Domino Treewidth

H.L. Bodlaender and J. Engelfriet

UU-CS-1994-11
February 1994

Utrecht University

Department of Computer Science

Padualaan 14, P.O. Box 80.089,
3508 TB Utrecht, The Netherlands,
Tel. : ... + 31 - 30 - 531454

Domino Treewidth

H.L. Bodlaender and J. Engelfriet

Technical Report UU-CS-1994-11
February 1994

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

ISSN: 0924-3275

Domino Treewidth

Hans L. Bodlaender* Joost Engelfriet†

Abstract

We consider a special variant of tree-decompositions, called *domino tree-decompositions*, and the related notion of *domino treewidth*. In a domino tree-decomposition, each vertex of the graph belongs to at most two nodes of the tree. We prove that for every k, d, there exists a constant $c_{k,d}$ such that a graph with treewidth at most k and maximum degree at most d has domino treewidth at most $c_{k,d}$. The domino treewidth of a tree can be computed in $O(n^2 \log n)$ time. There exist polynomial time algorithms that — for fixed k — decide whether a given graph G has domino treewidth at most k. If k is not fixed, this problem is NP-complete. The domino treewidth problem is hard for the complexity classes $W[t]$ for all $t \in \mathbf{N}$, and hence the problem for fixed k is unlikely to be solvable in $O(n^c)$, where c is a constant, not depending on k.

1 Introduction

A topic of much recent research in algorithmic graph theory is the treewidth of graphs. Applications of this research range from VLSI theory to expert systems (and many more). (See e.g., [5] for an overview.) In this paper, we introduce a special variant of treewidth: *domino treewidth*. This notion is derived from the usual notion of treewidth, by additionally requiring that every vertex $v \in V$ belongs to at most two node sets X_i. (See Section 2 for the precise definitions.) Our interest in this notion is largely due to a maybe somewhat surprising result: for graphs of bounded degree and bounded treewidth, there is a uniform upper bound on the domino treewidth. The proof of this result is given in Section 3. We also investigate the algorithmic aspects of domino treewidth. In Section 5, we show that the problem to determine the domino treewidth of a given graph is NP-hard, and also is hard

*Department of Computer Science, Utrecht University, P.O. Box 80.089, 3508 TB Utrecht, the Netherlands. This author was partially supported by the ESPRIT Basic Research Actions of the EC under contract 7141 (project ALCOM II).

†Department of Computer Science, Leiden University, P.O. Box 9512, 2300 RA Leiden, the Netherlands. This author was supported by the ESPRIT Basic Research Working Group COMPUGRAPH II.

for the complexity classes $W[t]$, for all $t \in \mathbf{N}$. The latter result tells us that it is unlikely that the problem, for fixed k, to decide whether a given graph has domino treewidth $\leq k$, can be solved in $O(n^c)$ time, where c is a constant, not depending on k. Some special cases can be solved in polynomial time: for fixed k, one can check in polynomial time whether the domino treewidth of a given graph is at most k. For trees, the domino treewidth can be computed in $O(n^2 \log n)$ time. These results are shown in Section 4.

2 Definitions and preliminary results

The notion of treewidth has been introduced by Robertson and Seymour [17].

Definition. A *tree-decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$ with $\{X_i \mid i \in I\}$ a collection of subsets of V, and $T = (I, F)$ a tree, such that

- $\bigcup_{i \in I} X_i = V$
- for all edges $\{v, w\} \in E$ there is an $i \in I$ with $v, w \in X_i$
- for all $i, j, k \in I$: if j is on the path from i to k in T, then $X_i \cap X_k \subseteq X_j$.

The width of a tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i| - 1$. The treewidth of a graph $G = (V, E)$ is the minimum width over all tree-decompositions of G.

Let $(\{X_i \mid i \in I\}, T = (I, F))$ be a tree-decomposition of $G = (V, E)$. For each vertex $v \in V$, we let T_v be the subgraph of T, induced by $\{i \in I \mid v \in X_i\}$. Condition (iii) above can be rephrased as: for every $v \in V$, T_v is connected, or as: for every $v \in V$, T_v is a tree.

We use $size(G)$ to denote the number of vertices of G, and $deg(v)$ to denote the degree of vertex v. The degree of G is the maximum degree of its vertices.

Definition. A tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ of $G = (V, E)$ is a *domino tree-decomposition*, if for every vertex $v \in V$, $size(T_v) \leq 2$, i.e., every vertex belongs to at most two sets X_i. The domino treewidth of a graph G is the minimum width over all domino tree-decompositions of G.

Equivalently, the definition of a domino tree-decomposition is obtained from the first definition by changing its third condition into the following requirement: for all $i, j \in I$, if $i \neq j$ and $\{i, j\} \notin F$, then $X_i \cap X_j = \emptyset$. For a connected graph G, this means that T is the intersection graph of the family $\{X_i \mid i \in I\}$.

Note that if d is the maximum degree of a graph G and k is its domino treewidth, then $d \leq 2k$. Hence the domino treewidth of a graph is at least half its maximum degree. This shows, e.g., that there is no bound on the domino treewidth of trees.

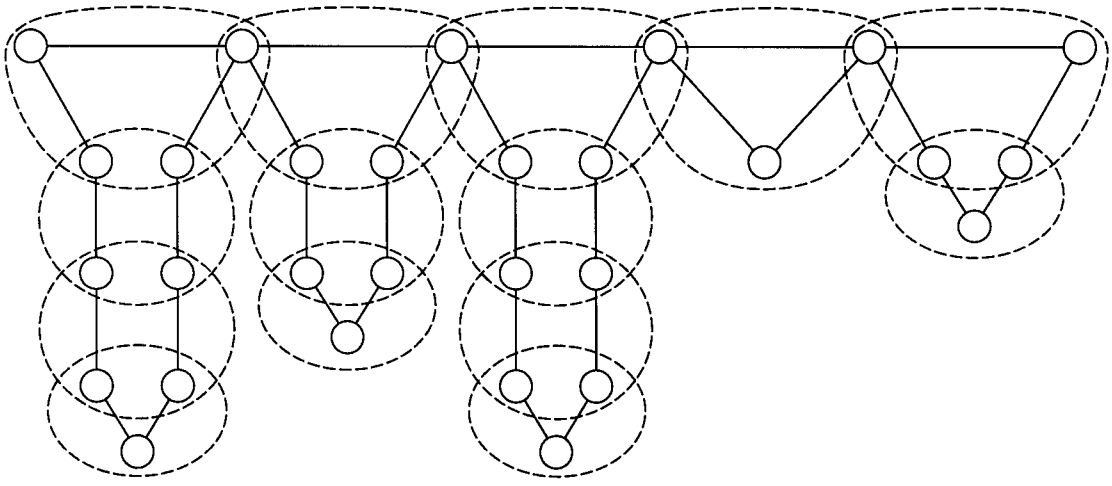

Figure 1: A domino tree-decomposition

As an example, the graph G in Figure 1 (and any similar graph) has domino treewidth 3. The dotted lines indicate a domino tree-decomposition of width 3. Note that domino tree-decompositions are easy to visualize. One easily observes that G has treewidth 2: G is outerplanar and all outerplanar graphs have treewidth ≤ 2 (see e.g., [3]).

A well-known lemma for tree-decompositions is the following.

Lemma 1 *Let $(\{X_i \mid i \in I\}, T = (I, F))$ be a tree-decomposition of $G = (V, E)$. Let $v_0, v_1, \ldots, v_r$ be a path in G. Suppose $v_0 \in X_i$, $v_r \in X_k$, and j is on the path from i to k in T. Then $X_j \cap \{v_0, \ldots, v_r\} \neq \emptyset$.*

A slightly stronger variant holds for domino tree-decompositions. The following lemma will be used later in this paper.

Lemma 2 *Let $(\{X_i \mid i \in I\}, T = (I, F))$ be a domino tree-decomposition of $G = (V, E)$. Let $v_0, v_1, \ldots, v_r$ be a path in G. Suppose $v_0 \in X_i$, $v_r \in X_k$, and j is on the path from i to k in T, $j \neq i$, $j \neq k$. Then $|X_j \cap \{v_0, \ldots, v_r\}| \geq 2$.*

Proof: By Lemma 1, $X_j \cap \{v_0, \ldots, v_r\} \neq \emptyset$. Suppose that $X_j \cap \{v_0, \ldots, v_r\} = \{v_\alpha\}$. Let j' be the unique node with $j' \neq j$ and $v_\alpha \in X_{j'}$. As the neighbors of v_α on the path do not belong to X_j, they must belong to $X_{j'}$. Either j is on the path from j' to i in T, or on the path from j' to k in T. Suppose the former. If $\alpha = 0$, then $v_\alpha \in X_j \cap X_{j'} \cap X_i$, and $i \neq j'$: contradiction with dominoness. If $\alpha > 0$, then $v_{\alpha-1} \in X_{j'}$, and hence, by Lemma 1, $X_j \cap \{v_0, \ldots, v_{\alpha-1}\} \neq \emptyset$, contradiction. In the case that j is on the path from j' to k, we can obtain a contradiction in a similar way. We conclude that X_j contains at least two vertices from the path $v_0, \ldots, v_r$. □

Another well-known lemma is the following (for a short proof, see e.g. [9]).

Lemma 3 *Let* $(\{X_i \mid i \in I\}, T = (I, F))$ *be a tree-decomposition of* $G = (V, E)$. *Suppose* $W \subseteq V$ *forms a clique in* G. *Then there must be a node* $i \in I$ *with* $W \subseteq X_i$.

3 Bounded treewidth and degree corresponds with bounded domino treewidth

In this section, we give the proof of a structural result: graphs with bounded degree and bounded treewidth have bounded domino treewidth. A result like this was first shown in the context of graph grammars in [14].

The proof of this result is rather lengthy, and will be given with the help of several lemmas. First, we introduce some notions, needed for the proof.

For technical reasons, we assume in this section that the trees T taken from tree-decompositions are rooted. This induces rootedness of each T_v in a natural way: the root of T_v is the node of T_v that is closest to the root of T. For a node i of a rooted tree, the parent of i and children of i (if they exist) are defined in the usual way. The *rank* of i is the number of children of i. The rank of a tree is the maximum rank of its nodes. A *leaf* is a node of rank 0. Whenever we write down a tree arc $\{i, j\}$ as (i, j), this implicitly means that i is the parent of j, i.e., i is closer to the root of T than j. The height of a tree is the maximum distance between the root and the leaves. If i is a node, and T a tree, we sometimes denote the fact that x is a node of T as $x \in T$.

