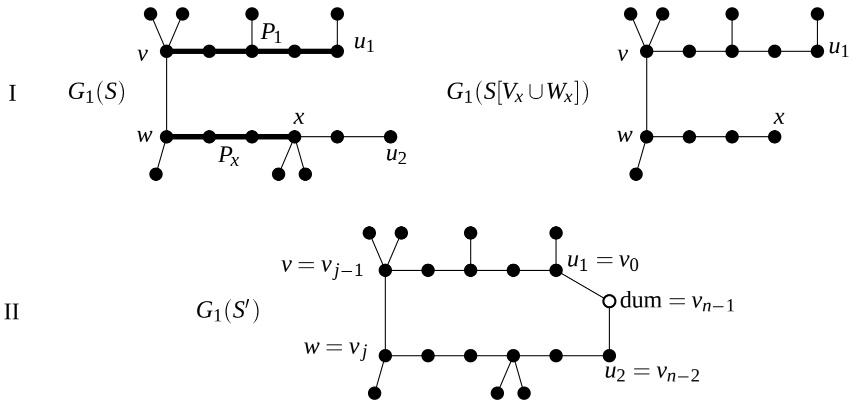

Figure 3: An example of $G_1(S)$ and $G_1(S[V_x \cup W_x])$ for Corollary 4.5 (part I), and of $G_1(S')$ for the proof of Corollary 4.5 (part II).

Proof. Let dum denote a dummy vertex and let S' be the sandwich graph obtained from S defined as follows.

$$\begin{aligned} V(S') &= V(S) \cup \{\mathsf{dum}\} \\ E_1(S') &= E_1(S) \cup \{\{u_1, \mathsf{dum}\}, \{u_2, \mathsf{dum}\}\} \\ E_2(S') &= E_2(S) \cup \{\{v, \mathsf{dum}\} \mid v \in V(S)\} \end{aligned}$$

Part II of Figure 3 shows an example of $G_1(S')$ for sandwich graph S of part I of the figure. Note that the underlying graph of S' is a cycle C with sticks. Number the vertices of C by $v_0, \ldots, v_{n-1}$ such that $E(C) = \{\{v_i, v_{i+1}\} \mid 0 \leq i < n\}$ and furthermore, $u_1 = v_0$, $\mathsf{dum} = v_{n-1}$ and $u_2 = v_{n-2}$. Let j, $0 \leq j < n$, be such that $w = v_j$.

It can be seen that finding y is boils down to finding the smallest l, $j \leq l \leq n \Leftrightarrow 2$, for which $PW2(S', \{e\}, 0, l).ft.ok$ holds. This value can easily be derived from the table that is built in algorithm 3-ISG_CWS. □

4.2 Sandwich Blocks with Sticks

We now consider sandwich graphs of which the underlying graph is a block with sticks. The algorithm for this case has the same structure as the algorithm given in Section 3 for sandwich blocks: first it checks if the input sandwich graph S has the right structure, i.e. if its block is a path of cycles. Then for each sub sandwich graph which is induced by a chordless cycle with sticks it computes whether there is a path decomposition with the correct edges in the leftmost and rightmost nodes, and it combines these results to get an answer for the complete sandwich graph.

For sandwich blocks, it is possible to first compute the results for each chordless cycle separately, and after that, combine these results. However, for sandwich blocks with sticks, this is not possible: we first compute results for the first chordless cycle in the path of cycles, then, with these results, we compute results for the first and the second cycle. With these results, we compute results for the first, second and third cycle, etc.

In Section 5 and 6, we use the algorithm for sandwich blocks with sticks as a building block. As input, we usually give a sandwich block with sticks S, a vertex or edge of the block of S which must occur in the leftmost node of the path decomposition, and a vertex of edge of the block which must occur in the rightmost node of the path decomposition. Therefore, we extend the notion of starting and ending points for sandwich blocks with sticks. Let S be a sandwich block with sticks, let B denote the block of $G_1(S)$. A starting point or ending point of S is an element of $V(B) \cup E(B) \cup \{\mathsf{nil}\}$. The algorithm is now as follows

```
Algorithm 3-ISG_SBWS($S, sp, ep$)
Input: Sandwich block with sticks $S$
       Starting point $sp$ of $S$
       Ending point $ep$ of $S$
Output: true if there is a three-intervalization of $S$, false otherwise
1.  if $G_1(S)$ is a cycle with sticks then return 3-ISG_CWS($S$,$sp$,$ep$)
2.  Compute the cell completion $\overline{G_1}(S)$ of $G_1(S)$, the block $B$ of $\overline{G_1}(S)$ and the set $W$ of sticks
    of $B$.
3.  Check if $\overline{G_1}(S)$ is sandwiched in $S$, and if $B$ is a path of cycles. If not, return false.
4.  Find a cycle path $(C,E)$ of $B$ with $C = (C_1, \dots, C_p)$ and $E = (e_1, \dots, e_{p-1})$, such that
    $sp \in V(C_1) \cup (E(C_1) \Leftrightarrow \{e_1\}) \cup \{\mathsf{nil}\}$, and $ep \in V(C_p) \cup (E(C_p) \Leftrightarrow e_{p-1}) \cup \{\mathsf{nil}\}$. If this is
    not possible, return false
5.  for $i \leftarrow 1$ to $p \Leftrightarrow 1$
6.      do Let $v_i$, $v'_i$, $W_i$ and $W'_i$ be such that $e_i = \{v_i, v'_i\}$, $W_i$ is the set of sticks of $v_i$, and $W'_i$
           is the set of sticks of $v'_i$.
7.         $S_i \leftarrow S[V(C_i) \cup \{$ sticks of $V(C_i) \Leftrightarrow W_{i-1} \Leftrightarrow W'_{i-1}\}]$.
8.  Let $ft$, $ft_1$, $ft_2$, $lt$, $lt_1$ and $lt_2$ be variables, each with a field $ok$ which is a boolean, and a
    field $st$ which is a set of vertices.
9.  $ft \leftarrow$ 3-ISG_CWS'($S_1$,$sp$,$v_1$,$v'_1$)
10. $lt \leftarrow$ 3-ISG_CWS'($S_1$,$sp$,$v'_1$,$v_1$)
11. if $\neg(ft.ok \vee lt.ok)$ then return false
12. $i \leftarrow 1$
13. while $i \leq p \Leftrightarrow 1$
```

```
14.     do $ft_1, ft_2, lt_1, lt_2 \leftarrow$ (false, $\emptyset$)
15.        if $ft.ok$
16.           then $ft_1 \leftarrow$ 3-ISG_CWS$'(S[V(S_i) \cup (W'_{i-1} \Leftrightarrow ft.st)], e_{i-1}, v_i, v'_i)$
17.                $lt_1 \leftarrow$ 3-ISG_CWS$'(S[V(S_i) \cup (W'_{i-1} \Leftrightarrow ft.st)], e_{i-1}, v'_i, v_i)$
18.        if $lt.ok$
19.           then $ft_2 \leftarrow$ 3-ISG_CWS$'(S[V(S_i) \cup (W_{i-1} \Leftrightarrow lt.st)], e_{i-1}, v_i, v'_i)$
20.                $lt_2 \leftarrow$ 3-ISG_CWS$'(S[V(S_i) \cup (W_{i-1} \Leftrightarrow lt.st)], e_{i-1}, v'_i, v_i)$
21.        (* either $ft_1.st \subseteq ft_2.st$ or $ft_2.st \subseteq ft_1.st$ *)
22.        (* either $lt_1.st \subseteq lt_2.st$ or $lt_2.st \subseteq lt_1.st$ *)
23.        $ft.ok \leftarrow ft_1.ok \vee ft_2.ok$
24.        $ft.st \leftarrow ft_1.st \cup ft_2.st$
25.        $lt.ok \leftarrow lt_1.ok \vee lt_2.ok$
26.        $lt.st \leftarrow lt_1.st \cup lt_2.st$
27.        if $\neg(ft.ok \vee lt.ok)$ then return false
28.        $i \leftarrow i+1$
29.  if $ft.ok$ and 3-ISG_CWS$(S[V(S_p) \cup (W'_{p-1} \Leftrightarrow ft.st)], e_{p-1}, ep)$
30.     then return true
31.  if $lt.ok$ and 3-ISG_CWS$(S[V(S_p) \cup (W_{p-1} \Leftrightarrow lt.st)], e_{p-1}, ep)$
32.     then return true
33.  return false
```

For each i, $1 \le i \le p$, let $V_i = V(S_1) \cup \cdots \cup V(S_i)$. After the ith iteration ($1 \le i < p$) of the main loop in Lines 13 – 28, *ft* and *lt* have the following values.

- $ft.ok =$ true if and only if there is a path decomposition of width two of $S[V(S_1) \cup \cdots \cup V(S_i) \Leftrightarrow W'_i]$ with *sp* in the leftmost node and e_i in the rightmost node, and $ft.st \subseteq W'_i$ is the largest set for which there is a path decomposition of width two of $S[V(S_1) \cup \cdots \cup V(S_i) \Leftrightarrow W'_i \cup ft.st]$ with *sp* in the leftmost node and e_i in the rightmost node.

- $lt.ok =$ true if and only if there is a path decomposition of width two of $S[V(S_1) \cup \cdots \cup V(S_i) \Leftrightarrow W'_i]$ with *sp* in the leftmost node and e_i in the rightmost node, and $lt.st \subseteq W_i$ is the largest set for which there is a path decomposition of width two of $S[V(S_1) \cup \cdots \cup V(S_i) \Leftrightarrow W_i \cup lt.st]$ with *sp* in the leftmost node and e_i in the rightmost node.

This implies that 3-ISG_BWS correctly computes whether there is a path decomposition of width two of the input sandwich graph with the desired vertices or edges in the leftmost or rightmost node.

Suppose at the beginning of the algorithm, we are given an adjacency matrix of the graph $G_2(S)$. Then the total running time of the algorithm is $O(n^2)$. Hence we have the following result, which will be frequently used in Sections 5 and 6.

Lemma 4.6. *Let S be a sandwich block with sticks. Let sp be a starting point of S and ep an ending point of S. It takes $O(n^2)$ time to check whether there is a path decomposition of width two of S with sp in the leftmost node and ep in the rightmost node.*

As a corollary, we also have the following result.

Corollary 4.1. *There exists an $O(n^2)$ time algorithm that solves 3-ISG for sandwich blocks with sticks.*

For Sections 5 and 6, we also need a slightly different result. Let $S = (V, E_1, E_2)$ be a sandwich graph. We call S a *sandwich block with sticks and loose ends u_1 and u_2* if $u_1, u_2 \in V$, $\{u_1, u_2\} \notin E_1$, and $S' = (V, E_1 \cup \{\{u_1, u_2\}\}, E_2 \cup \{\{u_1, u_2\}\})$ is a sandwich block with sticks.

For an example of the underlying graph of a sandwich block with sticks and loose ends u_1 and u_2, see part I of Figure 4.

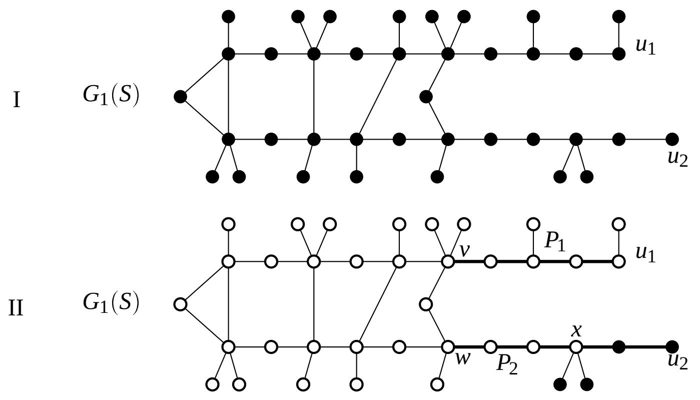

Figure 4: An example of $G_1(S)$ (part I) and of the subset V^x of $V(S)$ (part II) for Corollary 4.2.

From Theorem 3.5.1 of de Fluiter [1997] and Lemma 4.5, we can now derive the following result.

Corollary 4.2. *Let $S = (V, E_1, E_2)$ be a sandwich block with sticks and loose ends u_1, u_2, let $sp \in V(S) \cup E(S) \cup \{\mathsf{nil}\}$. Let $S' = (V, E_1 \cup \{\{u_1, u_2\}\}, E_2 \cup \{\{u_1, u_2\}\})$. We can check in $O(n)$ time whether the following conditions hold (see part I of Figure 4 for an example).*

1. *$G_1(S')$ has pathwidth two,*
2. *the cell completion $\overline{S'}$ of S' is sandwiched by S',*
3. *there is a cycle path $(\mathcal{C}, \mathcal{E})$ for $\overline{S'}$, $\mathcal{C} = (C_1, \ldots, C_p)$ and $\mathcal{E} = (e_1, \ldots, e_{p-1})$, in which $u_1, u_2 \in C_p$, $\{u_1, u_2\} \in E(C_p) \Leftrightarrow \{e_{p-1}\}$, and $sp \in V(C_1) \cup (E(C_1) \Leftrightarrow \{e_1\}) \cup \{\mathsf{nil}\}$.*

Suppose conditions 1 – 3 hold, and let $(\mathcal{C}, \mathcal{E})$ be as defined above. Let $e_{p-1} = \{v, w\}$ such that the path in S from u_1 to v does not contain w. Let P_1 denote the path from u_1 to v and P_2 the path from u_2 to w in S. For each $x \in V(P_2)$, let V^x denote the set of vertices of S which are unfilled in part II of Figure 4. Let $S^x = S[V^x]$.

In $O(n^2)$ time, we can compute the vertex $y \in V(P_2)$ for which $|V(P^y)|$ is minimum and for which there is a path decomposition of width two of S^y with u_1 and y in the rightmost node, and sp in the leftmost node.

5 Three-Intervalizing Sandwich Trees

In this section, we consider sandwich trees, i.e. sandwich graphs of which the underlying graph is a tree. The algorithm for solving 3-ISG on sandwich trees S first checks if $G_1(S)$ has path-width at most two, and if so, it finds the structure as described in Section 3.3 of de Fluiter [1997]. If not, then S does not have pathwidth at most two. Then it uses this structure to check whether S has pathwidth at most two. We mostly concentrate on this last step. So in the remainder of this section, we assume that, with a sandwich tree of pathwidth two, we are given the set $\mathcal{P}_2(G_1(S))$, and, for each $P \in \mathcal{P}_2(G_1(S))$, the set of partial one-paths which are connected to P in $G_1(S)$.

We first show that there is a path decomposition of width two of a sandwich tree S if and only if there is a path decomposition of width two of S which has some 'nice' structure. After that, we show how to compute for a given sandwich tree S of pathwidth two whether there is such a nice path decomposition of width two of S. First we distinguish different types of partial one-paths connected to a path, corresponding to the way they are connected to the path.

Definition 5.1 (Types of Partial One-Paths). Let H be a tree of pathwidth two, P a path in H such that $H[V \Leftrightarrow V(P)]$ has pathwidth one. Let $v \in V(P)$, and H' a component of $H[V \Leftrightarrow V(P)]$ such that H' has pathwidth one and has a vertex which is adjacent to v, i.e. H' is *connected* to v. Let $w \in V(H')$ be the vertex for which $\{v,w\} \in E(H)$. Let $P' \in \mathcal{P}_1(H')$. We say that

- H' is of type I if w is an end point of P', or if w is adjacent to an end point of P' and $w \notin V(P')$,
- H' is of type II if w is an inner vertex of P', and
- H' is of type III if $w \notin V(P')$ and w is adjacent to an inner vertex of P'.

Figure 5 gives an example for each type of partial one-path. The tree depicted in this figure consists of a path P with $u_1,u_2,u_3 \in V(P)$, and a partial one-path H_1 of type I connected to u_1, a partial one-path H_2 of type II connected to u_2, and a partial one-path H_3 of type III connected to u_3. Note that the type of a partial one-path H' connected to a vertex v of the path P does not

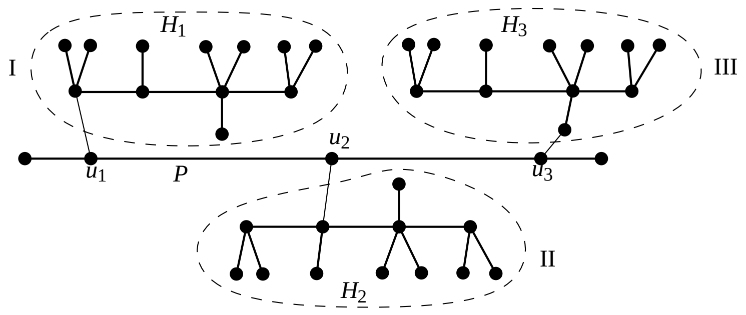

Figure 5: Example of a tree of pathwidth two which consists of a path with three partial one-paths connected to it.

depend on the choice of the path $P' \in P_1(H')$, since if $|P_1(H')| > 1$, then for each $P' \in P_1(H')$, $|V(P')| = 1$, so P' does not have any inner vertices, and hence H' has type I.

From now on, by partial one-paths connected to a path P, we only mean the partial one-paths of type I, II and III connected to P, and not the sticks connected to P.

We now give a definition of the kind of path decomposition that we want to use for the algorithm.

Definition 5.2 (Nice Path Decomposition). Let $S = (V, E_1, E_2)$ be a sandwich tree of pathwidth two, let $H = G_1(S)$, and let $PD = (V_1, \ldots, V_t)$ be a path decomposition of width two of S. Then PD is a *nice path decomposition* of width two of S if

- there are no two consecutive nodes which are equal, and
- node V_1 contains an edge $\{w, w'\} \in E_1$ and V_t contains an edge $\{x, x'\} \in E_1$, such that there is a path $P = (v_1, \ldots, v_s) \in P_2(H)$ for which there is a partial one-path H' that is connected to v_1 and a partial one-path H'' that is connected to v_s, $H' \neq H''$, $w, w' \in V(H')$, w is an end point of some path $P' \in P_1(H')$, $x, x' \in V(H'')$, and x is an end point of some path $P'' \in P_1(H'')$.

The path from w to x is called the *nice path* of S for PD.

Figure 6 shows an example of the underlying tree H of a sandwich tree S of pathwidth two and a (symbolic) nice path decomposition of width two of S. Note that $P_2(H) = (v_1, v_2, v_3)$, and the path $(u_1, u_2, u_3, u_4, u_5, u_6, u_7, u_8)$ is the nice path of S for PD. Vertex u_1 is an end point of the path $P_1(H_1)$, and u_8 is an end point of the path $P_1(H_4)$.

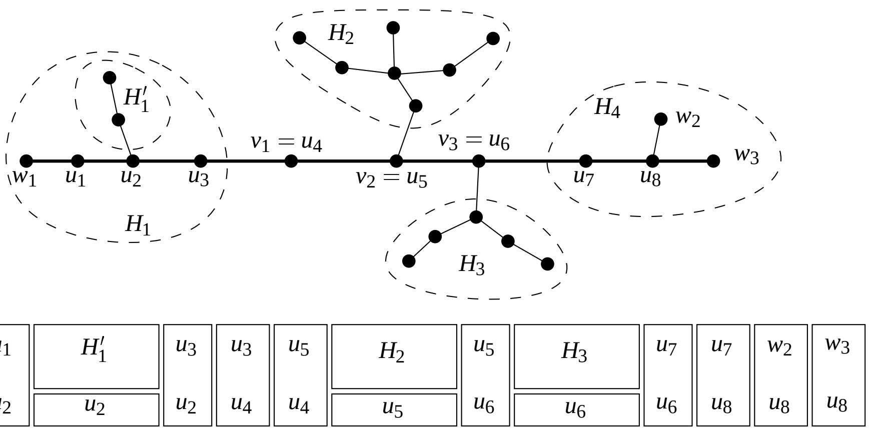

Figure 6: Example of the underlying graph H of a sandwich tree S with a nice path decomposition of S.

Next, we show that for a given sandwich tree S for which $G_1(S)$ is a tree of pathwidth two, there is a path decomposition of width two of S if and only if there is a nice path decomposition of width two of S. First we prove another lemma, which is needed for the case that $|P_2(G_1(S))| > 1$ (remember that in this case, each path in $P_2(G_1(S))$ consists of exactly one vertex).

Lemma 5.1. *Let S be a sandwich tree of pathwidth two, let $H = G_1(S)$. Suppose $|P_2(H)| > 1$. There is a path decomposition $PD = (V_1, \dots, V_t)$ of width two of S such that*

- *V_1 contains an edge $e \in E(H)$, V_t contains an edge $e' \in E(H)$, $e \neq e'$, and*
- *the shortest path P in H which contains e and e', contains a vertex $v \in V(S)$ for which $(v) \in P_2(H)$ and there are two or three components in $H[V \Leftrightarrow \{v\}]$ which have pathwidth one.*

Proof. Let $PD' = (V'_1, \dots, V'_q)$ be a path decomposition of width two of S. We transform PD into a path decomposition PD for which the condition holds. First delete the leftmost node of PD until it contains an edge, and do the same for the rightmost node of PD. Now let $e = \{u, u'\} \in E(H)$ such that $e \subseteq V_1$ and $e' = \{w, w'\} \in E(H)$ such that $e' \subseteq V_t$. Let P be the shortest path containing e and e', suppose w.l.o.g. that $P = (u, u', \dots, w', w)$. Note that $e \neq e'$, since if $e = e'$, then each vertex of H is either adjacent to v or to v', and H has pathwidth one, and so does S. However, it is possible that $u' = w'$.

If there is a $v \in V(P)$ such that $H[V \Leftrightarrow \{v\}]$ has pathwidth one and has two or three components of pathwidth one, PD' is the path decomposition that we need.

Suppose there is no $v \in V(P)$ for which this holds. We show that $H[V \Leftrightarrow V(P)]$ has exactly one component of pathwidth one. If $H[V \Leftrightarrow V(P)]$ has no components of pathwidth one, then H has pathwidth at most one. If $H[V \Leftrightarrow V(P)]$ has more than one component of pathwidth one, then there is a vertex $v \in V(P)$ such that $H[V \Leftrightarrow \{v\}]$ has more than one component of pathwidth one, which gives a contradiction.

Let H' be the component of $H[V \Leftrightarrow V(P)]$ which has pathwidth one, let $v \in V(P)$ and $v' \in V(H')$ such that $\{v, v'\} \in E(H)$. The tree $H[V \Leftrightarrow \{v\}]$ has exactly one component of pathwidth one, namely H'. This means that $v = u' = w'$ and that u and w both have degree one. Now transform PD' as follows. Delete all neighbors of v which have degree one from all nodes of PD', and for each such neighbor x, add a node $\{v, x\}$ on the left side of the leftmost node of PD'. Furthermore, delete the rightmost node from PD until it contains an edge. The resulting path decomposition is still a path decomposition of width two of S, and it satisfies the appropriate conditions, since the leftmost node contains an edge $\{v, x\}$, where x has degree one, while the rightmost node can not contain such an edge, and hence contains another edge. Hence the shortest path containing these two edges must contain a vertex y such that $H[V \Leftrightarrow \{y\}]$ has two or more components of pathwidth one. □

Theorem 5.1. *Let $S = (V, E_1, E_2)$ be a sandwich tree. Then S has pathwidth two if and only if there is a nice path decomposition of width two of S.*

Proof. Let $H = G_1(S)$. The 'if' part is clearly true.

For the 'only if' part, suppose there is a path decomposition of width two of S. If $|P_2(H)| > 1$, let $PD = (V_1, \dots, V_t)$ be a path decomposition of width two of S such that V_1 and V_t contain an edge, and the shortest path containing these edges contains a vertex v_1 for which $H[V \Leftrightarrow \{v_1\}]$ has pathwidth one, and has two or three components of pathwidth one. Furthermore, let $P = (v_1)$ ($s = 1$). If $|P_2(H)| = 1$, let $PD = (V_1, \dots, V_t)$ be an arbitrary path decomposition of width two of S, and let $P = P_2(H) = (v_1, \dots, v_s)$. Note that, by Corollary 3.3.1 of de Fluiter [1997], $H[V \Leftrightarrow V(P)]$ has at least two components of pathwidth one.

We show how PD can be ‘unfolded’ until it is a nice path decomposition of width two of S. Suppose PD is not a nice path decomposition.

First suppose $s > 1$. Let H_1 be the component of $H[V(H) \Leftrightarrow \{v_2\}]$ containing v_1, and let H_s be the component of $H[V(H) \Leftrightarrow \{v_{s-1}\}]$ containing v_s. For each $v \in V_1$ and $v' \in V_t$, the path from v to v' contains P, by Corollary 3.3.1 of de Fluiter [1997]. This means that $V_1 \subseteq V(H_1)$ and $V_t \subseteq V(H_2)$ or vice versa. If the second case holds, transform PD into rev(PD), which is the reversed path decomposition of PD that is obtained from PD by reversing the order of the nodes. Furthermore, $V(H_1) \cap V(H_2) = \emptyset$.

Suppose $s = 1$. If $|P_2(H)| = 1$, then for each $v \in V_1$ and each $v' \in V_t$, the path from v to v' contains P, and hence V_1 and V_t can not contain vertices of the same partial one-path connected to v_1. If $|P_2(H)| > 1$, then P is chosen in such a way that V_1 and V_t do not contain vertices of the same partial one-path connected to v_1. Let H_1 denote the induced subgraph of H consisting of vertex v_1 and all components of $H[V \Leftrightarrow \{v_1\}]$ of which V_1 contains a vertex, and let H_2 denote the induced subgraph of H consisting of v_1 and all components of $H[V \Leftrightarrow \{v_1\}]$ of which V_t contains a vertex. Note that again $V_1 \subseteq V(H_1)$, $V_t \subseteq V(H_2)$, and $V(H_1) \cap V(H_2) = \{v_1\}$.

The following cases may occur for V_1.

1. $V_1 = \{v, v'\}$ for some edge $\{v, v'\} \in E(H_1)$ such that v and v' both have at most one neighbor which does not have degree one.

2. V_1 contains no edge.

3. $|V_1| = 3$ and V_1 contains an edge.

4. $V_1 = \{v, v'\}$ for some edge $\{v, v'\} \in E(H_1)$, but v or v' has more than one neighbor which does not have degree one.

For V_t, the possible cases are similar.

If case 1 holds for V_1, then either v or v' has degree one. Suppose v' has degree one. Note that v and v' can not both have degree one, since then H has pathwidth one. Furthermore, $v \neq v_1$, since then v has at least two neighbors which do not have degree one, namely one neighbor in a partial one-path connected to v_1, and v_2 if $s > 1$, or a neighbor in another partial one-path connected to v_1 if $s = 1$. Furthermore, v can not be an inner vertex of $P_1(H')$ for some partial one-path H' which is connected to v_1, since then the two neighbors of v in $P_1(H')$ do not have degree one. Hence v is an end point of some path $P \in \mathcal{P}_1(H')$ for some partial one-path H' that is connected to v_1, which is exactly what we need.

Now, we repeatedly apply the transformations a, b and c described below on PD and H_1 and H_2, such that, after each transformation, the following holds.

- PD is a path decomposition of width two of S.

- H_1 and H_2 are subgraphs of the graphs H_1 and H_2 before the transformation, respectively.

- The leftmost node of PD only contains vertices of H_1, the rightmost node contains only vertices of H_2.

This means that, after each transformation, one of the cases 1 – 4 holds. We do this until case 1 holds for both V_1 and V_t, which means that *PD* is a nice path decomposition of width two of S. First, transformations a, b and c are done for V_1 and H_1 until case 1 applies for V_1, next they are done for V_t and H_2, until case 1 applies for V_t.

Transformation a. If case 2 applies, delete V_1.

Transformation b. If case 3 applies, let $e \in E(H_1)$ such that $e \subseteq V_1$, and add a node containing e only on the left side of V_1.

Transformation c. If case 4 applies, do the following. Suppose w.l.o.g. that the path from v to v_1 contains v'. Consider the components of $H[V \Leftrightarrow \{v\}]$ which consist of more than one vertex. Note that one of these components is a subgraph of H_1 which does not contain v_1 or v', and hence V_t does not contain any vertex of this component. Let H' be such a component. Now transform *PD* into rev$(PD[V(H') \cup \{v\}]) +\!\!+ PD[V \Leftrightarrow V(H')]$, and let $H_1 = H[V(H') \cup \{v\}]$. The new path decomposition is indeed a path decomposition of width two of S, since v is the only vertex that $H[V(H') \cup \{v\}]$ and $H[V \Leftrightarrow V(H')]$ have in common, and v occurs in the rightmost node of rev$(PD[V(H') \cup \{v\}])$ and in the leftmost node of $PD[V \Leftrightarrow V(H')]$. Furthermore, the new H_1 contains at least one vertex less than the old H_1, the leftmost node of the new *PD* contains only vertices of the new H_1 and the rightmost node of the new *PD* contains only vertices of H_2.

Note that the number of transformations that can be done is finite: if the transformation of case 4 is done, then H_1 or H_2 gets smaller, and after each time the transformation of case 4 is done, the transformations of case 2 and 3 can only be done a finite number of times before case 4 holds again. □

Let S be a sandwich tree such that $H = G_1(S)$ has pathwidth two. A path $P = (u_1, u_2, \dots, u_q)$ in H is called a *possible* nice path of S if

- P contains a path $(v_1, \dots, v_s) \in \mathcal{P}_2(H)$, for which
- there is a partial one-path H' connected to v_1 and a partial one-path H'' connected to v_s, $H' \neq H''$, such that u_1 is an end point of a path in $\mathcal{P}_1(H')$ and u_q is an end point of a path in $\mathcal{P}_1(H'')$.

Note that, for each nice path decomposition of width two of S with nice path P, P is a possible nice path of S.

The total number of possible nice paths in a sandwich tree S of which $G_1(S)$ has pathwidth two may be $\Omega(n^2)$, where $n = |V(H)|$. We construct an algorithm PW2, which checks for a given sandwich tree S of which $G_1(S)$ has pathwidth two whether S has pathwidth two. This algorithm has the following structure, in which algorithm Nice_Path(P) returns true if there is a nice path decomposition of width two of S with nice path P, and false otherwise.

Algorithm PW2(S)
Input: Sandwich tree S for which $G_1(S)$ has pathwidth two

```
Output: true if $S$ has pathwidth two, false otherwise
1.  for certain possible nice paths $P$ of $S$
2.      do if Nice_Path($P$) then return true
3.  return false
```

The algorithm will run in $O(n^2)$ time, because the number of nice paths that is tried is bounded by a constant, and function Nice_Path runs in $O(n^2)$ time. In the remainder of this section, we first show which possible nice paths have to be tried, and which possible nice paths do not have to be tried. After that, we show how function Nice_Path works. First, we prove some lemmas.

Lemma 5.2. *Let S be a sandwich tree of pathwidth two, let $PD = (V_1,\dots,V_t)$ be a nice path decomposition of width two of S with nice path P. Let $v \in V(P)$ and suppose H' a partial one-path connected to v, let $w \in V(H')$ such that $\{v,w\} \in E(G_1(S))$.*

1. *If H' is of type* II*, then there is an i, $1 \le i \le t$, such that*

$$PD' = (V_1,\dots,V_i,\{v,w\},V_{i+1},\dots,V_t)$$

   is a nice path decomposition of width two of S with nice path P.

2. *If H' is of type* III*, then let w' be the inner vertex of $P_1(H')$ that is adjacent to w. There is an i, $1 \le i \le t$, such that $V_i = \{v,w,w'\}$.*

Proof. Let $H = G_1(S)$. Suppose H' occurs in $(V_j,\dots,V_{j'})$. Each node V_i, $j \le i \le j'$, contains at most two vertices of H'. There is a node containing v and w, since $\{v,w\} \in E(H)$. First we prove the case that H' has type II.

1. If there is a node $V_i = \{v,w\}$, then we are done. Suppose there is no such node. Suppose $\{v,w\}$ occurs in $(V_l,\dots,V_{l'})$. Note that edges of one component of $H'[V(H') \Leftrightarrow \{w\}]$ occur on the left side of V_l and edges of another component of $H'[V(H') \Leftrightarrow \{w\}]$ occur on the right side of $V_{l'}$ (Lemma 3.3.5 of de Fluiter [1997]). Furthermore, note that v is not an end point of the path P, since, by definition, there are no partial one-paths connected to the end points of a nice path. Hence edges of one component of $H[V \Leftrightarrow V(H') \Leftrightarrow \{v\}]$ occur on the left side of V_l and edges of another component of $H[V \Leftrightarrow V(H') \Leftrightarrow \{v\}]$ occur on the right side of $V_{l'}$. No edges of $H[V \Leftrightarrow \{v,w\}]$ occur within $(V_l,\dots,V_{l'})$, since each node already contains v and w. If $v \notin V_{l-1}$, then there is a neighbor u of v in one of the four components with edges of $H[V \Leftrightarrow \{v,w\}]$ with $u \in V_l$. If $w \notin V_{l-1}$, then there is a neighbor u of w in one of the components of the four components with edges of $H[V \Leftrightarrow \{v,w\}]$.

Let u be the neighbor of v or w which occurs in V_l. Similarly, let u' be the neighbor of v or w which occurs in $V_{l'}$. Note that $u' \neq u$, since u and u' are in different components of $H[V \Leftrightarrow \{v,w,\}]$. Hence $V_l = \{v,w,u\}$ and $V_{l'} = \{v,w,u'\}$. This implies that there must be a node V_i, $l \le i < l'$, such that $V_i \cap V_{i+1} = \{v,w\}$, and hence $(V_1,\dots,V_i,\{v,w\},V_{i+1},\dots,V_t)$ is also a path decomposition of width two of S.

2. Now suppose that H' has type III. Because of the structure of path decompositions of width two, there is no node containing w but not w', since w' is an inner vertex of $P_1(H)$, and w is a stick connected to w'. Hence there must be a node containing w, w' and v, since $\{w,v\} \in E$. □

Lemma 5.3. *Let S be a sandwich tree of pathwidth two, $PD = (V_1,\dots,V_t)$ a nice path decomposition of width two of S with nice path $P = (v_1,\dots,v_q)$. Let $v_m \in V(P)$ and let $H_1,\dots,H_l$ be the partial one-paths connected to v_m. There are at most two partial one-paths in $H_1,\dots,H_l$ which have a vertex w for which $\{v,w\} \notin E_2(S)$.*

Proof. Let $H = G_1(S)$. Suppose $v_1 \in V_1$ and $v_q \in V_t$, and suppose v_m occurs in $(V_j,\dots,V_{j'})$. Let H' and H'' be the components of $H[V \Leftrightarrow \{v_m\}]$ which contain vertices of P, such that $v_1 \in V(H')$ and $v_q \in V(H'')$ (note that H' is the empty graph if and only if $m = 1$, and H'' is the empty graph if and only if $m = q$). If $m > 1$, then there is an edge of H' which occurs on the left side of $(V_j,\dots,V_{j'})$, and if $m < q$, then there is an edge of H'' which occurs on the right side of $(V_j,\dots,V_{j'})$. Lemma 3.1.1 of de Fluiter [1997] shows that there is at most one partial one-path connected to v_m of which an edge occurs on the left side of V_j, and at most one of which an edge occurs on the right side of $V_{j'}$. Hence of all other partial one-paths connected to v_m, all vertices w occur within $(V_j,\dots,V_{j'})$, which means that $\{v,w\} \in E_2(S)$. □

Lemma 5.4. *Let S be a sandwich tree of pathwidth two. Let PD be a nice path decomposition of width two of S with nice path P. There is a nice path decomposition of width two of S with nice path P in which for each $v \in V(P)$ such that there are two or more partial one-paths connected to v, PD contains a node $\{v\}$.*

Proof. Let $H = G_1(S)$ and $V = V(S)$. Let $PD = (V_1,\dots,V_t)$. For each $v \in V(P)$ for which there are two or more partial one-paths connected to v, transform PD as follows. If v is the left or right end point of P, then add a node $\{v\}$ on the left or right side of PD, respectively.

Suppose v is an inner vertex of P. Suppose v occurs in $(V_j,\dots,V_{j'})$. Let H_1 be the induced connected subgraph of H containing v and all components of $H[V \Leftrightarrow \{v\}]$ of which there is an edge occurring on the left side of V_j, and let H_2 be the induced subgraph containing v and all components of $H[V \Leftrightarrow \{v\}]$ of which there is an edge occurring on the right side of $V_{j'}$. Note that $V(H_1) \cap V(H_2) = \{v\}$, since no component of $H[V \Leftrightarrow \{v\}]$ can have edges occurring on the left side of V_j and edges occurring on the right side of $V_{j'}$.

Furthermore, let H_3 be the induced subgraph of H containing v and all components of $H[V \Leftrightarrow \{v\}]$ which are not in H_1 or H_2. Then $H = H_1 \cup H_2 \cup H_3$. If there are vertices of H_1 which occur on the right side of $V_{j'}$, then they can be deleted from these nodes, since there are no edges containing these vertices occurring on the right side of $V_{j'}$. Similarly for H_2 on the left side of V_j, and for H_3 on the right side of $V_{j'}$ and on the left side of V_j. Let PD' be the path decomposition PD after deleting these vertices. Then $PD'' = PD'[V(H_1)] +\!\!+ (\{v\}) +\!\!+ PD'[V(H_3)] +\!\!+ (\{v\}) +\!\!+ PD'[V(H_2)]$ is a nice path decomposition of width two of S with nice path P, since the rightmost node of $PD[V(H_1)]$ contains v, the leftmost node of $PD[V(H_2)]$ contains v, and all nodes of $PD[V(H_3)]$ contain v. □

The following lemmas are important to bound the number of possible nice paths that have to be tried during the algorithm.

Lemma 5.5. *Let S be a sandwich tree of pathwidth two, $H = G_1(S)$. Suppose there is no vertex $v \in V(H)$ for which $H[V \Leftrightarrow \{v\}]$ has pathwidth one. Let $P_2(H) = (v_1,\dots,v_s)$ and let PD be a nice path decomposition of width two of S with nice path $P = (u_1,\dots,u_q)$. The following holds.*

1. *If $H[V \Leftrightarrow \{v_1\}]$ has three or less components, then there is a partial one-path H' which is connected to v_1, and u_1 is an end point of some $P'' \in \mathcal{P}_1(H')$.*

2. *If $H[V \Leftrightarrow \{v_1\}]$ has four or more components, and there is a partial one-path connected to v_1 which has a vertex w for which $\{v_1, w\} \notin E_2(S)$, then there is a partial one-path H' which is connected to v_1 and which contains a vertex w for which $\{v_1, w\} \notin E_2(S)$, such that there is a nice path decomposition PD' of width two of S with nice path $P' = (w_1, \dots, w_r)$, such that $w_r = u_q$ and w_1 is end point of some $P'' \in \mathcal{P}_1(H')$.*

3. *If $H[V \Leftrightarrow \{v_1\}]$ has four or more components, and for each partial one-path H' connected to v_1, each vertex $w \in V(H')$, $\{v_1, w\} \in E_2(S)$, then for all partial one-paths H' connected to v_1, there is a nice path decomposition of width two of S with nice path $(w_1, \dots, w_r)$, such that $w_r = u_q$ and w_1 is end point of some path in $\mathcal{P}_1(H')$.*

The analog for v_s also holds.

Proof. Let $PD = (V_1, \dots, V_t)$.