Also for technical reasons, we will, for each tree-decomposition, fix a mapping $\psi : I \to \mathcal{P}(E)$, such that

- for every $i \in I$, $\{v, w\} \in \psi(i)$: $\{v, w\} \subseteq X_i$.
- for every $\{v, w\} \in E$, there is a unique $i \in I$ with $\{v, w\} \in \psi(i)$.

Note, that by the second condition of tree-decomposition, such a mapping ψ always exists (but, in general, there may be more than one). The mapping ψ fixes a distribution of the edges of G over the nodes in I.

Let $XT = (\{X_i \mid i \in I\}, T = (I, F))$ be a tree-decomposition of $G = (V, E)$, and let mapping ψ and subtrees T_v be defined as in Section 2 and above.

Let $v \in V$, and let $i \in I$ be a node in T_v (i.e., $v \in X_i$). We say that i is a *useful* node of T_v, if there exists an edge $\{v, w\} \in \psi(i)$, otherwise we say that i is a *useless* node of T_v.

Our first aim is to transform a tree-decomposition in such a way, that the tree has no useless nodes of rank 0 or 1 (cf. Lemma 7.) We start with removing useless nodes of degree 1.

We say that a tree-decomposition XT has property **(U1)**, if for every vertex $v \in V$, T_v has no useless nodes of degree 1.

Lemma 4 *Let $(\{X_i \mid i \in I\}, T = (I, F))$ be a tree-decomposition of $G = (V, E)$ of width k. There exists a tree-decomposition $(\{Y_i \mid i \in I\}, T = (I, F))$ of $G = (V, E)$ of width $\leq k$ that has property* **(U1)**.

Proof: If i is a useless node of T_v of degree 1, then remove i from T_v, i.e., remove v from X_i. Repeat this procedure as long as possible. The resulting tree-decomposition has property **(U1)**, and width $\leq k$. □

We need some extra terminology. Let $XT = (\{X_i \mid i \in I\}, T = (I, F))$ be a tree-decomposition, let the subtrees T_v and mapping ψ be as before. For every node $i \in I$, define

$$chain_{XT}(i) = \{v \in X_i \mid i \text{ is a useless node of rank 1 in } T_v\}$$

For an arc (i, j) of T (recall that the notation means that i is the parent of j), define

$$pass_{XT}(i, j) = chain_{XT}(i) \cap X_j$$

Note that if j, j' are distinct children of i, then $pass_{XT}(i, j) \cap pass_{XT}(i, j') = \emptyset$. We define

$$maxp(XT) = \max\{|pass_{XT}(i, j)| \mid (i, j) \in F\}$$

Lemma 5 *For every k, $r \in \mathbf{N}$, there exist k', $r' \in \mathbf{N}$, such that for every graph $G = (V, E)$ and for every tree-decomposition XT of G with property* **(U1)** *and of width $\leq k$, with the rank of $T \leq r$, if $maxp(XT) > 0$, then there exists a tree-decomposition $XT' = (\{X'_i \mid i \in I'\}, T' = (I', F'))$ of G with property* **(U1)** *and of width $\leq k'$ with the rank of $T' \leq r'$, such that $maxp(XT') < maxp(XT)$. Moreover the height and size of T' are at most the height and size of T, respectively.*

Proof: We begin by making an intermediate tree-decomposition XT'', fulfilling the property, that for every arc (i, j), either $|pass_{XT''}(i, j)| < maxp(XT)$, or for every arc (j, h) it holds that $|pass_{XT''}(j, h)| < maxp(XT)$.

Proposition 6 *For every k, $r \in \mathbf{N}$, there exist k'', $r'' \in \mathbf{N}$, such that for every graph $G = (V, E)$ and for every tree-decomposition XT of G with property* **(U1)** *and of width $\leq k$, with the rank of $T \leq r$, if $maxp(XT) > 0$, then there exists a tree-decomposition $XT'' = (\{X''_i \mid i \in I''\}, T'' = (I'', F''))$ of G with property* **(U1)** *and of width $\leq k''$ with the rank of $T'' \leq r''$, such that for every arc (i, j) in T'', either $|pass_{XT''}(i, j)| < maxp(XT)$, or for every arc (j, h) it holds that $|pass_{XT''}(j, h)| < maxp(XT)$. Moreover the height and size of T'' are at most the height and size of T, respectively.*

Proof: We say that an arc (i,j) is *maximal*, if $|pass_{XT}(i,j)| = maxp(XT)$, and we say that a node i is maximal, if it has a maximal arc (i,j). The idea of the construction is to fold certain paths of T, that consist of maximal arcs (i,j), all with the same set of vertices $pass_{XT}(i,j)$. These vertices can then be removed from the folded path. We first define these paths. To this aim, we define a marking of the nodes and arcs of T, in a top-down fashion. A node can either stay unmarked, or can be marked as *initial*, *middle*, or *final*. An arc (i,j) can either stay unmarked, or can be marked with the set of vertices $pass_{XT}(i,j)$.

We now describe the marking procedure in detail. Consider a node i of T, and assume that the nodes and arcs on the path from i to the root already have been considered. Let h be the parent of i, if existing. We distinguish two cases.

Case 1. The arc (h,i) is unmarked, or i is the root of T. If i is a maximal node, then mark i as *initial*, and mark each maximal arc (i,j) with $pass_{XT}(i,j)$. After that, consider the children of i.

Case 2. The arc (h,i) is marked with the set of vertices π. If there exists an arc (i,j) with $pass_{XT}(i,j) = \pi$, then mark this arc with π, and mark i as *middle*. Such an arc is necessarily unique. If there is no such arc, then mark i as *final*. After that, consider the children of i.

It should be clear that the marked subgraph of T is a disjoint collection of trees. Each such tree has a root, marked *initial*, which is of rank at least 1; all other nodes in such a tree are marked either as *middle*, in which case they have rank 1, or as *final*, in which case they have rank 0. Moreover, all the arcs on one path from the initial node to a final node have the same mark; we will call this the mark of the path. Note that the marks of distinct paths are disjoint. Figure 2(a) shows an example of a marked subtree (in which all unlabeled nodes are marked 'middle', and π_1, π_2 are disjoint sets of vertices); the figure also shows some (unmarked) arcs that do not belong to the marked subtree.

We now describe the procedure, that computes the desired tree-decomposition XT''. Perform the following steps for each marked subtree of T.

1. Identify the initial node with all the final nodes. The resulting node of T'' is called a *bridge* node.

2. For each path from the initial node to a final node, let $i_1, \ldots, i_\beta$ be the middle nodes on the path, $\beta \geq 0$. For α, $1 \leq \alpha \leq \beta$, identify i_α with $i_{\beta-\alpha+1}$, i.e., i_1 with i_β, i_2 with $i_{\beta-1}$, i_3 with $i_{\beta-2}$, etc. The resulting nodes of T'' are called *folded* nodes.

3. Define the sets X''_j of folded or bridge nodes as the union of the sets X_i of the corresponding identified nodes. The mapping ψ'' is obtained similarly. (However, see below at point 5!)

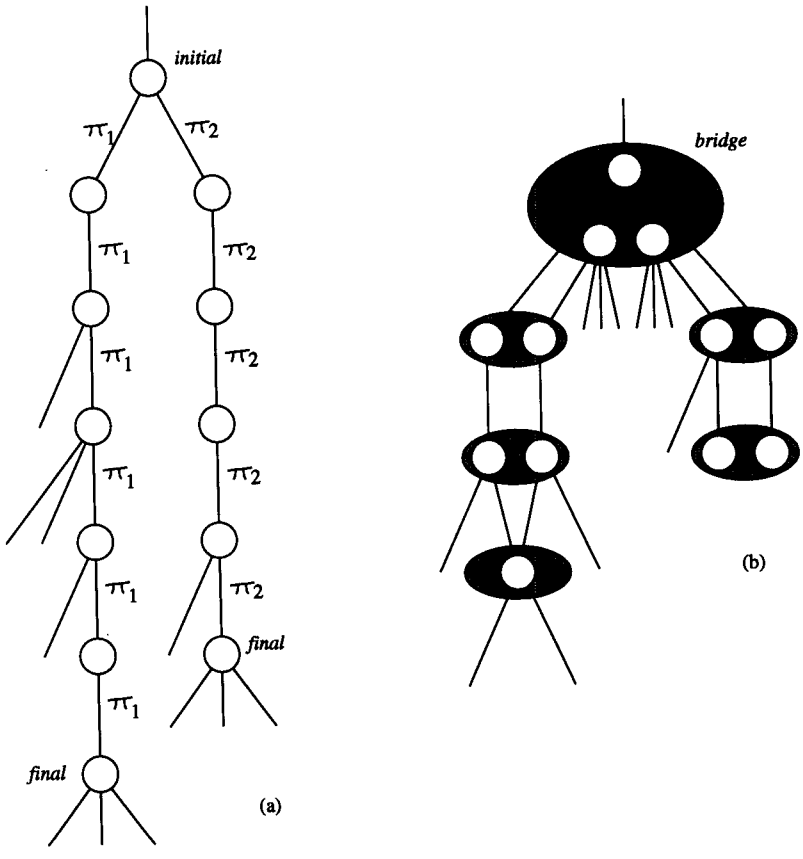

Figure 2: (a) A marked subtree of T. (b) The folded marked subtree.

4. Unmarked nodes of T will stay the same in XT'', with the same values for X_i and ψ. These will be called the *old* nodes of T''.

5. For each folded node j, remove π from X''_j, where π is the mark of the path to which the corresponding middle nodes belong.

In other words, each marked path from an initial node to a final node is folded in two, and the vertices in the mark of that path are removed from the sets X_i. Note that if in Step 2, β is odd, then the node that is half-way the path is not identified with another node. Figure 2(b) shows how the marked subtree of Figure 2(a) is folded. The nodes of T'' are shown as shaded circles, containing the corresponding identified nodes of T (where all unlabeled nodes of T'' are folded nodes).