1. If $H[V \Leftrightarrow \{v_1\}]$ has three of less components, then clearly condition 1 holds.

2. If $H[V \Leftrightarrow \{v_1\}]$ has four or more components, and at least one of these components has a vertex w for which $\{v_1, w\} \notin E_2(S)$, then PD is transformed as follows. Let H' be the partial one-path connected to v_1 for which $u_1 \in V(H')$. If H' contains a vertex w for which $\{v_1, w\} \notin E_2(S)$, then no transformation is performed. Otherwise, first the transformation of the proof in Lemma 5.4 is done. The resulting path decomposition $PD = (V_1, \dots, V_t)$ has a node $\{v_1\}$, and is still a nice path decomposition with nice path P. Suppose v_1 occurs in $(V_j, \dots, V_{j'})$, let l, $j \leq l \leq j'$, be such that $V_l = \{v_1\}$. For each partial one-path H'' connected to v_1 that has an edge occurring on the left side of V_j and for which for each vertex w, $\{v_1, w\} \in E_2(S)$, do the following. Make a path decomposition of width one of $S[V(H'')]$ and add v_1 to each node. The result is a path decomposition PD' of width two of $S[V(H'') \cup \{v_1\}]$. Delete all vertices of H'' from all nodes of PD, and add PD' between V_l and V_{l+1} in PD. Let PD denote the obtained path decomposition of H, and suppose again that v_1 occurs in $(V_j, \dots, V_{j'})$. If there is no partial one-path connected to v_1 of which an edge occurs on the left side of V_j, let H'' denote a partial one-path connected to v_1 which does contain a vertex w for which $\{v_1, w\} \in E_2(S)$. Then H'' occurs within $(V_j, \dots, V_t)$. Note that $v_1 \in V_1$. Let $PD' = \text{rev}(PD[V(H'') \cup \{v_1\}]) + PD[V \Leftrightarrow V(H'')]$. Now use unfolding as in the proof of Lemma 5.1 to make sure that PD is a nice path decomposition and that the end point of the nice path is an end point of a path $P'' \in \mathcal{P}_1(H'')$. Condition 2 now holds.

3. If $H[V \Leftrightarrow \{v_1\}]$ has four or more components, but for each vertex w of each partial one-path connected to v_1, $\{v_1, w\} \in E_2(S)$, then PD can be transformed as follows. First apply the transformations as in the proof of Lemma 5.4. Let V_l denote a node of PD for which $V_l = \{v_1\}$. Next, for each partial one-path H' that is connected to v_1, delete all vertices of H' from PD, make a path decomposition of width one of $S[V(H')]$, add v_1 to each node of this path decomposition, and put the obtained path decomposition of width two of $S[V(H') \cup \{v_1\}]$ between V_l and V_{l+1}. Delete all empty nodes from PD. Note that V_1 contains v_1. For each partial one-path H' connected to v_1 and for each end point w of a path $P' \in \mathcal{P}_1(H')$, we can now

make a nice path decomposition of width two of S with nice path $P = (u_1, \dots, u_q)$, such that $u_1 = w$ as follows. Make a path decomposition $PD' = (W_1, \dots, W_r)$ of width one of $S[V(H')]$, such that $w \in W_1$. Let $w' \in V(H')$ such that $\{v_1, w'\} \in E(H)$. Let m, $1 \le m \le r$, be such that W_m is the rightmost node which contains w'. If $m = 1$, then let PD' be $\text{rev}(PD')$, and let $m = r$. Add v_1 to each W_i, $i \ge m$. Let PD' denote this path decomposition. Then $PD' +\!\!+ PD[V \Leftrightarrow V(H')]$ is a nice path decomposition of width two of S that satisfies condition 3. □

The next lemma is the analog of Lemma 5.5 for the case that the underlying tree H has a vertex v for which $H[V(H) \Leftrightarrow \{v\}]$ has pathwidth one.

Lemma 5.6. *Let S be a sandwich tree of pathwidth two, $H = G_1(S)$, and suppose there is a $v \in V(H)$ for which $H[V \Leftrightarrow \{v\}]$ has pathwidth one. Let $P = (v_1) \in \mathcal{P}_2(H)$ such that $H[V(H) \Leftrightarrow \{v_1\}]$ has at least two components which have pathwidth one. Suppose there is a nice path decomposition PD of width two of S with nice path $P = (u_1, \dots, u_q)$ such that P contains v_1. Then the following holds.*

1. *If $H[V \Leftrightarrow \{v_1\}]$ has three or less components, then there are two partial one-paths H' and H'', $H' \neq H''$, connected to v_1, such that u_1 is an end point of some path in $\mathcal{P}_1(H')$, and u_q is an end point of some path in $\mathcal{P}_1(H'')$.*
2. *If $H[V \Leftrightarrow \{v_1\}]$ has four or more components and there are two or more partial one-paths connected to v_1 which have a vertex w for which $\{v_1, w\} \notin E_2(S)$, then there are two partial one-paths H' and H'', $H' \neq H''$, connected to v_1, such that H' and H'' both contain a vertex w for which $\{v_1, w\} \notin E_2(S)$, and there is a nice path decomposition of width two of S with nice path $(w_1, \dots, w_r)$ such that w_1 is an end point of some path in $\mathcal{P}_1(H')$, and w_r is an end point of some path in $\mathcal{P}_1(H'')$.*
3. *If $H[V \Leftrightarrow \{v_1\}]$ has four or more components and exactly one partial one-path H' connected to v_1 has a vertex w for which $\{v_1, w\} \notin E_2(S)$, then for each partial one-path H'' connected to v_1, $H' \neq H''$, there is a nice path decomposition of width two of S with nice path $(w_1, \dots, w_r)$ such that w_1 is an end point of some path in $\mathcal{P}_1(H')$, and w_r is an end point of some path in $\mathcal{P}_1(H'')$.*
4. *If $H[V \Leftrightarrow \{v_1\}]$ has four or more components and for each vertex w of each partial one-path connected to v_1, $\{v_1, w\} \in E_2(S)$, then for each two partial one-paths H' and H'' connected to v_1, $H' \neq H''$, there is a nice path decomposition PD' of width two of S with nice path $(w_1, \dots, w_r)$ such that w_1 is an end point of some path in $\mathcal{P}_1(H')$, and w_r is an end point of some path in $\mathcal{P}_1(H'')$.*

Proof. Similar to the proof of Lemma 5.5. □

Let S be a sandwich tree, such that $H = G_1(S)$ has pathwidth two. It now follows that the number of possible nice paths that have to be tried to find out whether there is a nice path decomposition of width two of S is bounded by a constant. Let $\mathcal{A}$ be a set of possible nice paths of S. We call $\mathcal{A}$ a set of *potentially nice paths* if there are sets $U_1, U_2 \subseteq V(S)$, for which the following conditions hold. First suppose $P_2(H) = (v_1, \dots, v_s)$ for some $s > 1$. Let $\mathcal{H}$ denote the set of all partial one-paths connected to v_1, and let $\mathcal{H}'$ denote the set of all partial one-paths connected to v_1 which have a vertex w for which $\{v_1, w\} \notin E_2(S)$.

1. $\mathcal{A} = \{P = (u_1, \dots, u_q) \mid u_1 \in U_1 \wedge u_q \in U_2 \wedge P$ is path from u_1 to $u_q\}$
2. If $|\mathcal{H}| \leq 3$, then U_1 is the set of all end points of all paths in $\mathcal{P}_1(H')$, for all $H' \in \mathcal{H}$.
3. If $|\mathcal{H}| \geq 4$ and $|\mathcal{H}'| \geq 1$, then U_1 is the set of all end points of all paths in $\mathcal{P}_1(H')$ for all $H' \in \mathcal{H}$.
4. If $|\mathcal{H}| \geq 4$ and $|\mathcal{H}'| = 0$, then there is a partial one-path $H' \in \mathcal{H}$ such that U_1 is the set of end points of all paths in $\mathcal{P}_1(H')$.
5. The analogs of conditions 2 – 4 also hold for U_2 with respect to the partial one-paths connected to v_s.

If for each $P \in \mathcal{P}_2(H)$, $P = (v)$ for some $v \in V(S)$, then we can give a similar set of conditions, derived from Lemma 5.6.

Lemmas 5.5 and 5.6 imply the following result.

Theorem 5.2. *Let S be a sandwich tree for which $H = G_1(S)$ has pathwidth two. Let $\mathcal{A}$ be a set of potentially nice paths of S. The following holds.*

- *The size of $\mathcal{A}$ is bounded by a constant.*
- *There is a (nice) path decomposition of width two of S if and only if there is a nice path decomposition of width two of S with nice path P such that $P \in \mathcal{A}$.*

Algorithm PW2 described on page 31 now looks as follows.

```
Algorithm PW2($S$)
Input: Sandwich tree $S$ for which $G_1(S)$ has pathwidth two
Output: true if $S$ has pathwidth two, false otherwise
1.   $\mathcal{A} \leftarrow$ set of potentially nice paths of $S$
2.   for all $P \in \mathcal{A}$
3.         do if Nice_Path($P$) then return true
4.   return false
```

We now concentrate on algorithm Nice_Path, which checks for a given potentially nice path whether there is a nice path decomposition of width two of S with this nice path. The basic structure of this algorithm is as follows. The algorithm walks along the given nice path $P = (v_1, \dots, v_q)$, from vertex v_1 to vertex v_q. During this walk, it 'processes' the partial one-paths that are connected to the vertex v_i that it currently passes. To be able to describe the processing step more precisely, we first further analyze the structure of a nice path decomposition of width two of a sandwich tree.

In the following discussion, let $S = (V, E_1, E_2)$ be a sandwich tree of pathwidth two, let $H = G_1(S)$, and let $PD = (V_1, \dots, V_t)$ be a nice path decomposition of S with nice path $P = (v_1, v_2, \dots, v_q)$.

We first show that the number of partial one-paths that is connected to one vertex of the nice path for which the algorithm has to perform substantial computations is bounded.

Lemma 5.7. *There is a nice path decomposition PD' of width two of S with nice path P in which for each $v \in V(P)$ for which there are at least two partial one-paths connected to v, the following holds. For each partial one-path H' that is connected to v, if H' contains only vertices w for which $\{v,w\} \in E_2$, then H' occurs within the occurrence of v in PD'.*

Proof. Follows directly from Lemmas 5.3 and 5.4. □

Lemma 5.7 and Lemma 5.3 show that if a vertex v of the nice path has two or more partial one-paths connected to it, then the algorithm has to do significant computations for at most two partial one-paths connected to v, since there are at most two of these partial one-paths which have a vertex w for which $\{v,w\} \notin E_2$.

Lemma 5.8. *There is a nice path decomposition PD' of width two of S with the same nice path P in which no two partial one-paths of $H[V \Leftrightarrow V(P)]$ overlap, i.e. for each pair of distinct partial one-paths H' and H'' connected to P, there is no node V_i containing a vertex of H' and a vertex of H''.*

Proof. Suppose there are two partial one-paths H' and H'' connected to $v \in V(P)$ and $v' \in V(P)$, respectively, for which there is a node V_m containing vertices of H' and of H''. Suppose the vertices of H' occur in $(V_j, \dots, V_{j'})$ and the vertices of H'' occur in $(V_l, \dots, V_{l'})$. It is not possible that $j \leq l \leq l' \leq j'$, since each V_i, $j \leq i \leq j'$, contains a vertex of P and a vertex of H', but H'' has pathwidth one. Similarly, it is not possible that $l \leq j \leq j' \leq l'$. Suppose w.l.o.g. that $j \leq l \leq j' \leq l'$. Let i be such that $l \leq i \leq j'$. V_i does not contain an edge of H' or an edge of H'', since H' and H'' have no vertices in common. This means that $V_{j'}, \dots, V_l$ all contain the same vertex of H', say w, the same vertex of H'', say w', and the same vertex of P, say v. Hence $j' = l$. But w and w' are not adjacent, hence V_l can be split into V'_l and V''_l, with $V'_l = \{v,w\}$, and $V''_l = \{v,w'\}$. Then $PD' = (V_1, \dots, V_{l-1}, V'_l, V''_l, V_{l+1}, \dots, V_t)$ is also a nice path decomposition of width two of width two of S with nice path P.

In this way, all overlaps can be removed from PD, which results in a nice path decomposition with nice path P, without overlapping partial one-paths. □

From now on, we assume that in any nice path decomposition, the partial one-paths connected to the nice path do not overlap, and hence this also holds for PD.

Lemma 5.9. *Let $v_m \in V(P)$, let H' be a partial one-path connected to v_m, and suppose H' occurs in $(V_j, \dots, V_{j'})$. Let $v_l \in V(P)$ be the leftmost vertex on P which occurs in $(V_j, \dots, V_{j'})$ (i.e. there is no $i < l$ for which v_i occurs in $(V_j, \dots, V_{j'})$), and $v_{l'} \in V(P)$ the rightmost.*

Then $v_l \in V_j$, $v_{l'} \in V_{j'}$, and for all i, $l < i < l'$, v_i and sticks adjacent to v_i occur only within $(V_j, \dots, V_{j'})$, and there is no partial one-path connected to v_i, except H' if $m = i$.

Proof. Node V_j contains a vertex on the path from v_1 to v_l. But V_j does not contain any vertex v_i with $1 \leq i < l$. Hence $v_l \in V_j$, and $v_{l'} \in V_{j'}$. Furthermore, V_j and $V_{j'}$ both contain an edge of H'. This means that V_j and $V_{j'}$ can not contain another vertex of $V(H) \Leftrightarrow V(H')$. Hence for each i, $l < i < l'$, it is not possible that v_i or any vertex of a stick or a partial one-path connected to v_i is an element of V_p for some p, $1 \leq p < j \vee j' < p \leq t$. So all vertices and edges on the path from v_l to $v_{l'}$ occur within $(V_j, \dots, V_{j'})$. Suppose there is a partial one-path $H'' \neq H'$ which is connected to v_i for some i, $l < i < l'$. Then H'' must occur within $(V_j, \dots, V_{j'})$. But each node in $(V_j, \dots, V_{j'})$ contains a vertex of P and a vertex of H'. This gives a contradiction. $\square$

Definition 5.3. Let $1 \leq m \leq q$, and let H' partial one-path connected to v_m, H' occurs in $(V_j, \dots, V_{j'})$. Let v_l be the leftmost vertex on P which occurs in $(V_j, \dots, V_{j'})$, and $v_{l'}$ the rightmost. We say that H' *uses* (the interval) $[l, l']$.

Figure 7 shows an example of Definition 5.3: partial one-path H' is connected to a vertex v_m of the path P. In the figure, only a part of the underlying graph H is drawn. The path $P_1(H')$ is the path from u to w. In the occurrence $(V_j, \dots, V_{j'})$ of H' in the path decomposition PD of width two, v_l, u and a stick u' of u occur in V_j, and $v_{l'}$, w and a stick w' of w occur in V_j. Hence H' uses $[l, l']$, which is shown by the dashed lines in the graph (note that the dashed lines are edges of the interval completion of PD). All vertices v_i, $l < i < l'$, and sticks adjacent to v_i occur only within $(V_j, \dots, V_{j'})$.

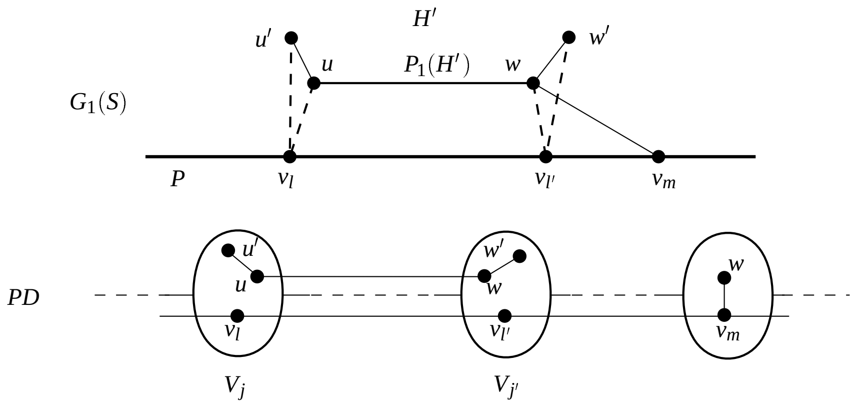

Figure 7: Example of a partial one-path H' that uses $[l, l']$.

As a corollary of Lemma 5.9, we also have the following result.

Corollary 5.1. *Let H' and H'' be two partial one-paths connected to P, suppose H' uses $[j, j']$ and H'' uses $[l, l']$. If $j' > l$, then H' occurs on the right side of H'', and if $l' > j$, then H'' occurs on the right side of H'.*

In the following corollaries, we summarize some earlier lemmas in terms of intervals.

Corollary 5.2. *Let H' be a partial one-path which is connected to v_m for some m, $1 \leq m \leq q$. Let H'' be another partial one-path which is connected to P. Suppose H' uses $[j, j']$ and H'' uses $[l, l']$. The following holds.*

1. *Either $j \geq l'$ or $l \geq j'$.*

2. *Either $l' \leq m$ or $l \geq m$.*

Proof. Part 1 follows from Lemma 5.8 and Lemma 5.9. Part 2 follows from Lemma 5.9. □

The following corollary is depicted in Figure 8.

Corollary 5.3. *Let $v_m \in V(P)$, $H_1, \dots, H_{nr}$ the partial one-paths connected to v_m. For each i, $1 \le i \le nr$, suppose H_i uses $[j_i, j_i']$.*

1. *There is at most one i, $1 \le i \le nr$, for which $j_i' > m$ and there is at most one i', $1 \le i' \le nr$, for which $j_i < m$, and all others have $j_i = j_i' = m$.*
2. *If there is an i such that $j_i < m$ and $j_i' > m$, then $nr = 1$.*
3. *If $nr \ge 2$, then PD can be transformed into nice path decomposition of width two of S with the same nice path, such that for each H_i, $1 \le i \le nr$, if each vertex $w \in V(H_i)$ has $\{v_m, w\} \in E_2$, then $j_i = j_i' = m$.*

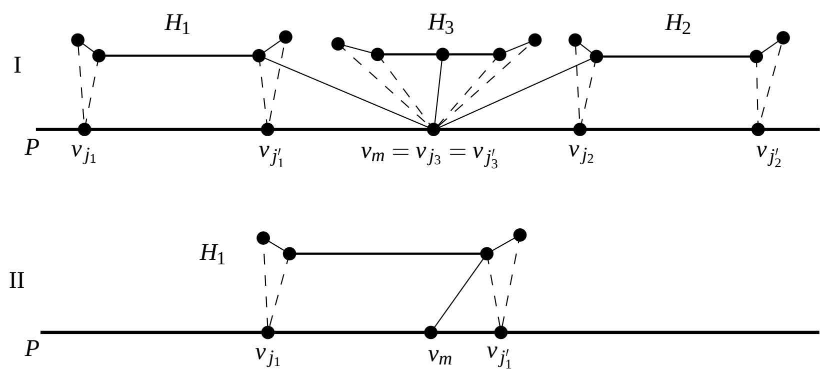

Figure 8: Example for Corollary 5.3. In Part I, $nr = 3$. In Part II, H_1 uses $[j_1, j_1']$ with $j_1 < m < j_1'$. Hence $nr = 1$.

Proof. Part 1 follows from Lemma 5.3, part 2 from Lemma 5.4, and part 3 from Lemma 5.7. □

In the next lemmas, we further bound the number of possible values for the intervals $[j, j']$ that a partial one-path connected to P can use.