We first prove that XT'' is a tree-decomposition of G. It is easy to see that after we have applied Steps 1 – 4, we still have a tree-decomposition XT^* of G. For each path from an initial node to a final node with mark π, the middle nodes are useless nodes of rank 1 in T_v for every $v \in \pi$. Hence, in T^*_v, the corresponding folded nodes form a chain of useless nodes of rank 1, ending in a useless leaf. So, we can safely remove v from all X^*_j for all folded nodes j (as in the proof of Lemma 4), which is done in Step 5.

Since at most $r+1$ nodes are identified with each other, the width of XT'' is at most $(r+1)(k+1)-1$, and the rank of T'' is at most $(r+1)r$.

It remains to show that XT'' has property **(U1)** and the property, stated in the proposition. For the second property, it is clearly sufficient to show that for every arc (i,j) in T'':

- if j is a folded node, then $|pass_{XT''}(i,j)| = 0$.
- if i is an old node or a bridge node, then $|pass_{XT''}(i,j)| < maxp(XT)$.

We first show that if j is a folded node, then $|pass_{XT''}(i,j)| = 0$. Consider a path from an initial node i_0 to a final node $i_{\beta+1}$, and let $i_1, \ldots, i_\beta$ be the middle nodes on the path, in top-down order. Let π be the mark of the path.

First, we consider the case that i and j are both folded nodes. Suppose i is the result of identifying i_α and $i_{\beta-\alpha+1}$, and j is the result of identifying $i_{\alpha+1}$ and $i_{\beta-\alpha}$. Suppose $v \in pass_{XT''}(i,j)$. Then $v \notin \pi$, (i,j) is an arc of T''_v, and i is a useless node of rank 1 in T''_v. Then, by the third property of tree-decomposition, either $(i_\alpha, i_{\alpha+1})$ or $(i_{\beta-\alpha}, i_{\beta-\alpha+1})$ is an arc of T_v, and i_α and $i_{\beta-\alpha+1}$, when belonging to T_v, are useless in T_v. The arcs (i_α, h) with $h \neq i_{\alpha+1}$ and the arcs $(i_{\beta-\alpha+1}, h')$, $h' \neq i_{\beta-\alpha+2}$ do not belong to T_v, otherwise, i has rank more than 1 in T''_v. In the case that $(i_\alpha, i_{\alpha+1})$ is an arc of T_v, then we obtain that $v \in pass_{XT}(i_\alpha, i_{\alpha+1}) = \pi$, contradiction. In the case that $(i_{\beta-\alpha+1}, i_{\beta-\alpha+2})$ is an arc of T_v, then we obtain that $v \in pass_{XT}(i_{\beta-\alpha+1}, i_{\beta-\alpha+2}) = \pi$, contradiction. The remaining case is that $(i_{\beta-\alpha}, i_{\beta-\alpha+1})$ is an arc of T_v, but $(i_{\beta-\alpha+1}, i_{\beta-\alpha+2})$ is not. In this case, $i_{\beta-\alpha+1}$ is a useless leaf of T_v, which contradicts property **(U1)**. So, if i and j are both folded nodes, then $pass_{XT''}(i,j) = \emptyset$.

The other case to consider is that i is a bridge node. Suppose i is the result of identifying i_0 with all final nodes, including $i_{\beta+1}$, and suppose j is the result of identifying i_1 with i_β. Assume that $v \in pass_{XT''}(i,j)$. As before, we have that $v \notin \pi$, and either (i_0, i_1) or $(i_\beta, i_{\beta+1})$ is an arc of T_v. In the case that (i_0, i_1) is an arc of T_v, we obtain that $v \in pass_{XT}(i_0, i_1)$. In the case that $(i_\beta, i_{\beta+1})$ is an arc of T_v, we obtain that $i_{\beta+1}$ is a useless leaf of T_v. In both cases, we have a contradiction. This proves that if j is a folded node, then $|pass_{XT''}(i,j)| = 0$.

Next, we show that for old nodes i, $|pass_{XT''}(i,j)| < maxp(XT)$. j is either an old node, or the result of an identification of some nodes, one of which is a child of i. Let j_0 be the node in T, corresponding to j, such that there is an arc (i, j_0) in T. Since (i, j_0) is unmarked in T, (i, j_0) is not maximal, and hence $|pass_{XT}(i, j_0)| < maxp(XT)$. Clearly, $pass_{XT}(i, j_0) = pass_{XT''}(i, j)$: i is a useless node of rank 1 in T_v, if and only if it is a useless node of rank 1 in T''_v. Hence $|pass_{XT''}(i,j)| < maxp(XT)$.

We now show that for bridge nodes i, $|pass_{XT''}(i,j)| < maxp(XT)$. Let i_0 be the initial node, and $f_1, \ldots, f_m$ be the final nodes of T, corresponding to i. There are three cases.

Case 1. There is an unmarked arc (i_0, j_0), such that j_0 corresponds to j. Consider $v \in pass_{XT''}(i,j)$. Then (i_0, j_0) is in T_v, and hence $v \notin pass_{XT}(i_0, h)$ for all $h \neq j_0$ (otherwise, we would get a contradiction with the rank of i_0 in T_v). So, every child h of i_0 with $v \in X_h$ corresponds to a node h' in T'' with $v \in X''_{h'}$. It follows that $v \in pass_{XT}(i_0, j_0)$. As (i_0, j_0) is unmarked, and hence not maximal, we have that $|pass_{XT''}(i,j)| \leq |pass_{XT}(i_0, j_0)| < maxp(XT)$.

Case 2. There is a marked arc (i_0, j_0), such that j_0 corresponds to j and j_0 is a middle node. Then j is a folded node, and hence, by the proof given above, $|pass_{XT''}(i,j)| = 0$.

Case 3. j corresponds to a node j_0, that is a child in T of a final node f_δ, for some δ, $1 \leq \delta \leq m$. Let (i_0, h) be the first arc of T on the path from i_0 to f_δ. We now claim that $pass_{XT''}(i,j) = pass_{XT}(i_0, h) \cap pass_{XT}(f_\delta, j_0)$. It is easy to verify that $pass_{XT}(i_0, h) \cap pass_{XT}(f_\delta, j_0) \subseteq pass_{XT''}(i,j)$. For the other direction, let $v \in pass_{XT''}(i,j)$. It is easy to see that $v \in pass_{XT}(f_\delta, j_0)$. To show that $v \in pass_{XT}(i_0, h)$, we consider two cases.

The first case is that $h \neq f_\delta$. Then, by property **(U1)**, f_δ is not a node of degree 1 in T_v, hence (h', f_δ) is in T_v, where h' is the last middle node on the path from i_0 to f_δ. In T'', h and h' are identified (or equal), and so there is an arc (i, h'') in T^*_v with $h'' \neq j$. (Recall that XT^* is the tree-decomposition after Steps 1 – 4 of the operation, described above. h'' is either the result of identifying h and h' or $h'' = h = h'$.) Since (i, h'') is not in T''_v (because $v \in pass_{XT''}(i,j)$), v is removed

from $X''_{h''}$ in Step 5 of the procedure, so v belongs to the mark of the path from i_0 to f_δ, hence $v \in pass_{XT}(i_0, h)$.

The second case is that $h = f_\delta$. Since f_δ is not a node of degree 1 in T_v, (i_0, h) is in T_v. From this it easily follows that $v \in pass_{XT}(i_0, h)$.

We conclude that $pass_{XT''}(i, j) = pass_{XT}(i_0, h) \cap pass_{XT}(f_\delta, j_0)$. Now, since f_δ is final, $pass_{XT}(f_\delta, j_0) \neq pass_{XT}(i_0, h)$. Since (i_0, h) is maximal, $|pass_{XT}(f_\delta, j_0)| \leq |pass_{XT}(i_0, h)|$. Consequently, $pass_{XT}(i_0, h) \cap pass_{XT}(f_\delta, j_0)$ is a proper subset of $pass_{XT}(i_0, h)$, and hence $|pass_{XT''}(i, j)| < |pass_{XT}(i_0, h)| = maxp(XT)$. This proves the desired property.

Finally, we show that property **(U1)** holds for XT''. As this proof is very similar to the proofs given above, we just consider a few cases, and leave the remaining cases to the reader. Consider a path from an initial node i_0 to a final node $i_{\beta+1}$, let $i_1, \ldots, i_\beta$ be the middle nodes on the path, and let π be the mark of the path.

Let $1 \leq \alpha \leq \beta$ with $\alpha + 1 \leq \beta - \alpha$, and let i be the node of T'' that is the result of identifying middle nodes i_α and $i_{\beta-\alpha+1}$. Assume now that i is a useless leaf of T''_v (of degree 1). Then either $(i_{\alpha-1}, i_\alpha)$ or $(i_{\beta-\alpha+1}, i_{\beta-\alpha+2})$ is in T_v. All arcs (i_α, j) are not in T_v, arc $(i_{\beta-\alpha}, i_{\beta-\alpha+1})$ is not in T_v, and all arcs $(i_{\beta-\alpha+1}, h)$ with $h \neq i_{\beta-\alpha+2}$ are not in T_v. Hence, if $(i_{\alpha-1}, i_\alpha)$ is in T_v, then i_α is a useless leaf of T_v, and if $(i_{\beta-\alpha+1}, i_{\beta-\alpha+2})$ is in T_v, then $i_{\beta-\alpha+1}$ is a useless root of rank 1 of T_v. This contradicts property **(U1)** of XT.

As a second and last case, we assume $\beta = 2\gamma - 1$, and consider the middle node i_γ which is half-way the path from i_1 to i_β. Then i_γ is also a node of T''. Assume that i_γ is a useless leaf of T''_v. Note that this implies that $v \notin \pi$. Either $(i_{\gamma-1}, i_\gamma)$ or $(i_\gamma, i_{\gamma+1})$ is in T_v, and all arcs (i_γ, j) with $j \neq i_{\gamma+1}$ are not in T_v. If $(i_\gamma, i_{\gamma+1})$ is in T_v, then $v \in pass_{XT}(i_\gamma, i_{\gamma+1}) = \pi$, a contradiction. If $(i_\gamma, i_{\gamma+1})$ is not in T_v, then $(i_{\gamma-1}, i_\gamma)$ is in T_v, and i_γ is a useless leaf of T_v, contradicting property **(U1)** of XT. This ends the proof of Proposition 6 □

Now we can prove Lemma 5. First use the construction from Proposition 6. Apply the following operation to XT'': identify each node i_0 in T'' that is at even distance to the root with all its children $i_1, \ldots, i_s$ $(0 \leq s \leq r'')$, see Figure 3. If j is the resulting node of T', then $X'_j = X''_{i_0} \cup X''_{i_1} \cup \cdots \cup X''_{i_s}$, and $\psi'(j) = \psi''(i_0) \cup \psi''(i_1) \cup \cdots \cup \psi''(i_s)$. Let $XT' = (\{X'_i \mid i \in I'\}, T' = (I', F'))$ be the resulting tree-decomposition. The width k' of XT' is at most $(r'' + 1)(k'' + 1) - 1$, and the rank r' of T' is at most r''^2.

It remains to show that $maxp(XT') < maxp(XT)$, i.e., that $|pass_{XT'}(i, j)| < maxp(XT)$ for every arc (i, j) of T'.

Consider an arc (i, j) of T'. Let i be the result of identifying a parent i_0 with its children $i_1, i_2, \ldots, i_s$, and let j be the result of identifying parent j_0 with its children $j_1, \ldots, j_{s'}$. Since (i, j) is an arc in T', there must be an arc (i_α, j_0) in T'' for some α, $1 \leq \alpha \leq s$.

We now claim that $pass_{XT'}(i, j) = pass_{XT''}(i_0, i_\alpha) \cap pass_{XT''}(i_\alpha, j_0)$. The inclusion $pass_{XT''}(i_0, i_\alpha) \cap pass_{XT''}(i_\alpha, j_0) \subseteq pass_{XT'}(i, j)$ is obvious. For the other

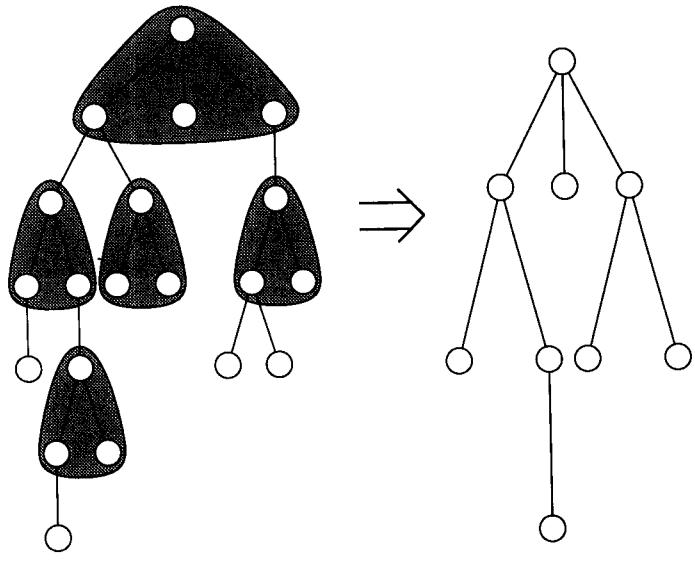

Figure 3: Example of operation in the proof of Lemma 5

direction, let $v \in pass_{XT'}(i,j)$. It is easy to see that $v \in pass_{XT''}(i_\alpha, j_0)$. As XT'' has property **(U1)**, this implies that i_0 is a node in T''_v, i.e., that $v \in X''_{i_0}$. Since i is useless in T'_v, i_0 is useless in T''_v. Suppose that i_0 is not of rank 1 in T''_v. Then, for some $\beta \neq \alpha$, i_β is a node of T''_v. Since $v \in chain_{XT'}(i)$, i_β must be a useless leaf of T''_v, contradicting property **(U1)**. Hence $v \in pass_{XT''}(i_0, i_\alpha)$.

From the equality $pass_{XT'}(i,j) = pass_{XT''}(i_0,i_\alpha) \cap pass_{XT''}(i_\alpha, j_0)$, we can conclude that $|pass_{XT'}(i,j)| \leq maxp(XT)$ as follows. If $|pass_{XT''}(i_0,i_\alpha)| < maxp(XT)$, then $|pass_{XT'}(i,j)| \leq |pass_{XT''}(i_0,i_\alpha)| < maxp(XT)$. Otherwise, by Proposition 6, $maxp(XT) > |pass_{XT''}(i_\alpha, j_0)| \geq |pass_{XT'}(i,j)|$.

In a similar way, one can show that XT' has property **(U1)**. □

Lemma 7 *For every k, $r \in \mathbf{N}$, there exist k', $r' \in \mathbf{N}$, such that for every graph $G = (V,E)$ and for every tree-decomposition XT of G of width $\leq k$, with the rank of $T \leq r$, there exists a tree-decomposition $XT' = (\{X'_i \mid i \in I'\}, T' = (I', F'))$ of G of width $\leq k'$ with the rank of $T' \leq r'$, such that for every vertex $v \in V$, if $size(T'_v) \geq 2$, then T'_v has no useless nodes of rank 1 or 0. Moreover the height and size of T' are at most the height and size of T, respectively.*

Proof: First we apply Lemma 4. Let XT'' be the resulting tree-decomposition. Note that $maxp(XT'') \leq k+1$. By applying Lemma 5 at most $k+1$ times, we obtain a tree-decomposition XT' of bounded width, and of bounded rank, such that $maxp(XT') = 0$. This implies that $chain_{XT'}(i) = \emptyset$, for every node i in T', and hence that for every vertex $v \in V$, if $size(T'_v) \geq 2$, then T'_v has no useless nodes of rank 1 or 0. □

Lemma 8 *For every k, r, $d \in \mathbf{N}$, there exist k', $r' \in \mathbf{N}$, such that for every graph $G = (V, E)$ with maximum degree $\leq d$ and for every tree-decomposition XT of G of width $\leq k$, with the rank of $T \leq r$, there exists a tree-decomposition $XT' = (\{X'_i \mid i \in I'\}, T' = (I', F'))$ of G of width $\leq k'$ with the rank of $T' \leq r'$, such that for every vertex $v \in V$, T'_v has height at most one. Moreover the height and size of T' are at most the height and size of T, respectively.*

Proof: First, we apply Lemma 7. Let XT' be the resulting tree-decomposition with width $\leq k'$ and rank of T' at most r'.

Note that every tree T'_v contains at most $deg(v)$ useful nodes, so at most $deg(v)$ nodes of rank 1 or 0. It follows that each T'_v contains at most $2d-1$ nodes.

Now apply the following operation to XT': let i_0 be the root of T'. Identify all trees T'_v with $v \in X'_{i_0}$, i.e., we have one node i'_0, with $X''_{i'_0} = \bigcup_{\{j \in T'_v \mid v \in X'_{i_0}\}} X'_j$. Repeat this procedure with all subtrees in the forest, obtained by removing the set $\{j \in T'_v \mid v \in X'_{i_0}\}$ from T'.

Let XT'' be the resulting tree-decomposition. The width of XT'' is at most $(2d-1)(k'+1)^2 - 1$, and the rank of T'' is at most $(2d-1)(k'+1)r'$, as we identify never more than $(2d-1)(k'+1)$ nodes.

Suppose j is a node in T''_v that is not the root of T''_v. Suppose i_0 is the root node in T' of the subtree that was contracted to j. It follows that $v \in X'_{i_0}$, hence all nodes i' in T' with $v \in X_{i'}$ that are a descendant of i_0 are contracted with i_0 to j. So j is a leaf in T''_v. As nodes in T''_v are either leaves or roots of T''_v, the height of T''_v is at most one. □

Theorem 9 *For every k, r, $d \in \mathbf{N}$, there exist k', $r' \in \mathbf{N}$, such that for every graph $G = (V, E)$ with maximum degree $\leq d$ and for every tree-decomposition XT of G of width $\leq k$, with the rank of $T \leq r$, there exists a domino tree-decomposition $XT' = (\{X'_i \mid i \in I'\}, T' = (I', F'))$ of G of width $\leq k'$ with the rank of $T' \leq r'$. Moreover the height and size of T' are at most the height and size of T, respectively.*

Proof: First apply Lemma 8. Let XT' be the resulting tree-decomposition with width $\leq k'$ and rank $\leq r'$. Then apply the following operation: identify all nodes that have a common parent in T'. Let XT'' be the resulting tree-decomposition, see Figure 4. The rank of T'' is at most r'^2, and the width of XT'' is at most $r' \cdot (k'+1) - 1$. Moreover, each T''_v consists of at most two nodes. □

Corollary 10 *For every k, $d \in \mathbf{N}$, there exists $k' \in \mathbf{N}$, such that every graph with treewidth at most k and maximum degree at most d has domino treewidth at most k'. Moreover, given a graph $G = (V, E)$ with treewidth at most k and maximum degree at most d, one can find a domino tree-decomposition of width at most k' in $O(|V|)$ time.*

Proof: It is well known that a graph with treewidth k has a tree-decomposition of width k and of rank 2. The first part of the result now follows directly from Theorem 9. One can find a tree-decomposition of G with width $\leq k$ and with $O(|V|)$ nodes

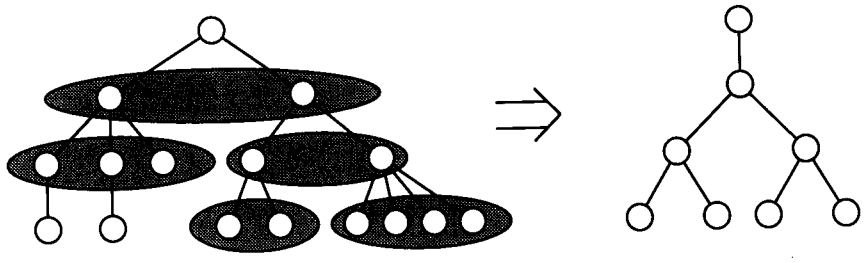

Figure 4: Example of operation in the proof of Theorem 9

in $O(|V|)$ time [4]. It is easy to transform this tree-decomposition into one with the same width and of rank 2 in $O(|V|)$ time. The constructions, described in the proofs of Lemmas 4, 5, 7, 8, and Theorem 9 can be carried out in $O(|V|)$ time in total. (Each of the described operations costs $O(|I|) = O(|V|)$ time, and at most $2k + 5$ such operations are carried out.) □

Inspecting the proofs in this section, it can be seen that k' is double exponential in k. We do not know whether this is optimal.