Lemma 5.10. *Let $v_m, v_{m'} \in V(P)$, $m' > m$, and let H' be a partial one-path connected to v_m, H'' a partial one-path connected to $v_{m'}$. Suppose H' uses $[j, j']$, $m' \le j \le j' \le q$ and H'' uses $[l, l']$, $1 \le l \le l' \le q$. Then*

1. *either $l' \le m$ or $l \ge j'$, and*
2. *if $l \ge j'$ then H'' occurs on the right side of H' and $j' = j = m'$.*

Proof. There are three possibilities for $[l, l']$, namely

a. $1 \leq l \leq l' \leq m$,

b. $j' \leq l \leq l' \leq q$, or

c. $m \leq l \leq l' \leq j$ and neither case a nor case b holds.

We first show that case c is not possible. Suppose $m \leq l \leq l' \leq j$ and cases a and b do not hold. Suppose H' occurs in $(V_r, \dots, V_{r'})$, H'' occurs in $(V_s, \dots, V_{s'})$. See also Figure 9. Vertex v_l is the only vertex of $H[V \Leftrightarrow V(H'')]$ occurring in V_s and $m < l'$, which means that v_m does not occur in $V_{s'}$ or on the right side of $V_{s'}$. Furthermore, $v_{l'}$ is the only vertex of $H[V \Leftrightarrow V(H'')]$ occurring in $V_{s'}$ and $l < j'$, which means that vertices of H' occur on the right side of $V_{s'}$. But $V_{s'}$ does contain a vertex of H'' or vertex v_m, as can be seen from Figure 9, which gives a contradiction. Hence only cases a and b are possible, which means that condition 1 holds.

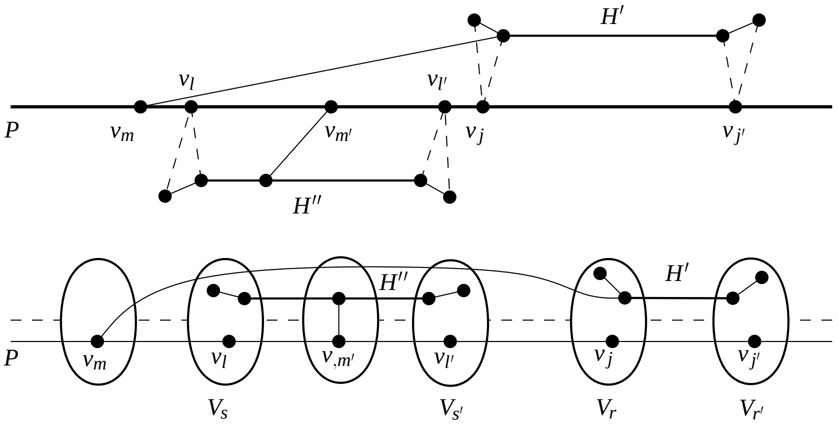

Figure 9: Example of partial one-paths H' and H'' as used in the proof of condition 1 of Lemma 5.10.

We now have to prove condition 2. Suppose that $l \geq j'$ and H'' occurs on the left side of H', see part I of Figure 10. Suppose again that H' occurs in $(V_r, \dots, V_{r'})$ and H'' occurs in $(V_s, \dots, V_{s'})$. Then $s \leq s' < r \leq r'$. $m < m' \leq l$, so v_m occurs only on the left side of V_s. But no node of $(V_s, \dots, V_{s'})$ contains a vertex of H' or v_m, which gives a contradiction. Hence H'' occurs on the right side of H', i.e. $s > r'$, see part II of Figure 10. Suppose $j' > m'$. Then $v_{m'}$ only occurs on the left side of $V_{r'}$. But $V_{r'}$ does not contain a vertex of H'', which gives a contradiction. Hence $j = j' = m'$. □

Lemma 5.11. *Let $v_m, v_{m'} \in V(P)$, $m' > m$, and let H' be a partial one-path connected to v_m, H'' a partial one-path connected to $v_{m'}$. Suppose H' uses $[j, j']$, $m' \leq j \leq j' \leq q$ and H'' uses $[l, l']$, $1 \leq l \leq l' \leq m$. Then $m' = m+1$ or $m' = m+2$ and v_{m+1} has degree two; there is a node in PD containing v_m, v_{m+1} and $v_{m'}$, and H' and H'' have type* I.

Proof. Suppose H' occurs in $(V_r, \dots, V_{r'})$ and H'' occurs in $(V_s, \dots, V_{s'})$. Then $s' < r$, since $l' < j$. Let $V_{r'} = \{v_{j'}, u, u'\}$, $u, u' \in V(H')$ and $V_s = \{v_l, w, w'\}$, $w, w' \in V(H'')$. Suppose u is

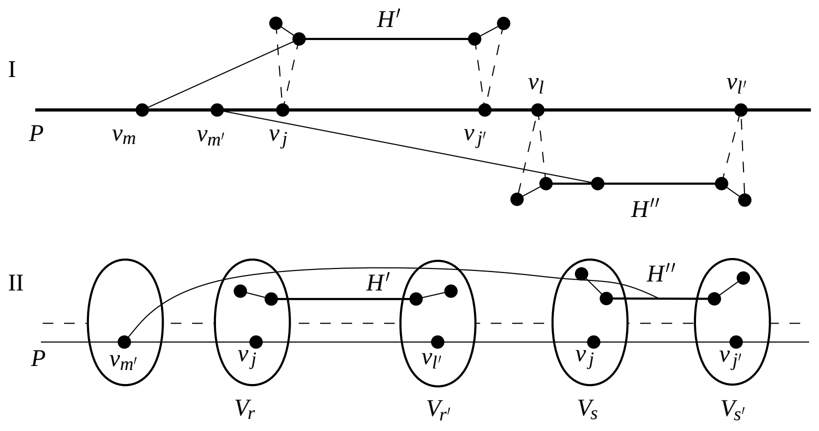

Figure 10: Example of partial one-paths H' and H'' as used in the proof of condition 2 of Lemma 5.10.

an end point of a path $P' \in P_1(H')$ and w is an end point of a path $P'' \in P_1(H'')$. See also part I of Figure 11. Vertex v_m does not occur in $(V_r,\dots,V_{r'})$, hence u and u' are not adjacent to v_m. Similarly, w and w' are not adjacent to $v_{m'}$. Let S' be the sandwich graph obtained from S by adding the edges $\{u',v_{j'}\}$ and $\{w',v_l\}\}$ to E_1. Note that S' is a sandwich graph, since $\{u',v_{j'}\},\{w',v_l\}\} \in E_2$. The path decomposition PD is also a path decomposition of S'. We first prove that $m' = m+1$ or $m' = m+2$ and v_{m+1} has degree two and that there is a node containing v_m, v_{m+1} and $v_{m'}$.

Suppose $m' > m+1$. Then $G_1(S)$ contains three disjoint paths between v_m and $v_{m'}$, as can be seen in Figure 11, part I. According to Lemma 3.2.2 of de Fluiter [1997], PD is a path decomposition of the sandwich graph S'' which is obtained from S' by adding edge $\{v_m,v_{m'}\}$. See part II of Figure 11 for graph $G_1(S'')$. Graph $G_1(S'')$ contains three chordless cycles which have edge $\{v_m,v_{m'}\}$ in common. At least one of these chordless cycles, say C, must have three vertices, and the vertex $v \in V(C)$ with $v \neq v_m,v_{m'}$ has degree two, i.e. it is only adjacent to v_m and $v_{m'}$. Cycle C can not be the cycle containing vertices of H' or H'', since u and u' are not adjacent to v_m in S, and w and w' are not adjacent to $v_{m'}$ in S. Hence it must be the cycle consisting of $v_m,v_{m+1},\dots,v_{m'}$. So either $m' = m+1$, or $m' = m+2$ and v_{m+1} has degree two. Furthermore, the two or three vertices v_m, v_{m+1} and $v_{m'}$ occur in one node.

We now prove that H' and H'' both have type I. Let C' be the chordless cycle of $G_1(S'')$ which contains v_l and let C'' be the chordless cycle of $G_1(S'')$ which contains $v_{j'}$. C' and C'' have edge $\{v_m,v_{m'}\}$ in common. All edges between vertices $v_l,\dots,v_{j'}$, edges between vertices from $v_{l+1},\dots,v_{j'-1}$ and their adjacent vertices, and all edges of H' and H'' occur within $(V_s,\dots,V_{r'})$, see part III of Figure 11. Suppose H' has type II or III, then let $v \in V(P_1(H'))$ be such that v is adjacent to v_m if H' has type II, or v has distance two to v_m if H' has type III (part II of Figure 11). Then $v \in V(C')$, and there is a vertex connected to v that does not have degree one. This means that v should occur in the leftmost node containing an edge of C'. This is node $V_{r'}$, but $V_{r'} = \{v_{j'},u,u'\}$, and $u',u \neq v$. Contradiction. □

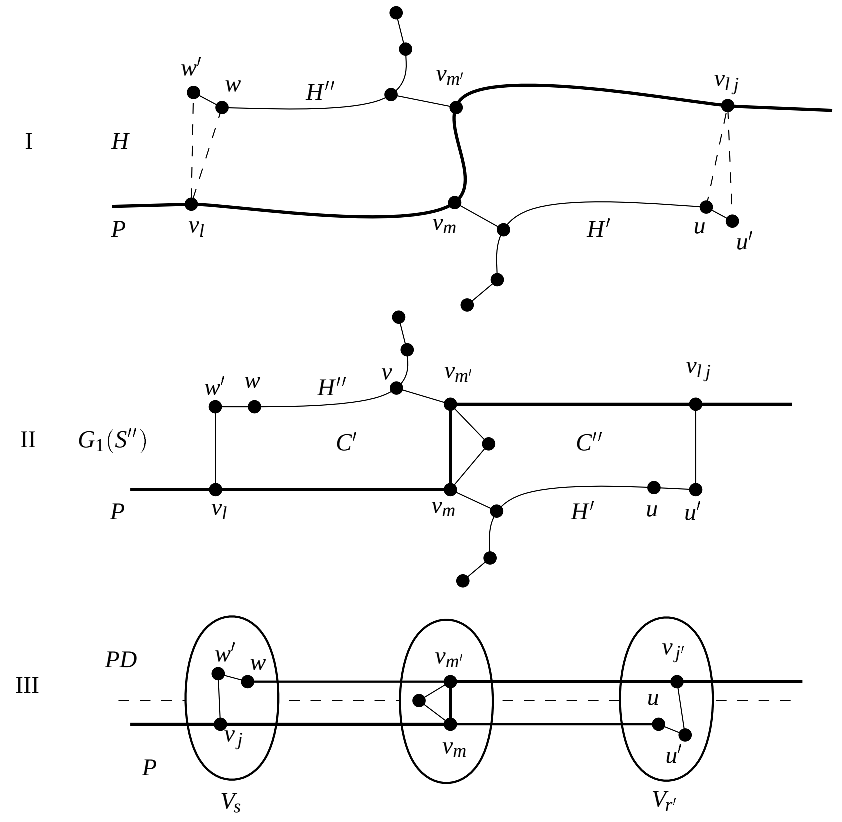

Figure 11: Example of the use of partial one-paths H' and H'' for Lemma 5.11.

Let $i_1, i_2, \dots, i_t$ be integers such that $1 \leq i_1 < i_2 < \cdots < i_t \leq q$, and

$$\{i_1, \dots, i_t\} = \{i \mid \text{there is a partial one-path connected to } v_i\}.$$

Furthermore, let $i_{-1} = i_0 = 1$ and $i_{t+1} = i_{t+2} = q$. Algorithm Nice_Path processes the sandwich tree from left to right, i.e. it starts with vertex v_{i_1}, it processes the partial one-paths connected to v_{i_1} and computes a 'partial' nice path decomposition of this. Then it goes to v_{i_2} and processes the partial one-paths connected to v_{i_2} with use of the partial nice path decomposition for v_{i_1}, and computes a new partial nice path decomposition from this, etc. We now define this partial nice path decomposition more precisely.

Definition 5.4 (Partial Nice Path Decomposition). Let $V' \subseteq V$, let $v \in V'$. Let

$$E_2' = E_2 \cup \{\{u, w\} \mid u, w \in V \wedge u \notin V'\}.$$

A *partial nice path decomposition* of $(S[V'], v)$ is a path decomposition PD of $S[V']$ with vertex v in the rightmost node, such that there is a PD' for which $PD +\!\!+ PD'$ is a nice path decomposition of the sandwich graph (V, E_1, E_2') with nice path P.

More informally, a partial nice path decomposition of $(S[V'], v)$ is a path decomposition of $S[V']$ with vertex v in the rightmost node and which can be extended to a nice path decomposition of S with nice path P if we forget about the limitations of E_2 in the rest of the graph.

We do not need all possible sets V' for partial nice path decompositions, so in the next definition, we give short cuts for the kind of sets we need.

Definition 5.5. For each m, i with $1 \leq m \leq i \leq q$, let $V_m^i \subseteq V$ be the vertex set defined as follows.

$$\begin{aligned} V_m^i = &\{v_j \mid 1 \leq j \leq i\} \cup \{w \in V(S) \mid \exists_j 1 \leq j < i \wedge w \text{ is a stick connected to } v_j\} \cup \\ &\{w \in V(S) \mid \exists_{H',j} 1 \leq j \leq m \wedge H' \text{ is a partial one-path connected to } v_j \wedge w \in V(H')\} \end{aligned}$$

Let $S_m^i = S[V_m^i]$. Furthermore, for each partial one-path H' connected to the path $v_1, \dots, v_m$, let $S_m^i \Leftrightarrow H'$ denote $S[V_m^i \Leftrightarrow V(H')]$.

The following definition gives the exact information that is computed by Nice_Path.

Definition 5.6. The information that Nice_Path computes consists of two variables *all* and *allbo*[1], both arrays from 0 to t, such that for each k, $0 \leq k \leq t$, $all[k]$ has two fields ok, which is a boolean, and min, which is an integer, and $allbo[k]$ has two fields ok, which is a boolean, and tr, which is a set of partial one-paths. After vertex v_{i_k} is processed for some k, $1 \leq k \leq t$, $all[k]$ and $allbo[k]$ have the following values (let $m = i_k$).

- $all[k].ok = \mathsf{true}$ if and only if there is a partial nice path decomposition of (S_m^j, v_j) for some j, $i_k \leq j \leq i_{k+1}$.

  If $all[k].ok = \mathsf{true}$, then $all[k]min$ denotes the smallest j, $i_k \leq j \leq i_{k+1}$, for which there is a partial nice path decomposition of (S_m^j, v_j). If $all[k].ok = \mathsf{false}$, then $all[k]min = \infty$.

[1] *all* stands for 'all partial one-paths of v_{i_k} are processed', and *allbo* stands for 'all but one partial one-paths of v_{i_k} are processed'

- $allbo[k].ok =$ true if and only if
  - there are two or more partial one-paths connected to v_m, and
  - there is a partial one-path H' connected to v_m, for which
    a. there is a partial nice path decomposition of $(S_m^m \Leftrightarrow H', v_m)$, and
    b. H' has a vertex w for which $\{v_m, w\} \notin E_2$.

    Furthermore, $all[0].ok =$ true, $all[0]min = 1$, $allbo[0].ok =$ false and $allbo[0].tr = \emptyset$.

  If $allbo[k].ok =$ true, then $allbo[k].tr$ is the set of partial one-paths H' connected to v_m for which condition a and b hold, otherwise, $allbo[k].tr = \emptyset$.

Clearly, there is a nice path decomposition of width two of S with nice path P if and only if $all[t].ok$ holds.

Algorithm Nice_Path looks as follows.

Algorithm Nice_Path(P)
Input: Path $P = (v_1, \ldots, v_q)$ which is a possible nice path of S
Output: true if there is a nice path decomposition of S with nice path P, false otherwise

```
1.  Let $i_1, \ldots, i_t$ be the set of integers $j \in \{1, \ldots, q\}$ for which there is at least one partial
    one-path connected to $v_j$, and such that for all $l$, $i_l < i_{l+1}$.
2.  $all[0].ok, all[0]min \leftarrow$ true, 1
3.  $allbo[0].ok, allbo[0].tr \leftarrow$ false, $\emptyset$
4.  for $k \leftarrow 1$ to $t$
5.      do compute $all[k]$ and $allbo[k]$ from $all[i]$ and $allbo[i]$, $i < k$
6.  return $all[t].ok$
```

In the remainder of this section, we describe the computation in line 5 in more detail. Let $k \geq 1$. Let $m = i_k$, $n = i_{k+1}$, $nn = i_{k+2}$, $p = i_{k-1}$ and $pp = i_{k-2}$. Let $H_1, \ldots, H_{nr}$ denote the partial one-paths connected to v_m.

For the computation of $all[k]$, we distinguish between two cases, namely the case that $nr > 1$ and the case that $nr = 1$. For the computation of $allbo[k]$, $allbo[k].ok =$ false if $nr = 1$, so for $allbo[k]$, we only consider the case that $nr \geq 2$.

The Computation of $all[k]$ for the Case that $nr > 1$

We first analyze the possible cases if there is a partial nice path decomposition of (S_m^a, v_a) $(m \leq a \leq n)$. Suppose a is an integer, $m \leq a \leq n$ and PD' is a partial nice path decomposition of (S_m^a, v_a). Suppose that for each i, H_i uses $[j_i, j_i']$. By Corollaries 5.1 – 5.3 and Lemma 5.11, there are two possibilities (see also Figure 12):

1. all partial one-paths connected to some v_l, $1 \leq l < m$, occur on the left side of all partial one-paths connected to v_m, or

2. $p < m$, there is one partial one-path H_c connected to v_m and one partial one-path F connected to v_p such that F occurs on the right side of H_c, all partial one-paths $H_i \neq H_c$ which are connected to v_m occur on the right side of F, and all partial one-paths $F' \neq F$ connected to some v_l, $l < m$, occur on the left side of H_i.

In the first case, for each i, either $p \leq j_i \leq j'_i \leq m$ or $m \leq j_i \leq j'_i \leq a$, and for each F' connected to v_l, $l < m$, F' uses $[b, b']$, where $b' \leq \min\{j_i \mid 1 \leq i \leq nr\}$ (part I of Figure 12). In the second case, $pp \leq j_c \leq j'_c \leq p$, for all $i \neq c$, $m \leq j_i \leq j'_i \leq a$, F uses $[m, m]$, and for all $F' \neq F$ connected to v_l, $l < m$, F' uses $[b, b']$, where $b' \leq j_c$ (part II of Figure 12).

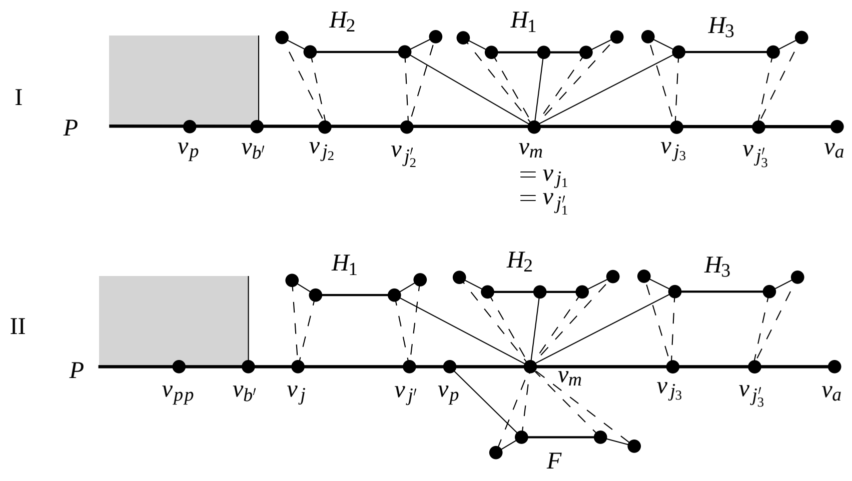

Figure 12: The two possible cases of the use of all H_i, $1 \leq i \leq 3$.

For each of these two cases, we have to check whether it is possible. Therefore, we compute two values and combine these.

Definition 5.7. Let *cl* and *fr* be variables, each having a boolean field *ok* and an integer field *min*, denoting the following.

- $cl.ok =$ true if and only if there is a partial nice path decomposition of (S^a_m, v_a) for some a, $m \leq a \leq n$, in which each partial one-path F connected to v_l, $l < m$, occurs on the left side of each partial one-path connected to v_m.

  If $cl.ok =$ true, then *clmin* denotes the smallest a for which this holds. Otherwise, $clmin = \infty$.

- $fr.ok =$ true if and only if there is a partial nice path decomposition of (S^a_m, v_a) for some a, $m \leq a \leq n$ in which

  - there is an i, $1 \leq i \leq nr$, H_i uses $[j, j']$ for some $j' \leq p$, and H_i has a vertex w for which $\{v_m, w\} \notin E_2$, and
  - there is a partial one-path F connected to v_p which uses $[m, m]$, and either F is the only partial one-path connected to v_p, or F has a vertex w' for which $\{v_p, w'\} \notin E_2$.

  If $fr.ok =$ true, then *frmin* denotes the smallest a for which this holds. Otherwise, $frmin = \infty$.

From the discussion above and Lemma 5.7, it follows that

$$\begin{aligned} all[k].ok &= cl.ok \vee fr.ok \\ all[k]min &= \min\{clmin, frmin\}. \end{aligned}$$

We now show how cl and fr can be computed. First consider cl.

Computation of cl

We first analyze the case that $cl.ok = \text{true}$.

Lemma 5.12. *Suppose $cl.ok$ and let $PD = (V_1, \dots, V_t)$ be a partial nice path decomposition of (S_m^{clmin}, v_{clmin}) in which no partial one-path connected to v_p occurs on the right side of a partial one-path connected to v_m. Then $all[k \Leftrightarrow 1].ok = \text{true}$, and there is a partial nice path decomposition PD' of (S_m^{clmin}, v_{clmin}) in which*

1. *no partial one-path connected to v_p occurs on the right side of a partial one-path connected to v_m,*
2. *$PD'[V_p^{all[k-1]min}]$ is a partial nice path decomposition of $(S_p^{all[k-1]min}, v_{all[k-1]min})$,*
3. *for each i, if for each $w \in V(H_i)$, $\{w, v_m\} \in E_2$, then H_i uses $[m, m]$,*
4. *for each i, if H_i uses $[j, j']$, then $j \geq all[k \Leftrightarrow 1]min$, and*
5. *there is an i, such that H_i uses $[j, clmin]$, $m \leq j \leq clmin$.*

Proof. According to Lemma 5.7, we may assume that condition 3 holds for PD, otherwise, we first transform PD such that 3 holds.

We next show that condition 5 holds already for PD. Let a, $1 \leq a \leq nr$, be such that no partial one-path H_i, $1 \leq i \leq nr$, occurs on the right side of H_a. Suppose H_a uses $[j, j']$. Then $m \leq j \leq j' \leq clmin$. Suppose H_a occurs in $(V_s, \dots, V_{s'})$. Then $v_{j'} \in V_{s'}$, and $(V_{s'+1}, \dots, V_t)$ contains only edges between vertices $\{v_{j'}, \dots, v_{clmin}\} \cup \{$ sticks of $v_{j'}, \dots, v_{clmin}\}$. Hence $(V_1, \dots, V_{s'})$ restricted to $V_m^{j'}$ is a partial nice path decomposition of $(S_m^{j'}, v_{j'})$ with the same properties as PD. This means that $clmin = j'$, and hence condition 5 holds.

Let V_r be the rightmost node of PD containing an edge of (S_p^p, v_p). If $v_p \in V_r$, then $(V_1, \dots, V_r)$ restricted to (S_p^p, v_p) is a partial nice path decomposition of (S_p^p, v_p). Hence $all[k \Leftrightarrow 1].ok = \text{true}$ and $all[k \Leftrightarrow 1]min = p$. Let $l = p$.

If $v_p \notin V_r$, then there is exactly one l, $p < l \leq m$, such that $v_l \in V_r$. It can be seen that in this case, $(V_1, \dots, V_r)$ restricted to V_p^l is a partial nice path decomposition of (S_p^l, v_l). Hence $all[k \Leftrightarrow 1].ok = \text{true}$ and $all[k \Leftrightarrow 1]min \leq l$.

We now construct a partial nice path decomposition PD' of (S_m^{clmin}, v_{clmin}) which satisfies conditions 1 – 5. Let $PD_2 = (V_{r+1}, \dots, V_t)$, and remove all occurrences of vertices of V_p^l and sticks of v_l from PD_2. Let PD_1 be a partial nice path-decomposition of $(S_p^{all[k-1]min}, v_{all[k-1]min})$. Let PD_3 be a path decomposition of width one of S',

$$S' = S[\{v_{all[k-1]min}, \dots, v_l\} \cup \{ \text{ sticks of } v_{all[k-1]min}, \dots, v_l\}],$$

with vertex $v_{all[k-1]min}$ in the leftmost node and vertex v_l in the rightmost node. Note that this is possible, since S' consists of path from $v_{all[k-1]min}$ to v_l with sticks. Let $PD' = PD_1 +\!\!+ PD_2 +\!\!+ PD_3$. It is easy to see that PD' is a partial nice path decomposition of (S_m^{clmin}, v_{clmin}) which satisfies conditions 1 – 5. □

The lemma implies that, if $all[k \Leftrightarrow 1].ok = \mathsf{false}$, then $cl.ok = \mathsf{false}$ and we do not have to compute anything. Suppose that $all[k \Leftrightarrow 1].ok = \mathsf{true}$, and let $min = all[k \Leftrightarrow 1]min$.

In order to compute cl, we compute the smallest value of a, $m \le a \le n$, for which there is a partial nice path decomposition PD of (S_m^a, v_a) in which

Condition 1. $PD[V_p^{min}]$ is a partial nice path decomposition of (S_p^{min}, v_{min}),

Condition 2. for each i, $1 \le i \le nr$, if each vertex w of H_i has $\{w, v_m\} \in E_2$, then H_i uses $[m,m]$, otherwise, H_i' uses $[j, j']$ for some $min \le j \le j' \le m$ or $m \le j \le j' = a$.

If this value for a exists, then $cl.ok = \mathsf{true}$ and $clmin = a$, otherwise $cl.ok = \mathsf{false}$.

Let $H_1', \ldots, H_{nr'}'$ denote the partial one-paths connected to v_m which have a vertex w for which $\{v_m, w\} \notin E_2$. Note that if $nr' > 2$, then $cl.ok = \mathsf{false}$. Suppose $nr' \le 2$. We distinguish between the cases that $nr' = 0$, $nr' = 1$ and $nr' = 2$.

The case that $nr' = 0$**.** If $nr' = 0$, then we can easily make a partial nice path decomposition of (S_m^m, v_m) from a partial nice path decomposition of (S_p^{min}, v_{min}) (see the proof of Lemma 5.12). Hence $cl.ok = \mathsf{true}$ and $clmin = m$.

The case that $nr' = 1$**.** Suppose $nr' = 1$. Suppose $cl.ok = \mathsf{true}$, and let PD be a partial nice path decomposition of (S_m^{clmin}, v_{clmin}) for which conditions 1 and 2 hold. Suppose H_1' uses $[j, j']$. There are two possibilities: either $min \le j \le j' \le m$ and $clmin = m$ or $m \le j \le j' \le n$ and $clmin = j'$.

For finding the value of cl as described above, we do the following. First we check whether H_1' can use $[j, j']$ for some $min \le j' \le j \le m$, i.e. if we can extend a partial nice path decomposition of (S_p^{min}, v_{min}) into a partial nice path decomposition of (S_m^m, v_m). If so, we make $cl.ok = \mathsf{true}$ and $clmin = m$. If not, then we find the smallest j', $m < j' \le n$, for which H_1' can use $[j, j']$ for some $m \le j \le j'$, i.e. for which we can extend a partial nice path decomposition of (S_p^{min}, v_{min}) into a partial nice path decomposition of $(S_m^{j'}, v_{j'})$ in which H_1' uses $[j, j']$. If we can find such a j', then we make $cl.ok = \mathsf{true}$ and $clmin = j'$. Otherwise, we make $cl.ok = \mathsf{false}$.

We now first show how to check whether H_1' can use $[j, j']$ for some $min \le j \le j' \le m$.

Let $P' \in P_1(H_1')$, let u and w be the two end points of P'. Let $v \in V(H_1')$ such that $\{v, v_m\} \in E_1$, and let V' be the subset of V containing all vertices of H_1', all vertices $v_{min}, \ldots, v_m$, and all sticks connected to $v_{min+1}, \ldots, v_{m-1}$. Let dum denote a dummy vertex, and let S_u denote the sandwich graph with

$$\begin{aligned} V(S_u) &= V' \cup \{\mathsf{dum}\} \\ E_1(S_u) &= E_1(S[V']) \cup \{\{\mathsf{dum}, u\}, \{\mathsf{dum}, v_{min}\}\} \\ E_2(S_u) &= E_2(S[V']) \cup \{\{\mathsf{dum}, v\} \mid v \in V'\}. \end{aligned}$$

If $v \neq w$ and v is not a stick of w, then additionally add edge $\{w, v_m\}$ to $E_1(S_u)$. Note that it may be the case that $\{w, v_m\} \notin E_2(S_u)$. Therefore, we call S_u an almost-sandwich graph. See also Figure 13: part I shows $G_1(S_u)$ for the case that $v \neq w$ and v is not a stick of w, and part II shows $G_1(S_u)$ for the other case. Now the almost-sandwich graph S_u is an almost-sandwich block with sticks.

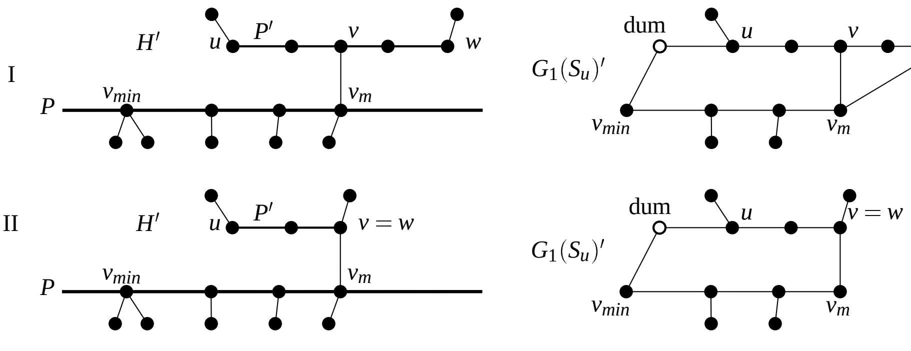

Figure 13: Example of the almost-sandwich graph S_u.

Define S_w in the same way, but with the roles of u and w exchanged.

Lemma 5.13. *There are j and j', $min \leq j \leq j' \leq m$, such that H'_1 can use $[j, j']$ if and only if one of the following holds.*

a. *S_u is a sandwich graph and there is a path decomposition of width two of S_u with edge $\{\mathsf{dum}, v_{min}\}$ in the leftmost node and vertex v_m in the rightmost node.*

b. *S_w is a sandwich graph and there is a path decomposition of width two of S_w with edge $\{\mathsf{dum}, v_{min}\}$ in the leftmost node and vertex v_m in the rightmost node.*

Proof. For the 'if' part, suppose condition a holds. Let PD be a partial nice path decomposition of (S_p^{min}, v_{min}). Let PD' be a path decomposition of width two of S_u with $\{\mathsf{dum}, v_{min}\}$ in the leftmost node and v_m in the rightmost node. Remove all occurrences of dum from PD', and on the left side, add a node $\{v_{min}, x\}$ for each stick x of v_{min}. Let PD'' be a path decomposition of width one of all partial one-paths except H'_1 that are connected to v_m, and add vertex v_m to each node of PD''. Now $PD +\!\!+ PD' +\!\!+ PD''$ is a partial nice path decomposition of S_m^m in which H'_1 uses $[j, j']$ for some $min \leq j \leq j' \leq m$.

For the 'only if' part, let $PD = (V_1, \ldots, V_t)$ be a partial nice path decomposition of (S_m^m, v_m) in which $PD[V_p^{min}]$ is a partial nice path decomposition of (S_p^{min}, v_{min}) and H'_1 uses $[j, j']$ for some $min \leq j \leq j' \leq m$. Suppose H'_1 occurs in $(V_s, \ldots, V_{s'})$. Suppose w.l.o.g. that there is a node V_a with $V_a = \{v_m\}$ and $a > j'$ (Lemma 5.4). Note that $j < m$, since H'_1 has a vertex x for which $\{v_m, x\} \notin E_2$. Note also that either V_s or $V_{s'}$ contains u and a stick of u, and either $V_{s'}$ or V_s contains w and a stick of w. Suppose w.l.o.g. that $u \in V_s$ and $w \in V_{s'}$. Let $u', w' \in V(H')$ such that $u' \in V_s$, $w' \in V_{s'}$, and u' is a stick adjacent to u, w' is a stick adjacent to w. For an example, see Figure 14. Part I shows again the case that $v \neq w$ and v not a stick of w, and part II the other case.

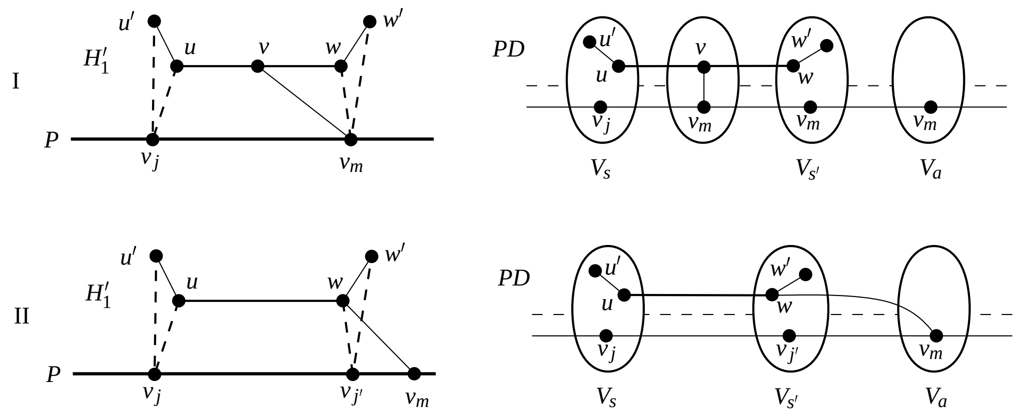

Figure 14: Examples for the proof of Lemma 5.13.

Consider the sequence $PD' = (V_s, \dots, V_a)$. Note that $v_m \in V_a$ and $V_s = \{u, u', v_j\}$. Let W be the subset of V containing all vertices of H'_1, vertices $v_j, \dots, v_m$ and sticks of vertices $v_{j+1}, \dots, v_{m-1}$. Remove all vertices in $V \Leftrightarrow W$ from PD'. Note that all vertices of W occur within PD', and all edges in E_1 between vertices of W occur within PD'. Hence PD' is a path decomposition of width two of $S[W]$.

Remember that v is the vertex of H'_1 for which $\{v, v_m\} \in E(S)$. If $v \neq w$ and v is not a stick of w, then $j' = m$, since v occurs only on the left side of $V_{s'}$, and there is a V_i, $s \leq i < s'$, with $v \in V_i$ and $v_m \in V_i$ (Lemma 3.3.5 of de Fluiter [1997]). Hence $\{w, v_m\} \subseteq V_{s'}$, which means that $\{w, v_m\} \in E_2$ and S_u is a sandwich graph.

Let V'' be the set of vertices $v_{min}, \dots, v_j$ and all sticks of $v_{min+1}, \dots, v_j$. Make a path decomposition PD'' of width one of $S[V'']$ with vertex v_{min} in the leftmost node and vertex v_j in the rightmost node. Add vertex dum to each node of PD''. Now $PD'' +\!\!+ PD'$ is a path decomposition of width two of S_u with edge $\{v_{min}, \mathsf{dum}\}$ in the leftmost node and vertex v_m in the rightmost node. This completes the proof. □

Lemma 5.14. *It takes $O(N^2)$ time to check whether H'_1 can use $[j, j']$ for some j and j', $min \leq j \leq j' \leq m$, where N is the number of vertices of S_u or S_w.*

Proof. S_u and S_w are almost-sandwich blocks with sticks, and hence the lemma follows from Lemma 4.6. □

We now show how to find the smallest j', $m < j' \leq n$, for which H'_1 can use $[j, j']$ for some $m \leq j \leq j'$. This is very similar to the previous computation.

Again, let $P' \in P_1(H'_1)$, let u and w be the two end points of P'. Let $v \in V(H'_1)$ such that $\{v, v_m\} \in E_1$. But now, let $V' \subseteq V(S)$ contain all vertices of H'_1, all vertices $v_m, \dots, v_n$, and all sticks connected to $v_{m+1}, \dots, v_{n-1}$. Let $S_u = S[V']$. If $v \neq w$ and v is not a stick of w, then additionally add edge $\{w, v_m\}$ to $E_1(S_u)$. See also Figure 15. (Note that again, it may be the case that $\{w, v_m\} \notin E_2(S_u)$.) Now the almost-sandwich graph S_u is an almost-sandwich block with sticks and loose ends u and v_n.

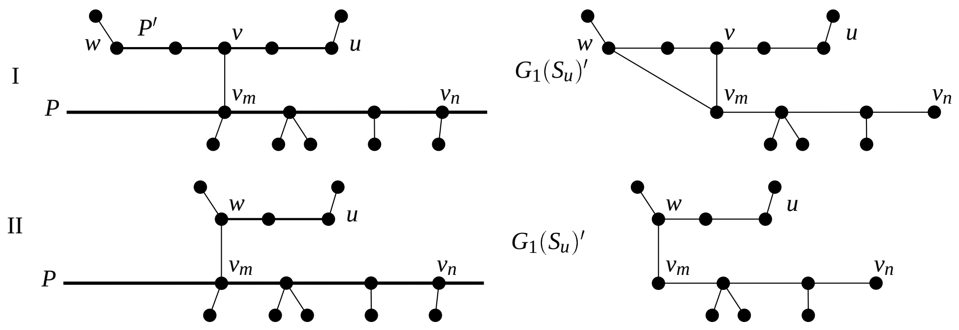

Figure 15: Example of S_u.

Define S_w in the same way, but with the roles of u and w exchanged. The following lemma resembles Lemma 5.13

Lemma 5.15. *Let j' be an integer, $m \le j' \le n$. There is a j, $m \le j \le j'$, such that H'_1 can use $[j, j']$ if and only if one of the following holds.*

a. *$S'_u = S_u[V' \Leftrightarrow \{v_{j'+1}, \dots, v_n\} \Leftrightarrow \{$ sticks of $v_{j'}, \dots, v_n\}]$ is a sandwich graph and there is a path decomposition of width two of S'_u with v_m in the leftmost node and $v_{j'}$, u and a stick u' of u in the rightmost node.*

b. *$S'_w = S_w[V' \Leftrightarrow \{v_{j'+1}, \dots, v_n\} \Leftrightarrow \{$ sticks of $v_{j'}, \dots, v_n\}]$ is a sandwich graph and there is a path decomposition of width two of S'_w with v_m in the leftmost node and $v_{j'}$, w and a stick w' of w in the rightmost node.*

Proof. For the 'if' part, suppose condition a holds. Let PD be a partial nice path decomposition of (S^m_p, v_m). Such a partial nice path decomposition exists since $all[k \Leftrightarrow 1].ok =$ true and $min \le m$. For each H_i, $H_i \ne H'_1$, make a path decomposition of width one of $S[V(H_i)]$, and add vertex v_m to each node. Add all these path decompositions on the right side of PD. Furthermore, add a node $\{v_m, x\}$ for each stick x of v_m on the right side of PD.