Corollary 11 *For every k, $d \in \mathbf{N}$, there exists $k' \in \mathbf{N}$, such that every graph $G = (V, E)$ with treewidth at most k and maximum degree at most d has a domino tree-decomposition of width at most k' and with the height of the tree $O(\log |V|)$.*

Proof: Using the algorithm of [16], one obtains a tree-decomposition of G with width $\leq 6(k + 1)$, and with the height of the tree $O(\log |V|)$. Observing this algorithm, it follows directly that the rank of the tree is bounded by the maximum number of components of a graph $G[V - S]$ with $|S| \leq 4(k + 1)$, which is at most $4(k+1)d$, assuming that G is connected. (Otherwise, apply the algorithm separately for every connected component.) So now we can apply Theorem 9. □

Corollary 12 *For every class of graphs $\mathcal{G}$, the following statements are equivalent:*

1. *There exists a constant $c \in \mathbf{N}$, such that every graph in $\mathcal{G}$ has domino treewidth at most c.*
2. *There exist constants k, $d \in \mathbf{N}$, such that every graph in $\mathcal{G}$ has treewidth at most k and maximum degree at most d.*

There is also a connection with the notion of *strong treewidth*, as introduced by Seese [18].

Definition. A *strong tree-decomposition* of a graph $G = (V, E)$ is a pair $(\{X_i \mid i \in I\}, T = (I, F))$ with $\{X_i \mid i \in I\}$ a collection of *disjoint* subsets of V, and $T = (I, F)$ a tree, such that

- $\bigcup_{i \in I} X_i = V$
- for all edges $\{v, w\} \in E$, either there is an $i \in I$ with $v, w \in X_i$, or there are $i, i' \in I$, that are adjacent in T ($(i, i') \in F$), and $v \in X_i$, $w \in X_{i'}$.

The width of a strong tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ is $\max_{i \in I} |X_i|$. The strong treewidth of a graph $G = (V, E)$ is the minimum width over all strong tree-decompositions of G.

Note that in general, a strong tree-decomposition of a graph G, is *not* a tree-decomposition of G. Note also that every tree has strong treewidth 1 (take $\{X_i \mid i \in I\}$ to consist of all singleton vertex sets).

As observed in [18], every graph of strong treewidth $\leq k$ is of treewidth $\leq 2k-1$. However, there is a set of graphs of treewidth 2, that is of unbounded strong treewidth. As an example, consider the set of all paths with one additional vertex that is adjacent to all vertices of the path. It is not difficult to see that if such a graph has a strong tree-decomposition of width k, then the tree has height at most 1, and the graph has at most $k(k+1)$ vertices. A similar example is the set of all 'wheels'. However, if a graph has bounded treewidth and bounded degree, then it is also of bounded strong treewidth.

In fact, a domino tree-decomposition of width $\leq k$ can easily be turned into a strong tree-decomposition of width $\leq k+1$: if T_v is a tree with two nodes i and j with j the child of i, then remove v from X_j. It is straightforward to check that this procedure gives a strong tree-decomposition. Thus, the following extension of Corollary 12 follows.

Corollary 13 *For every class of graphs $\mathcal{G}$, the following statements are equivalent:*

1. *There exists a constant $c \in \mathbf{N}$, such that every graph in $\mathcal{G}$ has domino treewidth at most c.*
2. *There exist constants k, $d \in \mathbf{N}$, such that every graph in $\mathcal{G}$ has treewidth at most k and maximum degree at most d.*
3. *There exist constants k', $d \in \mathbf{N}$, such that every graph in $\mathcal{G}$ has strong treewidth at most k' and maximum degree at most d.*

4 Algorithms for determining domino treewidth

4.1 Domino treewidth of trees

It is easy to see that every tree T has domino treewidth $\leq d-1$, where d is the maximum degree of T; thus, its domino treewidth lies between $\frac{1}{2}d$ and $d-1$.

In this subsection we show that there exists an $O(kn)$ time algorithm to determine whether the domino treewidth of a given tree is at most k. We use dynamic programming to do this.

Let $T = (V, E)$ be a tree. Choose an arbitrary vertex $r \in V$, and consider T as a rooted tree with r as root. Denote the subtree of T, formed by a vertex v and all its descendants by $T(v) = (V_v, E_v)$.

For each vertex $v \in V$, let $W_k(v)$ be defined as the minimum, over all domino tree-decompositions $(\{X_i \mid i \in I\}, \mathcal{T} = (I, F))$ of $T(v)$ of width at most k, of the minimum size of a set X_i with $v \in X_i$. $W_k(v)$ is ∞, if there does not exist a domino tree-decomposition of $T(v)$ with width at most k.

To test whether T has domino treewidth at most k, we compute for all $v \in V$, $W_k(v)$, in a bottom-up order. Clearly, the domino treewidth of T is at most k, if and only if $W_k(r) \neq \infty$.

Suppose $k \geq 1$. For a leaf vertex v, one can observe directly that $W_k(v) = 1$.

We now describe how we can compute $W_k(v)$ of a vertex v, given the values of $W_k(w_1), \ldots, W_k(w_p)$, for the children $w_1, \ldots, w_p$ of v.

Proposition 14 $W_k(v) = \min\{1 + \sum_{j \in S} W_k(w_j) \mid S \subseteq \{1, \ldots, p\}, \sum_{j \in S} W_k(w_j) \leq k \wedge \sum_{j \in \{1,\ldots,p\} - S} W_k(w_j) \leq k\}$.

Proof: Consider a set $S \subseteq \{1, \ldots, p\}$, such that $\sum_{j \in S} W_k(w_j) \leq k$ and $\sum_{j \in \{1,\ldots,p\} - S} W_k(w_j) \leq k$. For j, $1 \leq j \leq p$, take a domino tree-decomposition of $T(w_j)$ with a set X_{h_j} containing w_j and of size $W_k(w_j)$. Now, merge these domino tree-decompositions in the following way: take two new nodes i_0, i_1 with an arc between them. Put v in X_{i_0} and in X_{i_1}. Identify all nodes h_j, with $j \in S$, with i_0 (so, $X_{i_0} = \{v\} \cup \bigcup_{j \in S} X_{h_j}$). Identify all nodes h_j, with $j \in \{1, \ldots, p\} - S$, with i_1 (so, $X_{i_1} = \{v\} \cup \bigcup_{j \in \{1,\ldots,p\} - S} X_{h_j}$). The resulting domino tree-decomposition is a tree-decomposition of $T(v)$, with width at most k, and with a set, containing v of size $1 + \sum_{j \in S} W_k(w_j)$.

Now, suppose we have a domino tree-decomposition $(\{X_i \mid i \in I\}, \mathcal{T} = (I, F))$ of $T(v)$ of width at most k with the minimum size of a set X_i with $v \in X_i$. There must be at most 2 nodes, say i_1 and i_2 of which the sets contain v. (If there is only one such node, add a new node i' with $X_{i'} = \{v\}$.) Let $S = \{j \mid w_j \in X_{i_1}\}$. As $(\{X_i \cap V_{w_j} \mid i \in I\}, \mathcal{T} = (I, F))$ is a domino tree-decomposition of $T(w_j)$ of width at most k, it follows that X_{i_1} contains at least $W_k(w_j)$ vertices from $T(w_j)$ for each $j \in S$. So $|X_{i_1}| \geq 1 + \sum_{j \in S} W_k(w_j)$. Since $X_{i_1} \leq k + 1$, this implies that $\sum_{j \in S} W_k(w_j) \leq k$. With similar arguments one can show that $|X_{i_2}| \geq 1 + \sum_{j \in \{1,\ldots,p\} - S} W_k(w_j)$ and that $\sum_{j \in \{1,\ldots,p\} - S} W_k(w_j) \leq k$. □

Finding the set S which achieves the minimum value, as described in Proposition 14 above, when given the values $W_k(w_j)$ for all children of v, corresponds to an instance of a variant of the KNAPSACK problem, and can be solved by a standard dynamic programming algorithm in $O(p \cdot k)$ time. (See e.g., [15], Chapter 5.) So, $W_k(v)$ can be computed in $O(deg(v) \cdot k)$ time, given the values $W_k(w_j)$ for all children of v.

The total time to compute $W_k(r)$ hence is $O(\sum_{v \in V} deg(v) \cdot k) = O(k \cdot |V|)$: compute successively all values $W_k(v)$, in a bottom-up manner in the tree.

Theorem 15 *There exists an $O(kn)$ algorithm to compute whether the domino treewidth of a tree with n vertices is at most k.*

The algorithm can output a corresponding domino tree-decomposition within the same time bounds.

To find the domino treewidth of a given tree T, one can do binary search on the value of k, and use the procedure described above. This approach costs, in the worst case, $O(n^2 \log n)$ time. We conjecture that some improvements will be possible to this bound.

4.2 Fixed parameter algorithms

In this section, we show that the problem whether the domino treewidth of a given graph G is at most k, for constant k, is solvable in polynomial time. Our algorithm has a similar structure as the $O(n^{k+2})$ algorithm from Arnborg et al [2] to recognize graphs with treewidth at most k. The additional technicalities are involved.

For $R \subseteq V$, $G[R] = (R, \{\{v,w\} \in E \mid v,w \in R\})$ denotes the subgraph of $G = (V,E)$, induced by R.

For sets $S \subseteq V$, $B \subseteq S$, $D \subseteq \{\{v,w\} \in E \mid v \in S\}$, define

$R(S,D) = \{v \in V |$ there exists a path from v to a vertex in S that does not use edges in $D\}$

$P_k(S,D,B) \Leftrightarrow$ there exists a domino tree-decomposition $(\{X_i \mid i \in I\}, T = (I,F))$ of $G[R(S,D)]$ with width at most k, such that there exists an $i_0 \in I$ with $X_{i_0} = S$ and for all $i \in I : i \neq i_0 \Rightarrow X_i \cap B = \emptyset$.

Note that if $v \in R(S,D) - S$, then v can only be adjacent to vertices in $R(S,D)$. From now on we assume (w.l.o.g.) that the node sets of a tree-decomposition are non-empty.

Lemma 16 *Let $G = (V,E)$ be a connected graph. The domino treewidth of G is at most k, if and only if there exists a set $S \subseteq V$ with $S \neq \emptyset$, $|S| \leq k+1$, and $P_k(S,\emptyset,\emptyset)$.*

Proof: $\Rightarrow$: Take a domino tree-decomposition $(\{X_i \mid i \in I\}, T = (I,F))$ of G with width at most k, and take $S = X_i$ for an arbitrary $i \in I$.