Let PD' be a path decomposition of width two of S'_u with v_m in the leftmost node and u and $v_{j'}$ in the rightmost node. Now $PD + \!\!+ PD'$ is a partial nice path decomposition of $(S^{j'}_m, v_{j'})$, in which H'_1 uses $[j, j']$ for some $m \le j \le j'$.

The 'only if' part can be proved in almost the same way as the 'only if' part of Lemma 5.13. □

Lemma 5.16. *It takes $O(N^2)$ time to check whether H'_1 can use $[j, j']$ for some $m \le j \le j' \le n$ and to find the smallest j' for which this holds. (N is the number of vertices of S_u or S_w.)*

Proof. S_u and S_w are almost-sandwich blocks with sticks and loose ends. By Corollary 4.2, we can find the smallest j, $m \le j \le n$, for which there is a path decomposition PD of width two of S'_u as defined in condition a of Lemma 5.15 with v_m in the leftmost node and $v_{j'}$ and u in the rightmost node. Since j is minimal, the rightmost node of PD also contains a stick of u. This means that we can find the smallest j for which condition a holds in $O(N^2)$ time. The same holds for condition b. □

This completes the description of the computation of cl for the case that $nr' = 1$.

The case that $nr' = 2$. Suppose $nr' = 2$, i.e. there are two partial one-paths H'_1 and H'_2 connected to v_m which have a vertex w for which $\{v_m, w\} \notin E_2$. Remember that $clmin$ is the smallest value of a, $m \leq a \leq n$, for which there is a partial nice path decomposition PD of (S^a_m, v_a), which satisfies conditions 1 and 2 as described on page 46. If there is such an a, then $cl.ok = \mathsf{true}$ and $clmin = a$, otherwise, $cl.ok = \mathsf{false}$.

Suppose $cl.ok = \mathsf{true}$, and PD is a partial nice path decomposition of (S^{clmin}_m, v_{clmin}) for which conditions 1 and 2 on page 46 hold. Suppose H'_1 uses $[j_1, j'_1]$ and H'_2 uses $[j_2, j'_2]$. There are two possibilities: either $min \leq j_1 \leq j'_1 \leq m$, $m \leq j_2 \leq j'_2 \leq n$ and $clmin = j'_2$, or $min \leq j_2 \leq j'_2 \leq m$, $m \leq j_1 \leq j'_1 \leq n$ and $clmin = j'_1$.

For finding cl as described above, we do the following. First we check

- whether H'_1 can use $[j_1, j'_1]$ for some $min \leq j_1 \leq j_1 \leq m$, i.e. we can extend a partial nice path decomposition of (S^{min}_p, v_{min}) into a partial nice path decomposition of $(S^m_m \Leftrightarrow H'_2, v_m)$ in which H'_1 uses $[j_1, j'_1]$ for some $min \leq j_1 \leq j'_1 \leq m$, and
- whether H'_2 can use $[j_2, j'_2]$ for some $m \leq j_2 \leq j'_2 \leq n$, i.e. we can extend a partial nice path decomposition of $(S^m_m \Leftrightarrow H'_2, v_m)$ into a partial nice path decomposition of $(S^{j'_2}_m, v_{j'_2})$, and we find the smallest j'_2 for which this holds ($j'_2 = \infty$ if it does not hold).

Then we check whether H'_2 can use $[l_2, l'_2]$ for some $min \leq l_2 \leq l'_2 \leq m$ and H'_1 can use $[l_1, l'_1]$ for some $m \leq l_1 \leq l'_1 \leq n$ and we find the smallest l'_1 for which this holds ($l'_1 = \infty$ if it does not hold). If one of them is possible, then we make $cl.ok = \mathsf{true}$ and $clmin = \min\{j'_2, l'_1\}$. Otherwise, we make $cl.ok = \mathsf{false}$.

The algorithm to checking whether H'_i ($i = 1, 2$) can use $[j, j']$ for some $min \leq j \leq j' \leq m$ is described above, for the case that $nr' = 1$ (pages 46 – 48). The algorithm for computing the smallest l', $m \leq l' \leq n$, for which there is an l, $m \leq l \leq l'$, such that H'_i ($i = 1, 2$) can use $[l, l']$ is also described above for the case that $nr' = 1$ (pages 48 – 49). Both algorithms take $O(N^2)$ time, where N denotes the number of vertices in H'_1, H'_2, $v_{min}, \ldots, v_n$ and all sticks of $v_{min+1}, \cdots, v_{n-1}$.

This completes the description of the computation of cl for the case that there are two or more partial one-paths connected to v_m. We conclude with the following corollary.

Corollary 5.4. *If there are two or more partial one-paths connected to v_m, then it takes $O(N^2)$ time to compute cl, where N denotes the number of vertices $v_p, \ldots, v_n$, all sticks of $v_{p+1}, \ldots, v_{n-1}$, and vertices of all partial one-paths connected to v_m.*

Computation of fr

We assume that $k > 1$, otherwise, $ft.ok = \mathsf{false}$.

Let $H'_1, \ldots, H'_{nr'}$ denote the partial one-paths connected to v_m which contain a vertex w with $\{w, v_m\} \notin E_2$. Let $F_1, \ldots, F_c$ denote the partial one-paths connected to v_p. Let $F'_1, \ldots, F'_{c'}$ denote the partial one-paths connected to v_p which contain a vertex w for which $\{w, v_p\} \notin E_2$.

Note that if $nr' > 2$, then $fr.ok = \mathsf{false}$. If $nr' = 0$, then, by definition of fr, $fr.ok = \mathsf{false}$. Similarly, if $c' > 2$, or if $c \geq 2$ and $c' = 0$, we make $fr.ok = \mathsf{false}$.

Suppose $1 \leq nr' \leq 2$, $1 \leq c' \leq 2$, and either $c' > 0$ or $c = 1$. We distinguish between two cases, namely the case that $c = 1$ and the case that $c > 1$.

The case that $c = 1$. We first analyze the case that $fr.ok = \text{true}$.

Lemma 5.17. *Suppose $fr.ok$ and let PD be a partial nice path decomposition of (S_m^{frmin}, v_{frmin}) in which there is a partial one-path H'_a, $1 \leq a \leq 2$, which occurs on the left side of partial one-path F_1. Then $all[k \Leftrightarrow 2].ok = \text{true}$, and there is a partial nice path decomposition of (S_m^{frmin}, v_{frmin}) in which*

1. *$PD[V_{pp}^{all[k-2]min}]$ is a partial nice path decomposition of $(S_{pp}^{all[k-2]min}, v_{all[k-2]min})$,*
2. *H'_a uses $[j, j']$, where $all[k \Leftrightarrow 2]min \leq j \leq j' \leq p$,*
3. *F_1 uses $[m, m]$,*
4. *for each partial one-path H_i, $1 \leq i \leq nr$, $H_i \notin \{H'_1, \dots, H'_{nr'}\}$, H_i uses $[m, m]$, and*
5. *if $nr' = 2$, then H'_{3-a} uses $[l, frmin]$, for some $m \leq l \leq frmin$.*

Proof. By Lemma 5.7, we may assume that condition 4 holds for PD. Furthermore, we may assume that condition 5 holds: if $nr' = 2$ and H'_{3-a} uses $[l, l']$ for some $m \leq l \leq l' < frmin$, then we can prove that $frmin$ is not minimal (see also proof of Lemma 5.12).

Let V_r be the rightmost node of PD' containing an edge of (S_{pp}^{pp}, v_{pp}). If $v_{pp} \in V_r$, then $(V_1, \dots, V_r)$ restricted to V_{pp}^{pp} is a partial nice path decomposition of (S_{pp}^{pp}, v_{pp}). Hence $all[k \Leftrightarrow 2].ok = \text{true}$ and $all[k \Leftrightarrow 2]min \leq pp$. Let $l = pp$.

If $v_{pp} \notin V_r$, then there is an l, $pp < l \leq p$, such that $v_l \in V_r$. It can be seen that in this case, $(V_1, \dots, V_r)$ restricted to V_{pp}^l is a partial nice path decomposition of (S_{pp}^l, v_l). Hence $all[k \Leftrightarrow 2].ok = \text{true}$ and $all[k \Leftrightarrow 2]min \leq l$.

We now construct a partial nice path decomposition of (S_m^{frmin}, v_{frmin}) for which conditions 1 – 5 hold. Let $PD_3 = (V_{r+1}, \dots, V_t)$, and remove all occurrences of vertices from (S_{pp}^l, v_l) from PD_3. Let PD_1 be a partial nice path decomposition of $(S_{pp}^{all[k-2]min}, v_{all[k-2]min})$. Let PD_2 be a path decomposition of width one of $S[\{v_{all[k-2]min}, \dots, v_l\} \cup \{ \text{ sticks of } v_{all[k-2]min}, \dots, v_l\}]$ with $v_{all[k-2]min}$ in the leftmost node and v_l in the rightmost node.

let $PD' = PD_1 + PD_2 + PD_3$. It is easy to see that PD' is a partial nice path decomposition of (S_m^{frmin}, v_{frmin}) which satisfies conditions 1 – 5. □

The lemma implies that, if $all[k \Leftrightarrow 2].ok = \text{false}$, then $fr.ok = \text{false}$ and we do not have to compute anything. Suppose that $all[k \Leftrightarrow 2].ok = \text{true}$, and let $min' = all[k \Leftrightarrow 2]min$.

We compute fr as follows. We let $frmin$ be smallest value, $m \leq frmin \leq n$, for which there is a partial nice path decomposition PD of (S_m^{clmin}, v_{clmin}) which satisfies conditions 1 – 5 of Lemma 5.17. If this value can be found, then $fr.ok = \text{true}$, otherwise, $fr.ok = \text{false}$.

Suppose $fr.ok = \text{true}$, and let PD be a partial nice path decomposition of (S_m^{frmin}, v_{frmin}) for which conditions 1 – 5 of Lemma 5.17 hold. For each i, $1 \leq i \leq nr'$, suppose H'_i uses $[j_i, j'_i]$. If $nr' = 1$, then $min' \leq j_1 \leq j'_1 \leq p$ and $clmin = m$. If $nr' = 2$, there are two possibilities: either

$min' \leq j_1 \leq j_1' \leq p$, $m \leq j_2 \leq j_2' \leq n$ and $frmin = j_2'$, or $min' \leq j_2 \leq j_2' \leq p$, $m \leq j_1 \leq j_1' \leq n$ and $frmin = j_1'$.

For finding the value of fr as described above, we do the following. If $nr' = 1$, then we check whether H_1' can use $[j, j']$ for some $min' \leq j \leq j' \leq p$ and at the same time F_1 can use $[m,m]$, i.e. whether we can extend a partial nice path decomposition of $(S_{pp}^{min'}, v_{min'})$ into a partial nice path decomposition of (S_m^m, v_m) in which H_1' uses $[j, j']$ for some $min' \leq j \leq j' \leq p$ and F_1 uses $[m,m]$. If so, we make $fr.ok = \mathsf{true}$ and $frmin = m$, otherwise, $fr.ok = \mathsf{false}$.

If $nr' = 2$, then for $i = 1,2$, we do the following.

a. We check whether H_i' can use $[j, j']$ for some $min' \leq j \leq j' \leq p$ and at the same time F_1 can use $[m,m]$, i.e. whether we can extend a partial nice path decomposition of $(S_{pp}^{min'}, v_{min'})$ into a partial nice path decomposition PD of $(S_m^m \Leftrightarrow H_{3-i}', v_m)$ in which H_i' uses $[j, j']$ for some $min' \leq j \leq j' \leq p$ and F_1 uses $[m,m]$.

b. We find the smallest l_i, $m \leq l_i \leq n$, for which H_{3-i}' can use $[l, l_i]$ for some $m \leq l \leq l_i$, i.e. for which we can extend a partial nice path decomposition of $(S_m^m \Leftrightarrow H_{3-i'}', v_m)$ into a partial nice path decomposition of $(S_m^{l_i}, v_{l_i})$ ($l_i = \infty$ if this is not possible).

If both a and b are not possible, then $l_i = \infty$. Now, if both l_1 and l_2 equal ∞, then $fr.ok = \mathsf{false}$. Otherwise, $fr.ok = \mathsf{true}$ and $frmin = \min\{l_1, l_2\}$.

In the case that $nr' = 2$, finding the smallest value of j', $m \leq j' \leq n$ for which a partial one-path H' connected to v_m can use $[j, j']$ for some $m \leq j \leq j'$ can be done in the way described for the computation of cl on pages 48 – 49. Therefore, we only describe how to check whether F_1 can use $[m,m]$ and H_1' can use $[j, j']$ for some $min' \leq j \leq j' \leq p$ at the same time. Note that this is only possible if both F_1 and H_1' are of type I, and if either $m = p+1$ or $m = p+2$ and v_{m+1} has no sticks (Lemma 5.11). So suppose this holds.

Let $P' \in P_1(H_1')$, let u be the end point of P' for which the path from v_m to u contains P'. Furthermore, let $P'' \in P_1(F_1)$ and let w be the end point of P'' for which the path from v_p to w contains P'' (see also part I of Figure 16). Let $V' \subseteq V(S)$ be the set containing all vertices of H_1' and of F_1, all vertices $v_{min'}, \ldots, v_m$, and all sticks connected to $v_{min'+1}, \ldots, v_{m-1}$. Let dum be a dummy vertex, and let S' denote the sandwich graph defined as follows.

$$\begin{aligned} V(S') &= V' \cup \{\mathsf{dum}\} \\ E_1(S') &= E_1(S[V']) \cup \{\{\mathsf{dum}, u\}, \{\mathsf{dum}, v_{min'}\}\} \\ E_2(S') &= E_2(S[V']) \cup \{\{\mathsf{dum}, v\} \mid v \in V'\}\}. \end{aligned}$$

See also Figure 16. The sandwich graph S' is a sandwich block with sticks and loose ends w and v_m (although loose end w is actually not ‘loose’).

Lemma 5.18. *H_1' can use $[j, j']$ for some $min' \leq j \leq j' \leq p$ and F_1 can use $[m,m]$ at the same time if and only if there is a path decomposition of width two of S' with edge $\{\mathsf{dum}, v_{min'}\}$ in the leftmost node and v_m and w in the rightmost node.*

Proof. For the ‘if’ part, we can easily combine a partial nice path decomposition of $(S_{pp}^{min'}, v_{min'})$ and a path decomposition of width two of S' with edge $\{\{\mathsf{dum}, v_{min'}\}$ in the leftmost node and

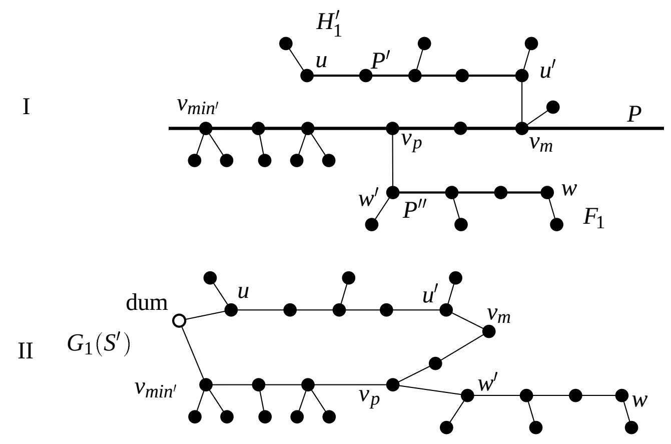

Figure 16: Example of S'.

with w and v_m in the rightmost node into a nice partial path decomposition of (S^m_m, v_m) if $nr' = 1$ and of $(S^m_m \Leftrightarrow H'_2, v_m)$ if $nr' = 2$.

For the 'only if' part, let $PD = (V_1, \ldots, V_t)$ be a partial nice path decomposition of (S^m_m, v_m) if $nr' = 1$ and of $(S^m_m \Leftrightarrow H'_2, v_m)$ if $nr' = 2$, in which $PD[V^{min'}_{pp}]$ is a partial nice path decomposition of $(S^{min'}_{pp}, v_{min'})$ and H'_1 uses $[j, j']$ for some $min' \leq j \leq j' \leq p$ and F_1 uses $[m,m]$, and all H_i, $H_i \notin \{H'_1, H'_2\}$, use $[m,m]$.

Suppose H'_1 occurs in $(V_r, \ldots, V_{r'})$ and F_1 occurs in $(V_s, \ldots, V_{s'})$. By Lemma 5.10, $r' \leq s$. Furthermore, by (the proof of) Lemma 5.11, $u \in V_r$ and $w \in V_{s'}$, and $v_j \in V_r$ and $v_m \in V_{s'}$. Consider the subsequence $PD' = (V_r, \ldots, V_{s'})$. Note that all vertices of H'_1 and F_1 and all vertices $v_j, \ldots, v_m$ and all sticks adjacent to $v_{j+1}, \ldots, v_{m-1}$ occur in PD'. Also, all edges between these vertices occur in PD'. Remove all occurrences of other vertices from PD'.

We transform PD' into a path decomposition of width two of S' with $\{\mathsf{dum}, v_{min'}\}$ in the leftmost node with v_m and w in the rightmost node. On the left side of PD', add a node $\{\mathsf{dum}, u, v_j\}$. Furthermore, make a path decomposition PD'' of the sub-sandwich graph of S induced by the vertices $v_{min'}, \ldots, v_j$ and the sticks adjacent to $v_{min'+1}, \ldots, v_j$ with vertex $v_{min'}$ in the leftmost node and vertex v_j in the rightmost node. Add vertex dum to each node of PD''. Now $PD'' \mathbin{+\!\!+} PD'$ is a path decomposition of width two of S' with the desired vertices in the leftmost and rightmost nodes. □

Lemma 5.19. *It takes $O(N^2)$ time to check whether H'_1 can use $[j, j']$ for some j and j', $min' \leq j \leq j' \leq p$ and F_1 can use $[m,m']$ at the same time (where N is the number of vertices of S').*

Proof. S' is a sandwich block with sticks and loose ends, and hence Corollary 4.2 implies the lemma. □

The case that $c > 1$. Suppose $c > 1$, i.e. there are $c > 1$ partial one-paths $F_1, \ldots, F_c$ connected to v_p. Remember that we assume that $1 \leq c' \leq 2$, otherwise $fr.ok = \mathsf{false}$.

We first analyze the case that $fr.ok = \mathsf{true}$.

Lemma 5.20. *Suppose $fr.ok$ and let PD be a partial nice path decomposition of (S_m^{frmin}, v_{frmin}) in which there are partial one-paths H'_a, $1 \leq a \leq nr'$ and F'_b, $1 \leq b \leq c'$, such that H'_a occurs on the left side of F'_b. Then $allbo[k \Leftrightarrow 1].ok = \mathsf{true}$, and there is a partial nice path decomposition of (S_m^{frmin}, v_{frmin}) in which*

1. *there is an $F' \in allbo[k \Leftrightarrow 1].tr$ such that*
   (a) *$PD[V_p^p \Leftrightarrow F']$ is a partial nice path decomposition of $(S_p^p \Leftrightarrow F', v_p)$, and*
   (b) *F' uses $[m,m]$,*
2. *H'_a uses $[p,p]$,*
3. *for each H_i, $1 \leq i \leq nr$ and $H_i \notin \{H'_1, \dots, H'_{c'}\}$, H'_i uses $[m,m]$, and*
4. *if $nr' = 2$, then H'_{3-a} uses $[l, frmin]$ for some $m \leq l \leq frmin$.*

Proof. We may assume that condition 4 holds for PD (see also proof of Lemma 5.17). By Corollaries 5.2 and 5.3 and Lemma 5.11, PD already is a partial nice path decomposition satisfying conditions 1 – 4. □

The lemma implies that, if $allbo[k \Leftrightarrow 1].ok = \mathsf{false}$, then $fr.ok = \mathsf{false}$ and we do not have to compute anything. Suppose that $allbo[k \Leftrightarrow 1].ok = \mathsf{true}$.

Lemma 5.20 shows that we can compute fr as follows. We let $frmin$ be the smallest value, $m \leq frmin \leq n$, for which there is a partial nice path decomposition PD of (S_m^{frmin}, v_{frmin}) which satisfies conditions 1 – 4 of Lemma 5.20. If this value can be found, then $fr.ok = \mathsf{true}$, otherwise $fr.ok = \mathsf{false}$.

Suppose $fr.ok = \mathsf{true}$, and let PD be a partial nice path decomposition of (S_m^{frmin}, v_{frmin}) for which conditions 1 – 4 hold. For each i, $1 \leq i \leq nr'$, suppose H'_i uses $[j_i, j'_i]$, and for each $F_a \in allbo[k \Leftrightarrow 1].tr$, suppose F_a uses $[l_a, l'_a]$. If $nr' = 1$, then $j_1 = j'_1 = p$. If $nr' = 2$, there are two possibilities for H'_1 and H'_2: either $j_1 = j'_1 = p$, $m \leq j_2 \leq j'_2 \leq frmin$, or vice versa. Similarly, if there is one $F_a \in allbo[k \Leftrightarrow 2].tr$, then $l_a = l'_a = m$, but if there are two $F_a, F_b \in allbo[k \Leftrightarrow 2].tr$, then either $l_a = l'_a = m$, or $l_b = l'_b = m$.

For finding the value of fr as described above, we do the following.

If $nr' = 1$ then we check whether H'_1 can use $[p,p]$ and at the same time there is an $F' \in allbo[k \Leftrightarrow 1].tr$ which can use $[m,m]$, i.e. whether there is an $F' \in allbo[k \Leftrightarrow 1].tr$ for which we can extend a partial nice path decomposition of $(S_p^p \Leftrightarrow F', v_p)$ into a partial nice path decomposition of (S_m^m, v_m) in which H'_1 uses $[p,p]$ and F' uses $[m,m]$. If so, then $fr.ok = \mathsf{true}$ and $frmin = m$.

If $nr' = 2$ then for $i = 1, 2$, we check whether H'_i can use $[p,p]$ and, at the same time, there is an $F' \in allbo[k \Leftrightarrow 1].tr$ which can use $[m,m]$, and if so, we find the smallest j_i, $m \leq j_i \leq n$, for which H'_{3-i} can use $[j, j_i]$ for some $m \leq j \leq j_i$, i.e. for which we can extend a partial nice path decomposition of $(S_m^m \Leftrightarrow H'_{3-i}, v_m)$ into a partial nice path decomposition of $(S_m^{j_i}, v_{j_i})$ ($j_i = \infty$ if this is not possible). If both possibilities do not work, then $fr.ok = \mathsf{false}$. Otherwise, $fr.ok = \mathsf{true}$ and $frmin = \min\{j_1, j_2\}$.

In the case that $nr' = 2$, finding the smallest value of j', $m \leq j' \leq n$ for which a partial one-path H' connected to v_m can use $[j, j']$ for some $m \leq j \leq j'$ can be done in the way described for the computation of cl on pages 48 – 49.

Checking whether F_1' can use $[m,m]$ and H_1' can use $[p, p]$ can be done in the same way as checking whether F_1' can use $[m,m']$ and H_1' can use $[j, j']$ for some $all[k \Leftrightarrow 2]min \leq j \leq j' \leq p$ (pages 52 – 53): use p instead of $all[k \Leftrightarrow 2]min$. The underlying graph of sandwich graph S' then looks as in Figure 17.

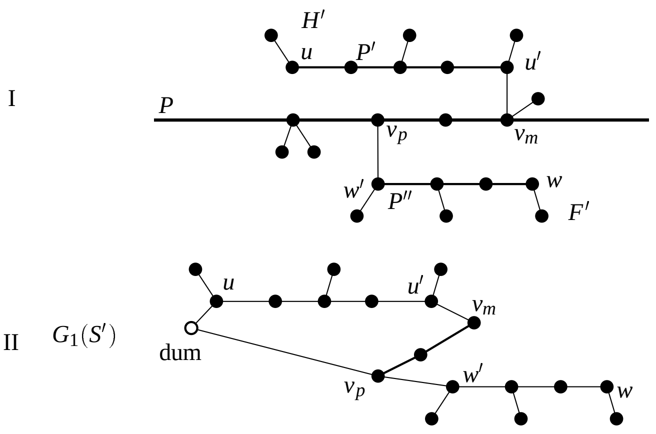

Figure 17: Example of S'.

All computations described above can be done in $O(N^2)$, where N is the number of vertices involved in the computation.

This completes the description of the case that $c' > 1$. We conclude with the following corollary.

Corollary 5.5. *It takes $O(N^2)$ time to compute fr, where N denotes the number of vertices $v_{pp}, \dots, v_n$, all sticks of $v_{pp+1}, \dots, v_{n-1}$, and vertices of all partial one-paths of v_p and v_m.*

This completes the description of the computation of $all[k]$ for the case that the number nr of partial one-paths connected to v_m is at least two.

Corollary 5.6. *It takes $O(N^2)$ time to compute $all[k]$ if $nr \geq 2$, where N denotes the number of vertices $v_{pp}, \dots, v_n$, all sticks of $v_{pp+1}, \dots, v_{n-1}$, and vertices of all partial one-paths connected to v_p and v_m.*

The Computation of $all[k]$ for the Case that $nr = 1$

We first analyze the possible cases if there is a partial nice path decomposition of (S_m^a, v_a) $(m \leq a \leq n)$. Suppose a is an integer, $m \leq a \leq n$ and PD' is a partial nice path decomposition of (S_m^a, v_a). Suppose that H_1 uses $[j, j']$. By Corollaries 5.1 – 5.3 and Lemma 5.11 there are two possibilities (see also Figure 18):

1. all partial one-paths connected to some v_i, $1 \leq i < m$, occur on the left side of H_1, or

2. $p < m$, there is one partial one-path F connected to v_p such that F occurs on the right side of H_1, and all partial one-paths $F' \neq F$ connected to some v_i, $i < m$, occur on the left side of H_1.

In the first case, $p \leq j \leq j' \leq n$, and for all partial one-paths F' connected to v_i, $i < m$, if F' uses $[b,b']$, then $b' \leq j$ (part I of Figure 18). In the second case, $pp \leq j \leq j' \leq p$, F uses $[l,l']$ for some $m \leq l \leq l' \leq n$, and all partial one-paths F' connected to some v_i, $i < m$ and $F' \neq F$, use $[b,b']$ for some $b' \leq j$ (part II of Figure 18).

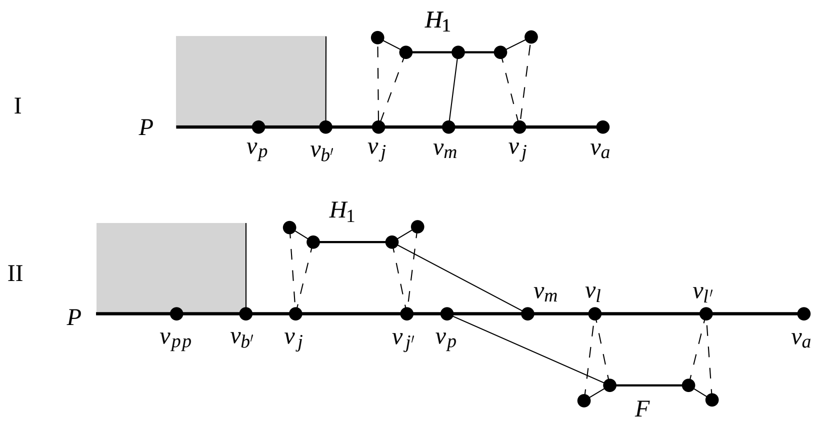

Figure 18: The two possible cases for the use of H_1.

Similarly to the case that $nr > 1$, for each of these two cases, we have to check whether it is possible. Therefore, we again compute two values and combine these.

Definition 5.8. Let cl and fr be variables, each having a boolean field ok and an integer field min, denoting the following.

- $cl.ok =$ true if and only if there is a partial nice path decomposition of (S_m^a, v_a) for some a, $m \leq a \leq n$, in which each partial one-path F connected to v_l, $l < m$, occurs on the left side of H_1.

  If $cl.ok =$ true, then $clmin$ denotes the smallest a for which this holds. Otherwise, $clmin = \infty$.

- $fr.ok =$ true if and only if there is a partial nice path decomposition of (S_m^a, v_a) for some a, $m \leq a \leq n$ in which

  - H_1 uses $[j, j']$ for some $j' \leq p$, and
  - there is a partial one-path F connected to v_p which uses $[m,m]$, and either F is the only partial one-path connected to v_p, or F has a vertex w' for which $\{v_p, w'\} \notin E_2$.

  If $fr.ok =$ true, then $frmin$ denotes the smallest a for which this holds. Otherwise, $frmin = \infty$.

From the discussion above and Lemma 5.7, it follows that

$$all[k].ok = cl.ok \vee fr.ok, \text{ and}$$
$$all[k]min = \min\{clmin, frmin\}.$$

We now show how cl and fr can be computed. First consider cl.

Computation of cl

We first analyze the case that $cl.ok = \mathsf{true}$.

Lemma 5.21. *Suppose $cl.ok$ and PD is a partial nice path decomposition of (S_m^{clmin}, v_{clmin}) in which no partial one-path connected to v_i, $i < m$, occurs on the right side of H_1. Then $all[k \Leftrightarrow 1].ok = \mathsf{true}$, and there is a partial nice path decomposition PD' of (S_m^{clmin}, v_{clmin}) in which*

1. *no partial one-path connected to v_p occurs on the right side of H_1,*
2. *$PD'[V_p^{all[k-1]min}]$ is a partial nice path decomposition of $(S_p^{all[k-1]min}, v_{all[k-1]min})$,*
3. *H_1 uses $[j, j']$ for some $all[k \Leftrightarrow 1]min \le j \le j' \le clmin$, and*
4. *the rightmost node of PD contains an edge of S_m^m.*

Proof. We first show that condition 3 holds already for PD. Let $PD = (V_1, \dots, V_t)$, and let V_r, $1 \le r \le t$, be the rightmost node containing an edge of S_m^m. Let v_j, $m \le j \le clmin$, be such that either $j = m$ or $v_m \notin V_r$ and $v_j \in V_r$. Note that v_j is uniquely defined. Now it can be seen that all edges between vertices of $\{v_m, \dots, v_j\} \cup \{$ sticks of $v_m, \dots, v_{j-1}\}$ occur within $(V_1, \dots, V_r)$. Hence $(V_1, \dots, V_r)$ restricted to V_m^j is a partial nice path decomposition of (S_m^j, v_j) with the same properties as PD. This means that $j = clmin$ and $r = t$.

The remainder of the proof is similar to the proof of Lemma 5.12 $\square$

The lemma implies that, if $all[k \Leftrightarrow 1].ok = \mathsf{false}$, then $cl.ok = \mathsf{false}$ and we do not have to compute anything. Suppose that $all[k \Leftrightarrow 1].ok = \mathsf{true}$, and let $min = all[k \Leftrightarrow 1]min$.

In order to compute cl, we let $clmin$ be the smallest value, $m \le clmin \le n$, for which there is a nice partial path decomposition PD of (S_m^a, v_a) which satisfies conditions 1 – 4 of Lemma 5.21. If we can find this value, then $cl.ok = \mathsf{true}$, otherwise, $cl.ok = \mathsf{false}$.

We distinguish between two cases, namely the case that H_1 has type I and the case that H_1 has type II or III. We start with the latter one.

The case that H_1 has type II **or** III. Lemma 5.2 shows that in each nice path decomposition of width two of S with nice path P, if H_1 uses $[j, j']$ for some $p \le j \le j' \le n$, then $j \le m \le j'$, and hence this means that, if $cl.ok = \mathsf{true}$ and PD is a partial nice path decomposition of (S_m^{clmin}, v_{clmin}) satisfying conditions 1 – 4 or Lemma 5.21, then the rightmost node of PD contains an edge of H_1, and hence H_1 uses $[j, clmin]$, for some $m \le j \le clmin$.

For finding $clmin$ as described above, we find the smallest value of j', $m \le j' \le n$, for which H_1 can use $[j, j']$ for some $min \le j \le j'$, i.e. we for which we can extend a partial nice path decomposition of (S_p^{min}, v_{min}) into a nice partial nice path decomposition of $(S_m^{j'}, v_{j'})$ in which H_1 uses $[j, j']$. If there is such an j', we make $cl.ok = \text{true}$ and $clmin = j'$, otherwise, we make $cl.ok = \text{false}$. We now first show how to find this minimum value for j'.

Let $P' \in P_1(H_1)$, let u and w be the two end points of P'. Let $v \in V(H_1)$ such that $\{v, v_m\} \in E_1$. Let $V' \subseteq V(S)$ contain all vertices of H_1, all vertices $v_{min}, \dots, v_n$, and all sticks connected to $v_{min+1}, \dots, v_{n-1}$. Let S_u denote the sandwich graph with

$$\begin{aligned} V(S_u) &= V' \cup \{\text{dum}\} \\ E_1(S_u) &= E_1(S[V']) \cup \{\{\text{dum}, u\}, \{\text{dum}, v_{min}\}\} \\ E_2(S_u) &= E_2(S[V']) \cup \{\{\text{dum}, v\} \mid v \in V'\}. \end{aligned}$$

See Figure 19 for an example of the underlying graph of S_u for the case that H_1 has type III. The sandwich graph S_u is a sandwich block with sticks and loose ends w and v_n.

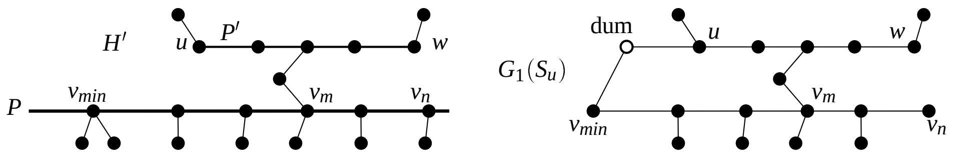

Figure 19: Example of $G_1(S_u)$.

Define S_w in the same way, but with the roles of u and w exchanged.

Lemma 5.22. *Let j' be an integer, $m \le j' \le n$. Then H_1' can use $[j, j']$ for some $min \le j \le m$ if and only if one of the following holds.*

a. There is a path decomposition of width two of

$$S_u[V(S_u) \Leftrightarrow \{v_{j'+1}, \dots, v_m\} \Leftrightarrow \{ \textit{ sticks of } v_{j'}, \dots, v_m\}]$$

with edge $\{\text{dum}, v_{min}\}$ in the leftmost node, and $v_{j'}$, w and a stick w' of w in the rightmost node.

b. There is a path decomposition of width two of

$$S_w[V(S_w) \Leftrightarrow \{v_{j'+1}, \dots, v_m\} \Leftrightarrow \{ \textit{ sticks of } v_{j'}, \dots, v_m\}]$$

with edge $\{\text{dum}, v_{min}\}$ in the leftmost node, and $v_{j'}$, u and a stick u' of u in the rightmost node.

Proof. The proof is similar to the proofs of Lemma 5.13 and Lemma 5.15. □

Lemma 5.23. *If H_1 has type* II *or* III*, then it takes $O(N^2)$ time to compute cl, where N is the number of vertices of S_u or S_w.*

Proof. S_u and S_w are sandwich blocks with sticks and loose ends, and hence the lemma follows from Corollary 4.2 and the proof of Lemma 5.16. □

The case that H_1 has type I. This case is a little more complicated, since if $cl.ok = \mathsf{true}$, then in a partial nice path decomposition of (S_m^{clmin}, v_{clmin}) satisfying conditions $1-4$ of Lemma 5.21, the rightmost node does not necessarily contain an edge of H_1, and hence it is possible that H_1 uses $[j, j']$, for some $j' < clmin$.

We compute the value of cl as follows. We compute the smallest a, $m \leq a \leq n$, for which we can extend a partial nice path decomposition of (S_p^{min}, v_{min}) into a partial nice path decomposition of (S_m^a, v_a) in which the rightmost node contains an edge of S_m^m (and hence H_1 uses $[j, j']$ for some $min \leq j \leq j' \leq a$). If there is no such a, then $cl.ok = \mathsf{false}$, otherwise, $cl.ok = \mathsf{true}$ and $clmin = a$.

Let $P' \in \mathcal{P}_1(H_1)$, let u and w be the end points of P', such that the path from u to v_m contains w. Furthermore, let w' be the neighbor of v_m in H_1. Note that either $w = w'$ or w' is a stick of w. Let $V' \subseteq V(S)$ contain all vertices of H_1, all vertices $v_{min}, \ldots, v_n$, and all sticks connected to $v_{min+1}, \ldots, v_{n-1}$. Let b be an integer defined as follows. If $min \leq m \Leftrightarrow 2$, then $b = 4$. If $min = m \Leftrightarrow 1$ then $b = 3$, and if $min = m$, then $b = 2$. For $i = 1, \ldots, b$, let x_i be vertices defined as follows: $x_1 = w$, $x_2 = w'$, if $b \geq 3$ then $x_3 = v_{m-1}$, and if $b = 4$ then $x_4 = v_{m-2}$ (note that $x_1 = x_2$ if $w = w'$). Let S_u denote the sandwich graphs with

$$\begin{aligned}
V(S_u) &= V' \cup \{\mathsf{dum}\} \\
E_1(S_u) &= E_1(S[V']) \cup \{\{\mathsf{dum}, u\}, \{\mathsf{dum}, v_{min}\}\} \\
E_2(S_u) &= E_2(S[V']) \cup \{\{\mathsf{dum}, v\} \mid v \in V'\}.
\end{aligned}$$

See also Figure 20 for and example of the underlying graphs of S_u. Note that for each i, $1 \leq i \leq b$, the sandwich graph S_u is a sandwich block with sticks and loose ends x_i and v_n.

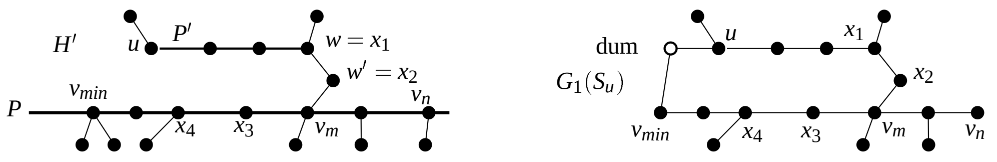

Figure 20: Example of $G_1(S_u)$ for $b = 4$.

Let c be an integer defined as follows: $c = 4$ if $n \geq m + 2$, $c = 3$ if $n = m + 1$ and $c = 2$ if $n = m$. For $i = 1, \ldots, c$, let y_i be vertices defined as follows: $y_1 = w$, $y_2 = w'$, if $c \geq 3$ then $y_3 = v_{m+1}$, and if $c = 4$ then $y_4 = v_{m+2}$ (note that $y_1 = y_2$ if $w = w'$). For each i, $1 \leq i \leq c$, let S_w^i denote the sandwich graph with

$$\begin{aligned}
V(S_w^i) &= V' \cup \{\mathsf{dum}\} \\
E_1(S_w^i) &= E_1(S[V']) \cup \{\{\{\mathsf{dum}, y_i\}, \{\mathsf{dum}, v_{min}\}\} \\
E_2(S_w^i) &= E_2(S[V']) \cup \{\{\mathsf{dum}, v\} \mid v \in V'\}.
\end{aligned}$$

See also Figure 21 for examples of the underlying graphs of S_w^i for $i = 1$ and $i = 3$ ($c = 4$). Now for $i = 1, \ldots, c$, the sandwich graph S_w^i is a sandwich block with sticks and loose ends u and v_n.