$\Leftarrow$: Note that $R(S,\emptyset) = V$. The result now follows from the definition. □

Note that $P_k(S,D,B)$ is true if $|S| \leq k+1$ and $R(S,D) = S$.

Lemma 17 *Let $G = (V,E)$ be a connected graph. Let $S \subseteq V$, $B \subseteq S$, $D \subseteq \{\{v,w\} \in E \mid v \in S\}$, $S \neq \emptyset$, $|S| \leq k+1$, and $R(S,D) \neq S$. $P_k(S,D,B)$ is true, if and only if at least one of the following two properties holds:*

1. $\exists S' \subseteq R(S,D) : \exists D' \subseteq \{\{v,w\} \in E \mid v \in S'\}$:

   - $|S'| \leq k+1$
   - $S' \neq \emptyset$
   - $B \cap S' = \emptyset$
   - $\neg\exists v \in S' - S : \exists w \in S - S' : \{v,w\} \in E$
   - $P_k(S', D', S \cap S')$
   - $R(S,D) = R(S',D') \cup (S - S')$
   - $(S - S') \cap R(S',D') = \emptyset$
   - $|R(S',D')| < |R(S,D)|$ *or* $(B = \emptyset$ *and* $S \cap S' \neq \emptyset)$

2. $\exists D_1 \subseteq \{\{v,w\} \in E \mid v \in S\}$: $\exists D_2 \subseteq \{\{v,w\} \in E \mid v \in S\}$: $\exists B_1 \subseteq S$: $\exists B_2 \subseteq S$:

   - $R(S,D_1) \cup R(S,D_2) = R(S,D)$
   - $R(S,D_1) \cap R(S,D_2) = S$
   - $|R(S,D_1)| < |R(S,D)|$
   - $|R(S,D_2)| < |R(S,D)|$
   - $P_k(S,D_1,B_1)$
   - $P_k(S,D_2,B_2)$
   - $S = B_1 \cup B_2$
   - $B \subseteq B_1 \cup B_2$

Proof: $\Rightarrow$: Suppose $P_k(S,D,B)$ holds. Let $(\{X_i \mid i \in I\}, T = (I,F))$ be a domino tree-decomposition of $G[R(S,D)]$ with width at most k, and let $i_0 \in I$ such that $X_{i_0} = S$ and for all $i \neq i_0$: $X_i \cap B = \emptyset$.

Note that if $v \in \bigcup_{i \in I} X_i - S$, then v can only be adjacent to vertices in $\bigcup_{i \in I} X_i$: if $\{v,w\} \in E$ and $v \in R(S,D) - S$, then we have $w \in R(S,D)$.

Let r be the number of neighbors of i_0 in T. As $R(S,D) \neq S$, we have that $r > 0$. We consider two cases.

Case 1.

Suppose $r = 1$. Let i_1 be the unique neighbor of i_0 in T. We will show that the first property, mentioned in the lemma holds.

Take $S' = X_{i_1}$. Write $W = \bigcup_{i \in I - \{i_0\}} X_i = R(S,D) - (S - S')$. Take $D' = \{\{v,w\} \in E \mid v \in S', w \notin W\}$.

Proposition 18 $R(S',D') = W$.

Proof: We first observe that vertices in $W - S'$ can only be adjacent to vertices in W. In fact, let $v \in W - S'$ and $\{v, w\} \in E$. Since the tree-decomposition is domino, $v \notin S$ and so $w \in R(S, D)$. Using the definition of tree-decomposition, it follows that $w \notin S - S'$. Hence $w \in W$.

Suppose $v \in R(S', D') - W$. There exists a path from v to a vertex in S', avoiding edges in D', say $v_0, v_1, \ldots, v_s$, $v = v_0$, $v_s \in S'$. Let v_j be the last vertex on the path that does not belong to W. $j \neq s$, because $S' \subseteq W$. Now $\{v_j, v_{j+1}\} \in E$, $v_j \notin W$, $v_{j+1} \in W$. Using the above observation, it follows that v_{j+1} must belong to S', which contradicts the fact that $\{v_j, v_{j+1}\} \notin D'$. So, $R(S', D') \subseteq W$.

Suppose $v \in W$. As G is connected, there is a path in G from v to a vertex in S'. Let s be the first vertex in S' on this path. By the above observation, the path up to s does not use edges in D', so $v \in R(S', D')$. □

From the proposition, it follows directly that $(S - S') \cap R(S', D') = \emptyset$, and that $R(S, D) = R(S', D') \cup (S - S')$. Clearly, $|S'| \leq k + 1$, $S' \neq \emptyset$.

$P_k(S', D', S \cap S')$ holds: consider the domino tree-decomposition $(\{X_i \mid i \in I - \{i_0\}\}, T[I - \{i_0\}])$.

If $v \in S' - S$ and $w \in S - S'$, there is no X_i containing both v and w, so by definition of tree-decomposition, $\{v, w\} \notin E$.

Finally, we consider two subcases:

Case 1.1. $S \subseteq S'$. Now $R(S', D') = R(S, D)$, $B = \emptyset$, and $S \cap S' = S \neq \emptyset$.

Case 1.2. $S \not\subseteq S'$. Clearly, now $|R(S', D')| < |R(S, D)|$ holds.

Case 2.

Suppose $r \geq 2$. Let $i_1, \ldots, i_r$ be the neighbors of i_0, and let $T_1 = (I_1, F_1), \ldots, T_r = (I_r, F_r)$ be the subtrees of $T[I - \{i_0\}]$, such that T_j contains the node i_j. Write $I' = I_1 \cup \cdots \cup I_{r-1}$, $I'' = I_r$, $W_1 = \bigcup_{i \in I'} X_i \cup X_{i_0}$, $W_2 = \bigcup_{i \in I''} X_i \cup X_{i_0}$. Using the properties of tree-decompositions, we have that $W_1 \cap W_2 = X_{i_0}$. Vertices in $W_1 - S$ can only be adjacent to vertices in W_1; vertices in $W_2 - S$ can only be adjacent to vertices in W_2.

Now take:

$$\begin{aligned} D_1 &= \{\{v, w\} \in E \mid v \in S, w \notin W_1\} \\ D_2 &= \{\{v, w\} \in E \mid v \in S, w \notin W_2\} \\ B_1 &= \{v \in S \mid \forall i \in I' : v \notin X_i\} \\ B_2 &= \{v \in S \mid \forall i \in I'' : v \notin X_i\} \end{aligned}$$

Similar as was done in Case 1 of this proof, one can show that $W_1 = R(S, D_1)$ and $W_2 = R(S, D_2)$. Now $R(S, D) = W_1 \cup W_2 = R(S, D_1) \cup R(S, D_2)$. $R(S, D_1) \cap$

$R(S, D_2) = W_1 \cap W_2 = X_{i_0} = S$. As $\bigcup_{i \in I'} X_i \neq \emptyset$ and $\bigcup_{i \in I''} X_i \neq \emptyset$, we have $R(S, D) - R(S, D_1) = \bigcup_{i \in I''} X_i \neq \emptyset$, and $R(S, D) - R(S, D_2) \neq \emptyset$.

$P_k(S, D_1, B_1)$ holds: look at the domino tree-decomposition $(\{X_i \mid i \in I' \cup \{i_0\}\}, T[I' \cup \{i_0\}])$. By definition of B_1, $X_i \cap B_1 = \emptyset$ for all $i \in I'$. In the same way it follows that $P_k(S, D_2, B_2)$ holds.

If $v \in S - B_1$, then there exists an $i' \in I'$ with $v \in X_{i'}$. When there exists also an $i'' \in I''$ with $v \in X_{i''}$, then v would belong to the three sets X_{i_0}, $X_{i'}$, $X_{i''}$. It follows that $v \in S - B_1$ implies $v \in B_2$. Hence $S = B_1 \cup B_2$.

Finally, it is clear that when $v \in B$, then also $v \in B_1$ and $v \in B_2$, so $B \subseteq B_1 \cap B_2$.

This ends the analysis of Case 2, and of the $\Rightarrow$ part of the proof.

$\Leftarrow$: First suppose we have sets S', D', fulfilling the first property mentioned in the lemma. Write $B' = S \cap S'$. There exists a domino tree-decomposition $(\{X_i \mid i \in I\}, T = (I, F))$ with width at most k of $G[R(S', D')]$ with for some $i_1 \in I$: $X_{i_1} = S'$ and for all $i \neq i_1$, $X_i \cap B' = \emptyset$.

Take a new node $i_0 \notin I$, and take $X_{i_0} = S$. Let T' be the tree, obtained by making i_0 adjacent to i_1 in T. We claim that $(\{X_i \mid i \in I \cup \{i_0\}\}, T')$ is a domino tree-decomposition for $G[R(S, D)]$ with $X_{i_0} = S$, and for all $i \neq i_0$, $X_i \cap B = \emptyset$. Clearly, the width of this domino tree-decomposition is at most k.

First, $\bigcup_{i \in I \cup \{i_0\}} X_i = \bigcup_{i \in I} X_i \cup X_{i_0} = R(S', D') \cup S = R(S', D') \cup (S - S') = R(S, D)$. Secondly, if $\{v, w\} \in E$, $v, w \in R(S, D)$, then

- if $v, w \in R(S', D') : \exists i \in I : v, w \in X_i$.
- if $v \in R(S', D')$, $w \in S - S'$, then $v \in S'$. Also, by assumption, $v \notin S' - S$. So $v, w \in X_{i_0}$.
- if $v, w \in S - S'$: $v, w \in X_{i_0}$.

Using that $S \cap R(S', D') \subseteq S'$, one easily verifies the third condition of tree-decomposition.

Now we check dominoness. Vertices in $S - S'$ only belong to X_{i_0} and to no other set X_i, $i \in I$. Vertices in $S \cap S'$ belong to X_{i_0} and X_{i_1}, but, by assumption, not to any set X_i, $i \in I - \{i_1\}$. Clearly, all vertices in $R(S, D) - S$ belong to at most two sets X_i, $i \in I$, and not to X_{i_0}.

Finally, for all $v \in B$, and for all $i \in I \cup \{i_0\}$: if $v \in X_i$, then $v \notin S'$, $v \in S$, so $v \notin R(S', D')$, hence $i = i_0$. This ends the analysis of the case that the first property holds.

Now suppose we have sets D_1, D_2, B_1, B_2 fulfilling the second property mentioned in the lemma. Take domino tree-decompositions $(\{X_i \mid i \in I_1\}, T_1 = (I_1, F_1))$ of $G[R(S, D_1)]$ and $(\{Y_i \mid i \in I_2\}, T_2 = (I_2, F_2))$ of $G[R(S, D_2)]$, with width at most k and with for some $i_1 \in I_1$, $i_2 \in I_2$: $X_{i_1} = Y_{i_2} = S$, for $i \in I_1 - \{i_1\}$: $X_i \cap B_1 = \emptyset$; for $i \in I_2 - \{i_2\}$: $X_i \cap B_2 = \emptyset$.

Let T be obtained by taking the union of T_1 and T_2 and identifying i_1 and i_2. Write for $i \in I_1$: $Z_i = X_i$, and for $i \in I_2$: $Z_i = Y_i$.

Consider the tree-decomposition $(\{Z_i \mid i \in I_1 \cup I_2\}, T)$. We claim that this is a domino tree-decomposition of $G[R(S,D_1) \cup R(S,D_2)] = G[R(S,D)]$ with width at most k, such that $Z_{i_1} = S$, and for all $i \in I_1 \cup I_2 - \{i_1\}$: $B \cap Z_i = \emptyset$. The latter condition follows from earlier made assumptions, as $B \subseteq B_1 \cap B_2$. It is straightforward to verify that we indeed have a tree-decomposition of width at most k, using that $R(S,D_1) \cup R(S,D_2) = R(S,D)$, and $R(S,D_1) \cap R(S,D_2) = S$.

Dominoness also holds: for $v \notin S$, it follows directly that v belongs to at most two sets Z_i, $i \in I_1 \cup I_2$. If $v \in B_2$, then v does not belong to a set Z_i, $i \in I_2 - \{i_2\}$. So v belongs to at most two sets Z_i, $i \in I_1 \cup I_2$, both with $i \in I_1$. A similar analysis applies when $v \in B_1$. This ends the analysis of the case that the second property holds. □

Theorem 19 *For fixed k, there exists an algorithm, that checks whether a given graph G with n vertices has domino treewidth at most k in $O(n^{2k+3})$ time.*

Proof: Since it clearly suffices to consider connected components, we may assume that G is connected. First, we compute for all $S \subseteq V$ with $|S| \leq k+1$, $D \subseteq \{\{v,w\} \in E \mid v \in S\}$, the set $R(S,D)$. Then, we sort all pairs (S,D) in order of increasing size of $R(S,D)$. In order of increasing size of $R(S,D)$, we will compute for all $B \subseteq S$, in order of decreasing size of B, the value of $P_k(S,D,B)$. For this, we use the result of Lemma 17. We check whether one of the properties holds. Note that all values of P_k, referred to in the characterization of Lemma 17 have been computed before they are needed. The first property can be checked by looking at all $O(n^{k+1})$ possible guesses for S', and all $O(1)$ guesses for D'. (This number only depends on the maximum degree of a vertex in G, and k; both may be assumed to be bounded by constants.) For each such guess, the property can be verified in $O(n)$ time. The second property can be checked similarly. It follows that the time per triple (S,D,B) to compute $P_k(S,D,B)$ is $O(n^{k+2})$. As we have $O(n^{k+1})$ triples (S,D,B) to look at, the total time becomes $O(n^{2k+3})$.

Finally, we look up whether for any S, $P_k(S,\emptyset,\emptyset)$ holds. (Cf. Lemma 16.) □

The algorithm can be modified such that it outputs — when existing — a domino tree-decomposition of the input graph with width at most k, and such that it still uses $O(n^{2k+3})$ time.

It might be possible to improve the running time somewhat.

5 Hardness results

In this section, we show that the domino treewidth problem is $W[t]$-hard for all $t \in \mathbf{N}$, where the notion of $W[t]$-hardness is taken from the work of Downey and Fellows (see [11, 12, 13, 1]). Also, we prove the problem to be NP-complete. Both results follow from the same transformation from LONGEST COMMON SUBSEQUENCE. To be precise, we establish $W[t]$-hardness for the following problem:

DOMINO TREEWIDTH
Instance: Graph $G = (V, E)$, integer k.
Parameter: k.
Question: Is the domino treewidth of G at most k?

The fact that domino treewidth is $W[t]$-hard for all $t \in \mathbf{N}$ (as in the theory on fixed parameter intractability, developed by Downey and Fellows), suggests strongly that it is impossible to find algorithms for the domino treewidth $\leq k$ problem that use $f(k)pol(n)$ time, i.e., where only the constant factor in the running time depends on k. (Note that such algorithms do exist for some related problems, like treewidth $\leq k$, pathwidth $\leq k$.) The classes $W[t]$ denote classes of *parameterized problems*: problems $Q \subseteq \{(k, s) \mid k \in \mathbf{N},\ s \in \Sigma^*\}$, for some alphabet Σ. To prove $W[t]$-hardness for a problem P, it suffices to find a $W[t]$-hard problem Q and functions $f, h : \mathbf{N} \to \mathbf{N}$, $g : \mathbf{N} \times \Sigma^* \to \Sigma^*$, such that for all $(k, s) \in \mathbf{N} \times \Sigma^*$: $(k, s) \in Q \Leftrightarrow (f(k), g(k, s)) \in P$, and f is any computable function, and g is computable in time $h(k) \cdot |s|^c$ for some constant c. (I.e., the time is polynomial in the length of the non-parameter part of the input, but may depend in any way on the parameter.)

The LONGEST COMMON SUBSEQUENCE problem is the following:

LONGEST COMMON SUBSEQUENCE
Instance: Alphabet Σ, strings $s^1, \ldots, s^r \in \Sigma^*$, integer $m \in \mathbf{N}$.
Parameter: r.
Question: Does there exist a string in Σ^* of length at least m, that is a subsequence of each string $s^1, \ldots, s^r$?

Very recently, it has been shown that LONGEST COMMON SUBSEQUENCE is $W[t]$-hard for all $t \in \mathbf{N}$ [7, 8].

Theorem 20 *For all $t \in \mathbf{N}$,* DOMINO TREEWIDTH *is $W[t]$-hard.*

Proof: We give a transformation from LONGEST COMMON SUBSEQUENCE.

Let an instance Σ, $s^1, \ldots, s^r$, m of LONGEST COMMON SUBSEQUENCE be given. We suppose we have a numbering of the characters in Σ, say $\Sigma = \{\sigma_1, \ldots, \sigma_l\}$.

Let $l = |\Sigma|$, $k = 20r^2 + 40r - 1$, $q = m(lr^2 + 1)$.

We now define a graph $G = (V, E)$, that consists of the following components:

Two anchors. Take two cliques, each with $k + 1$ vertices, with vertex sets $A_1 = \{a_i^1 \mid 1 \leq i \leq k + 1\}$ and $A_2 = \{a_i^2 \mid 1 \leq i \leq k + 1\}$.

The floor. Take q cliques of $\frac{2}{5}(k + 1) = 8r^2 + 16r$ vertices, $\{b_i^\alpha \mid 1 \leq i \leq \frac{2}{5}(k + 1),\ 1 \leq \alpha \leq q\}$. Take edges:

- $\{b_i^\alpha, b_j^\alpha\}$, $1 \leq \alpha \leq q$, $1 \leq i, j \leq \frac{2}{5}(k + 1)$, $i \neq j$.

- $\{b_i^\alpha, b_j^{\alpha+1}\}$, $1 \leq \alpha < q$, $1 \leq i, j \leq \frac{2}{5}(k+1)$.
- $\{a_i^1, b_j^1\}$, $1 \leq i, j \leq \frac{2}{5}(k+1)$.
- $\{a_i^2, b_j^q\}$, $1 \leq i, j \leq \frac{2}{5}(k+1)$.

The number $\frac{2}{5}$ is chosen because $2 \cdot \frac{2}{5} < 1$, and $3 \cdot \frac{2}{5} > 1$.

To ease presentation, we will denote vertices a_i^1 with $1 \leq i \leq \frac{2}{5}(k+1)$ also as b_i^0, and vertices a_i^2 with $1 \leq i \leq \frac{2}{5}(k+1)$ also as b_i^{q+1}.

The hills. We have m 'hills'. Take vertices $\{c_{i,j}^\alpha \mid 1 \leq \alpha \leq m,\ 1 \leq i \leq l \cdot r^2,\ 1 \leq j \leq \frac{1}{5}(k+1) - (2r+1)\}$. Take edges:

- $\{c_{i,j}^\alpha, b_1^{(lr^2+1)\cdot(\alpha-1)+i}\}$, $1 \leq \alpha \leq m$, $1 \leq i \leq l \cdot r^2$, $1 \leq j \leq \frac{1}{5}(k+1) - (2r+1)$.
- $\{c_{i,j}^\alpha, b_1^{(lr^2+1)\cdot(\alpha-1)+i+1}\}$, $1 \leq \alpha \leq m$, $1 \leq i \leq l \cdot r^2$, $1 \leq j \leq \frac{1}{5}(k+1) - (2r+1)$.

Each hill, together with the adjacent floor vertices, represents a character of the subsequence.

The string components. For each string s^i, $1 \leq i \leq r$, we have a string component. Below, we describe the different parts for the ith string component. Suppose $s^i = s_1^i s_2^i \cdots s_{l_i}^i$.

The string path. Take a path with $l_i \cdot (lr^2+1) + 2$ vertices, and attach it to a_1^1 and a_1^2: take vertices $\{d_j^i \mid 0 \leq j \leq l_i \cdot (lr^2+1) + 1\}$, edges $\{d_j^i, d_{j+1}^i\}$, $\{d_0^i, a_1^1\}$, $\{d_{l_i\cdot(lr^2+1)+1}^i, a_1^2\}$.