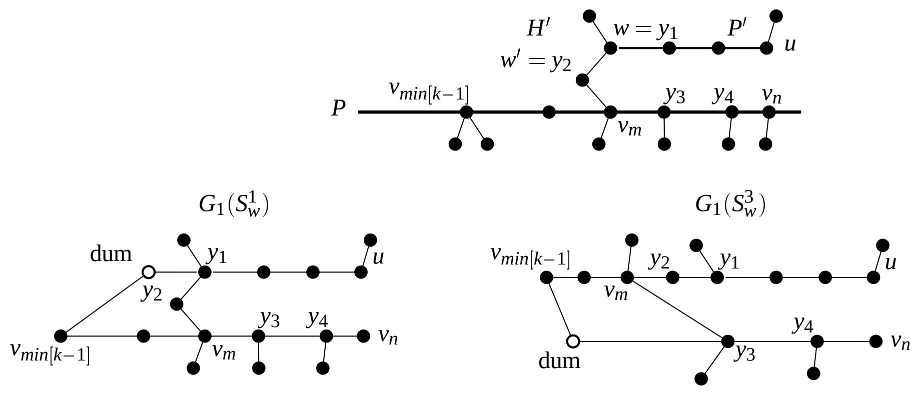

Figure 21: Example of S_w^1 and S_w^3 for $c = 3$.

Lemma 5.24. *Let j be an integer, $m \leq j \leq n$. We can extend a partial nice path decomposition of (S_p^{min}, v_{min}) into a partial nice path decomposition PD of (S_m^j, v_j) for which*

1. *H_1 uses $[j, j']$ for some $min \leq j \leq j' \leq clmin$, and*
2. *the rightmost node of PD contains an edge of S_m^m,*

if and only if one of the following holds.

a. *There is an i, $1 \leq i \leq b$, for which there is a path decomposition of width two of*

$$\tilde{S}_u = S_u[V(S_u) \Leftrightarrow \{v_{j+1}, \dots, v_n\} \Leftrightarrow \{\text{ sticks of } v_j, \dots, v_n\}]$$

with edge $\{\mathsf{dum}, v_{min}\}$ in the leftmost node, and v_j, x_i and a neighbor of x_i in S_m^m in the rightmost node.

b. *There is an i, $1 \leq i \leq c$ for which there is a path decomposition of width two of*

$$\tilde{S}_w^i = S_w^i[V(S_w^i) \Leftrightarrow \{v_{j+1}, \dots, v_n\} \Leftrightarrow \{\text{ sticks of } v_j, \dots, v_n\}]$$

with edge $\{\mathsf{dum}, v_{min}\}$ in the leftmost node, and v_j, u and a stick u' of u in the rightmost node.

Proof. For the 'if' part, we can easily combine a partial nice path decomposition of (S_p^{min}, v_{min}) and a path decomposition as described in a or b into a partial nice path decomposition of (S_m^j, v_j) satisfying conditions 1 and 2.

For the 'only if' part, suppose $PD = (V_1, \dots, V_t)$ is a partial nice path decomposition of (S_m^j, v_j) satisfying conditions 1 and 2. Suppose H_1 uses $[l, l']$ for some $min \leq l \leq l' \leq j$. There are three cases:

1. $min \leq l \leq m \leq l' \leq j$,
2. $min \leq l \leq l' < m$, and

3. $m < l \leq l' \leq j$.

1. In the same way as in the proof of Lemma 5.22, we can show that either

- there is a path decomposition of width two of $\tilde{S}_u$ with edge $\{\mathsf{dum}, v_{min}\}$ in the leftmost node and with vertices v_j, w and a stick of w in the rightmost node, and hence a holds, or
- there is a path decomposition of $\tilde{S}^1_w$ with edge v_{min} in the leftmost node and v_j, u and a stick of u in the rightmost node, and hence b holds.

2. We show that there is an i, $1 \leq i \leq b$, for which there is a path decomposition of width two of $\tilde{S}_u$ with edge $\{\mathsf{dum}, v_{min}\}$ in the leftmost node and v_j, x_i and a neighbor of x_i in S^m_m in the rightmost node.

Suppose H_1 occurs in $(V_r, \ldots, V_{r'})$. Then $V_r = \{u, u', v_l\}$ for some stick u' of u. Let W denote the set of vertices $v_l, \ldots, v_m$, the sticks of $v_{l+1}, \ldots, v_{m-1}$, and the vertices of H_1. Let $G = G_1(S[W])$, and let G' be the graph obtained from G by adding an edge between u and v_l. See also Figure 22. Note that G' is a cycle C with sticks.

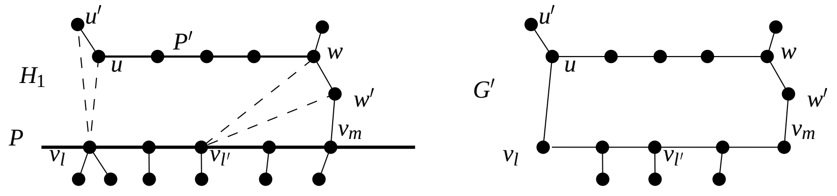

Figure 22: Example of G'.

Node V_t contains an edge of S^m_m and thus of G. Hence all edges of G occur in $(V_r, \ldots, V_t)$. Let PD' denote the sequence $(V_r, \ldots, V_t)$. Since $\{u, v_l\} \subseteq V_r$, this means that G' also occurs in PD'. Suppose C occurs in the subsequence $(V_s, \ldots, V_{s'})$ of PD'. Since v_m occurs in $(V_s, \ldots, V_{s'})$, $v_j \in V_t$, and there is a path from v_m to v_j, this means that $v_m \in V_{s'}$ (Lemma 3.4.3 of de Fluiter [1997]), and hence $V_{s'} \cap V(C) \subseteq \{v_m, x_1, x_2, \ldots, x_b\}$ (Lemma 3.4.5 of de Fluiter [1997]). But that means that V_t contains one of the vertices $x_1, \ldots, x_b$ and a neighbor of this vertex in G: if $t = s'$, then either $\{x_1, v_m\} \subseteq V_t$ or $\{x_3, v_m\} \subseteq V_t$, and if $t > s'$, then V_t contains a stick of C in G', and a vertex of C to which this stick is connected. Since v_m has no sticks in G', this means that $x_i \in V_t$ for some i, $1 \leq i \leq b$. Let i^*, $1 \leq i^* \leq b$, be such that $x_{i^*} \in V_t$, and let $y \in V(G)$ such that $\{x_{i^*}, y\} \in E(G)$ and $y \in V_t$.

We can now transform PD' into a path decomposition of width two of $\tilde{S}_u$ with $\{\mathsf{dum}, v_{min}\}$ in the leftmost node and v_j, x_{i^*} and y in the rightmost node. Let S' denote the sandwich graph of pathwidth one induced by the vertices $v_{min}, \ldots, v_l$ and the sticks of $v_{min+1}, \ldots, v_l$. Make a path decomposition PD_1 of width one of S' with vertex v_{min} in the leftmost node and vertex v_l in the rightmost node, and add vertex dum to each node of this path decomposition. Let PD' be $PD_1 +\!\!+ \{\{v_l, u, \mathsf{dum}\} +\!\!+ PD'$. Then PD' is the desired path decomposition of $\tilde{S}_u$.

3. In the same way as for 2, we can show that there is an i, $1 \leq i \leq c$, for which there is a path decomposition of width two of $\tilde{S}^i_w$ with v_{min} and dum in the leftmost node, and v_j, u and a stick of u in the rightmost node. □

The lemma implies that *clmin* is the smallest j, $m \leq j \leq n$, for which either a or b from the lemma holds. If such a j exists, then $cl.ok = \text{true}$, and otherwise, $cl.ok = \text{false}$. Hence we have the following result.

Lemma 5.25. *If H_1 has type* I, *then it takes $O(N^2)$ time to compute cl, where N is the number of vertices of S_u^1.*

Proof. For each i, S_u is a sandwich block with sticks and loose ends x_i and v_n, and S_w^i is a sandwich block with sticks and loose ends u and v_n. Hence the lemma follows from Corollary 4.2 and the proof of Lemma 5.16. □

This completes the description of the computation of *cl* for the case that $nr = 1$.

Computation of *fr*

An arguments similar to the argument for the computation of *fr* for the case that $nr > 1$ shows that Corollary 5.5 also holds for the case that $nr = 1$. We do not give a precise description of this argument here: all computations can be derived directly from the computations for the case that $nr > 1$.

This completes the description of the computation of $all[k]$ for the case that the number nr of partial one-paths connected to v_m is one. We conclude with the following theorem.

Theorem 5.3. *Let $k \geq 1$. Given the values of $all[l]$ and $allbo[l]$ for $l < k$, it takes $O(N^2)$ time to compute $all[k]$, where N denotes the number of vertices $v_{i_{k-2}}, \ldots, v_{i_{k+1}}$, all sticks of $v_{i_{k-2}+1}, \ldots, v_{i_{k+1}-1}$, and vertices of all partial one-paths connected to $v_{i_{k-1}}$ and v_{i_k}.*

The Computation of $allbo[k]$

If $nr = 1$, then $allbo[k].ok = \text{false}$. Suppose $nr > 1$. Let $H'_1, \ldots, H'_{nr'}$ denote the partial one-paths connected to v_m for which each vertex w has $\{w, v_m\} \in E_2$.

If $nr' = 0$, then by definition, we make $allbo[k].ok = \text{false}$. If $1 \leq nr' \leq 2$, then for $i = 1, 2$, we check whether there is a partial nice path decomposition of $(S_m^m \Leftrightarrow H'_i, v_m)$. If there is no i for which this holds, then $allbo[k].ok = \text{false}$, otherwise,

$$allbo[k].ok = \text{true, and}$$
$$allbo[k].tr = \{H'_i \mid 1 \leq i \leq 2 \wedge \text{ there is a partial nice path decomposition of } (S_m^m \Leftrightarrow H'_i, v_m).$$

To check whether there is a partial nice path decomposition of $(S_m^m \Leftrightarrow H'_i, v_m)$ for some i, $1 \leq i \leq nr'$, we use the same computations as are used for the determination of $all[k]$ if $nr > 1$. We do not describe these computations again, but immediately conclude with the following theorem.

Theorem 5.4. *Let $k \geq 1$. Given the values of $all[l]$ and $allbo[l]$ for $l < k$, it takes $O(N^2)$ time to compute $allbo[k]$, where N denotes the number of vertices $v_{ik-2}, \ldots, v_{i_{k+1}}$, all sticks of $v_{i_{k-2}+1}, \ldots, v_{i_{k+1}-1}$, and vertices of all partial one-paths connected to $v_{i_{k-1}}$ and v_{i_k}.*

This completes the description of the computations of *all* and *allbo*. From Theorem 5.4 it can be seen that algorithm Nice_Path as described on page 43 takes $O(n^2)$ time, where n denotes the number of vertices of the sandwich tree.

To complete this section, we give algorithm 3-ISG_Tree, which, given a sandwich tree S, returns true if there is a three-intervalization of S, and false otherwise.

```
Algorithm 3-ISG_Tree($S$)
Input: Sandwich tree $S = (V, E_1, E_2)$
Output: true if there is a three-intervalization of $S$, false otherwise
1.  Check if $G_1(S)$ has pathwidth two, if not, return false.
2.  Find the set $\mathcal{P}_2(G_1(S))$, and a set $\mathcal{A}$ of potentially nice paths of $S$, and for each $P \in \mathcal{A}$, the
    partial one-paths $H'$ connected to $P$ and there sets $\mathcal{P}_1(H')$.
3.  for all $P \in \mathcal{A}$
4.      do if Nice_Path($P$) then return true
5.  return false
```

This algorithm can again be made constructive.

Theorem 5.5. *There exists an $O(n^2)$ algorithm that solves* 3*-ISG for sandwich trees.*

6 Three-Intervalizing Sandwich Graphs

The algorithm for 3-ISG on sandwich graphs is very similar to the algorithm for 3-ISG on sandwich trees. Therefore, we only give a brief description of this algorithm.

Suppose we are given an input sandwich graph S. Let $G = G_1(S)$. If G is not connected, then we apply the algorithm for all connected components of G. Suppose G is connected. If S is a sandwich block with sticks, or if S is a sandwich tree, then we can use one of the algorithms given in Sections 4 and 5. Otherwise, the following is done. First, it is checked whether G has pathwidth at most two, and if so, the structure of G is computed as in Chapter 3 of de Fluiter [1997]: the set of paths $\mathcal{P}_G$ is computed, and for each path $P \in \mathcal{P}_G$, the set of partial one-paths connected to P is computed, and the interconnections between vertices of P, partial one-paths connected to P and blocks of G are determined.

From this set $\mathcal{P}_G$ of paths, it is then computed whether there is a path decomposition of width two of S. We again only consider nice path decompositions, which are defined slightly different from the nice path decompositions of sandwich trees.

Definition 6.1 (Nice Path Decomposition). Let $S = (V, E_1, E_2)$ be a sandwich graph of pathwidth two, let $G = G_1(S)$, suppose G is connected, but is not a tree. Let $P_G = (v_1, \dots, v_s)$, let $PD = (V_1, \dots, V_t)$ be a path decomposition of width two of S. Then PD is a *nice path decomposition* of S if there are no two consecutive nodes which are equal, V_1 contains an edge $e = \{v, v'\} \in E_1$ and V_t contains an edge $e' = \{x, x'\} \in E_1$, in such a way that $x \neq v$ and the path from v to x contains P_G. Furthermore, one of the following condition holds for V_1 and e, and analogously for V_t and e'.

1. $s = 0$, B is the only block of G, $e \in E(H')$ for some component H' of G_{tr} containing a vertex $w \in V(B)$ of state E1 or I1, such that v is an end point of the path P' containing $P_1(H')$ and w, and $v \neq w$.

2. $s = 0$, B is the only block of G, $e \in E(G)$, $v \in V(B)$ and either v' is a stick adjacent to v, or $v' \in V(B)$.

3. $s \geq 1$, $e \in E(H')$ for some partial one-path H' connected to v_1 such that v is an end point of some path $P' \in P_1(H')$,

4. $s \geq 1$, $e \in E(H')$ for some component H' of G_{tr} containing a vertex w of state E1 or I1 of a block containing v_1, such that v is an end point of the path P' containing $P_1(H')$ and w, and $v \neq w$.

5. $s \geq 1$, there is a block B containing v_1 such that $v \in V(B) \Leftrightarrow \{v_1\}$, and either $\{v, v'\} \in E(B)$ or v' is a stick adjacent to v.

The *nice path* P' corresponding to nice path decomposition PD is defined as follows. If $s = 0$, then P' is the empty path if condition 2 holds for both V_1 and V_t. If condition 1 holds for V_1, and 2 for V_t, then P' is the path from v to the vertex $w \in V(B)$ for which v and w are in the same component of G_{tr}. Analogously, if condition 1 holds for V_t and 2 holds for V_1, then P' is the path from the vertex $w \in V(B)$ to x, such that w and x are in the same component of G_{tr}. If condition 1 holds for both V_1 and V_t, then P' is the largest common subsequence of all paths from v to x. If $s \geq 1$, then P' is the largest common subsequence of all paths from w to w' in G, where $w = v_1$ if condition 5 holds for V_1, $w = v$ otherwise, and $w' = v_s$ if condition 5 holds for V_t, $w = x$ otherwise.

Figure 23 shows an example of all conditions in Definition 6.1. In G_1, $s = 0$, and in G_2, $s \geq 1$. If v and v' are equal to a_1 and a'_1, b_1 and b'_1 or c_1 and c'_1, then case 1 holds. If $v \in V(B_1)$, and v' is either a stick adjacent to v, or $\{v, v'\} \in E(B_1)$, then case 2 holds (e.g. if $v = d_1$ and $v' = d'_1$). If $v = b_2$, and v' is equal to b'_2 or b''_2, then case 3 holds. If $v = a_2$ and $v' = a'_2$, then case 4 holds, and if $v \in V(B_2)$, and v' is either a stick adjacent to v, or $\{v, v'\} \in E(B_2)$ (e.g. if $v = c_2$ and $v' = c'_2$)then case 5 holds.

The analog of Theorem 5.1 also holds for general sandwich graphs: S has pathwidth two if and only if there is a nice path decomposition of width two of S (which can again be proved by 'unfolding').

In the algorithm, we only check for a bounded number of nice paths (a set of 'potentially' nice paths) whether there is a nice path decomposition with this nice path. We can show with a lemma analogous to Lemma 5.5 and Lemma 5.6 that this is possible.

Checking whether there is a nice path decomposition with a given potentially nice path $P = (v_1, \dots, v_q)$ is done in the same way as for sandwich trees: we start with $m = 1$, and 'process' all partial one-paths connected to v_1, and, in addition, all blocks containing v_1. Then, we repeatedly increment m, and after each increment operation, we 'process' the partial one-paths connected to v_m, and the blocks containing v_m, by using the information from v_i, $i < m$. Finally, when $m = q$, we have processed all partial one-paths and blocks, and we know whether there is a nice path decomposition of S with nice path P.

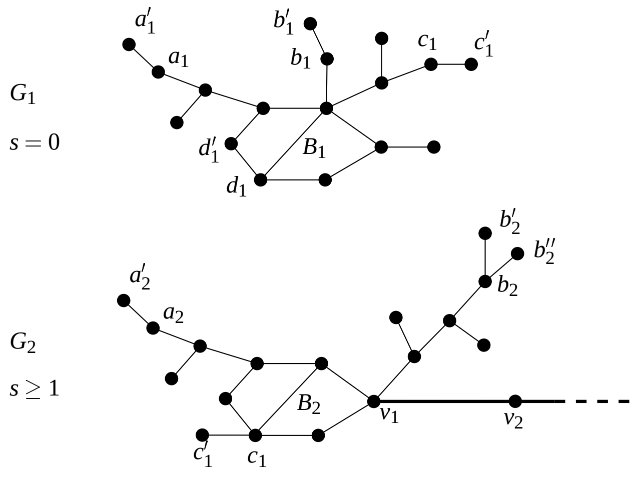

Figure 23: Examples of possible values of v and v' as defined in Definition 6.1.

The processing of all partial one-paths connected to a vertex v_m, and all blocks containing v_m strongly resembles the processing of partial one-paths as described in Section 5 for sandwich trees. There are a lot more cases to consider, but each case can be solved in a similar way, with the use of Lemma 4.6 and Corollary 4.2.

References

Bodlaender, H. L. and B. de Fluiter [1995]. Intervalizing k-colored graphs. Technical Report UU-CS-1995-15, Department of Computer Science, Utrecht University, Utrecht.

Bodlaender, H. L. and B. de Fluiter [1996]. On intervalizing k-colored graphs for DNA physical mapping. *Disc. Appl. Math. 71*, 55–77.

Bodlaender, H. L. and T. Kloks [1993]. A simple linear time algorithm for triangulating three-colored graphs. *J. Algorithms 15*, 160–172.

de Fluiter, B. [1997]. *Algorithms for Graphs of Small Treewidth*. Ph.D. thesis, Utrecht University.

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

Fellows, M. R., M. T. Hallett, and H. T. Wareham [1993]. DNA physical mapping: Three ways difficult (extended abstract). In T. Lengauer (Ed.), *Proceedings of the 1st Annual European Symposium on Algorithms ESA'96*, Volume 726 of Lecture Notes in Computer Science, pp. 157–168. Springer-Verlag, Berlin.

Golumbic, M. C., H. Kaplan, and R. Shamir [1994]. On the complexity of DNA physical mapping. *Advances in Applied Mathematics 15*, 251–261.

MÖHRING, R. H. [1990]. Graph problems related to gate matrix layout and PLA folding. In E. Mayr, H. Noltemeier, and M. Sysło (Eds.), *Computational Graph Theory, Computing Suppl. 7*, pp. 17–51. Springer-Verlag, Berlin.