The blobs. We take $l_i + 1$ cliques of $2r+2$ vertices, and attach these to the string paths. Take vertices $\{e_{j,j'}^i \mid 0 \leq j \leq l_i,\ 1 \leq j' \leq 2r+2\}$. Take edges:

- $\{e_{j,j'}^i, e_{j,j''}^i\}$, $0 \leq j \leq l_i$, $1 \leq j', j'' \leq 2r+2$, $j' \neq j''$.
- $\{e_{j,j'}^i, d_{j\cdot(lr^2+1)}^i\}$, $0 \leq j \leq l_i$, $1 \leq j' \leq 2r+2$.
- $\{e_{j,j'}^i, d_{j\cdot(lr^2+1)+1}^i\}$, $0 \leq j \leq l_i$, $1 \leq j' \leq 2r+2$.

The idea is that blobs cannot come on top of hills. This forces a precise way how the part between blobs falls over a hill (if it does). Each such part represents a character in the string s^i; the lr^2+1 vertices in such a part are needed for symbol checking.

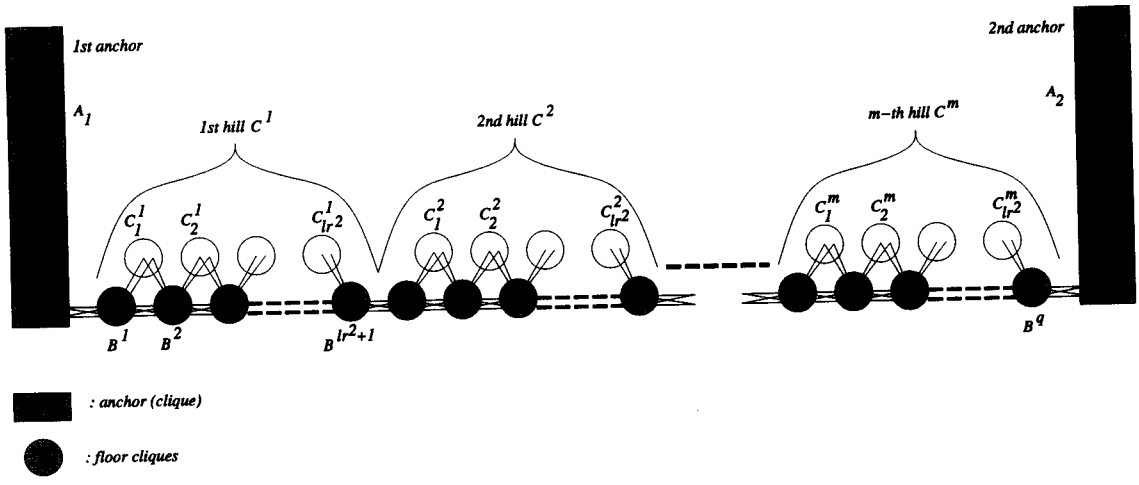

Figure 5: Anchors, floor, and hills.

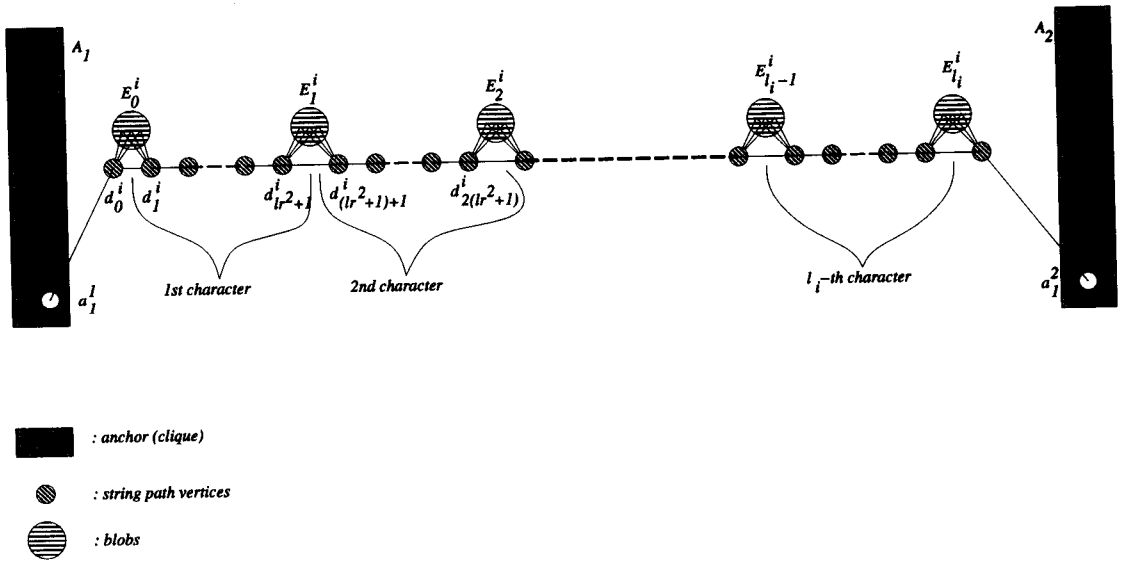

Figure 6: The i-th string component: string path, and blobs.

The symbol checkers. These vertices are used to check that the chosen characters for the different strings are the same. Take vertices $\{f^i_{\alpha,(i-1)rl+(j-1)l+t} \mid 1 \leq \alpha \leq l_i,\ 1 \leq j \leq r,\ \sigma_t = s^i_\alpha\} \cup \{f^i_{\alpha,(j-1)rl+(i-1)l+t} \mid 1 \leq \alpha \leq l_i,\ 1 \leq j \leq r,\ \sigma_t \neq s^i_\alpha\}$. If vertex $f^i_{\alpha,\beta}$ exists, make it adjacent to the vertices $d^i_{(\alpha-1)(lr^2+1)+\beta}$ and $d^i_{(\alpha-1)(lr^2+1)+\beta+1}$.

Let $G = (V, E)$ be the resulting graph. Parts of this construction are illustrated in Figures 5 and 6. We denote: $B^\alpha = \{b^\alpha_i \mid 1 \leq i \leq \frac{2}{5}(k+1)\}$, $C^\alpha = \{c^\alpha_{i,j} \mid 1 \leq i \leq l \cdot r^2,\ 1 \leq j \leq \frac{1}{5}(k-1) - (2r+1)\}$, $C^\alpha_i = \{c^\alpha_{i,j} \mid 1 \leq j \leq \frac{1}{5}(k-1) - (2r+1)\}$, and $E^i_j = \{e^i_{j,j'} \mid 1 \leq j' \leq 2r+2\}$.

Proposition 21 *The domino treewidth of G is at most k, if and only if $s^1, \ldots, s^r$ have a common subsequence of length at least m.*

Proof: $\Rightarrow$: Suppose $(\{X_i \mid i \in I\}, T = (I, F))$ is a domino tree-decomposition of G of width $\leq k$. First, note that there must be nodes h_0, h_γ with $A_1 \subseteq X_{h_0}$, $A_2 \subseteq X_{h_\gamma}$ (Lemma 3). Consider the path in T between h_0 and h_γ, and number the nodes on this path consecutively $h_0, h_1, h_2, \ldots, h_{\gamma-1}, h_\gamma$. In fact, as $|A_1| = |A_2| = k+1$, $A_1 = X_{h_0}$ and $A_2 = X_{h_\gamma}$. It follows that all vertices of the form a^1_i or b^1_j, $1 \leq i, j \leq \frac{2}{5}(k+1)$, must be element of X_{h_1}: these form a clique, which (by Lemma 3) must be contained in a node-set of a node, adjacent to h_0 (by dominoness), and by Lemma 1, it cannot be another node than h_1 (because there is a path in G from a vertex b^1_j to a vertex in A_2 that avoids vertices in A_1).

Now, we claim, with induction to α, that for all j, $1 \leq j \leq \frac{2}{5}(k+1)$, b^α_j and $b^{\alpha+1}_j$ are in $X_{h_{\alpha+1}}$. We know this holds for $\alpha = 0$. Let $0 < \alpha \leq q$. All nodes b^α_j and $b^{\alpha+1}_{j'}$ form together a clique. By Lemma 3, there is a node i', with $X_{i'}$ containing this clique. $i' \neq h_\alpha$, otherwise X_{h_α} contains too many vertices. Hence i' is a neighbor of h_α (by dominoness). If $i' \neq h_{\alpha+1}$, then note that there are $\frac{2}{5}(k+1)$ vertex disjoint paths from vertices in the set $\{b^{\alpha+1}_{j'} \mid 1 \leq j' \leq \frac{2}{5}(k+1)\}$ to vertices in the set A_2, that avoid vertices of the form $b^{\alpha-1}_j$ or b^α_j. By Lemma 1, this gives in total $\frac{6}{5}(k+1)$ vertices that belong to X_{h_α}, contradiction. So, all nodes of the form b^α_j and $b^{\alpha+1}_{j'}$ belong to $X_{h_{\alpha+1}}$. It follows that $\gamma = q+2$.

For each hill vertex $c^\alpha_{i,j}$, note that there is a unique node containing the neighbors of the vertex, namely $h_{(\alpha-1)(lr^2+1)+i+1}$, so $c^\alpha_{i,j} \in X_{h_{(\alpha-1)(lr^2+1)+i+1}}$. For each blob, there must be a node, containing all vertices in the blob (as it is a clique). However, such a node cannot be a node of the form $h_{(\alpha-1)(lr^2+1)+i+1}$, as the number of hill vertices plus the blob size plus the number of floor vertices that would belong to the node would be larger than $k+1$. Call nodes of this form *hill nodes*. Nodes of the form $h_{\alpha(lr^2+1)+1}$ are called *valley nodes*.

Note that each hill node also contains at least two vertices per string path (because of Lemma 2). These $2r$ vertices, with the hill vertices and the floor vertices give in total k vertices, so only one extra vertex is possible.

Consider a blob, with vertices $E^i_j = \{e^i_{j,j'} \mid 1 \leq j' \leq 2r+2\}$. There must be a node $i' \in I$, with $E^i_j \cup \{d_{j(lr^2+1)}, d_{j(lr^2+1)+1}\} \subseteq X_{i'}$ (Lemma 3). We have already

argued that i' is not a hill node. Suppose i'' is the first node on the path from i' to h_0 that is also on the path from h_0 to h_γ. We claim that i'' is a valley node. If i' is a valley node, then we are done, as in that case $i' = i''$. Suppose i' is not a valley node, hence i' is not on the path from h_0 to h_γ. It follows that $X_{i''}$ must contain at least four nodes of the ith string path (Lemma 2), and, hence, that i'' cannot be a hill node, so must be a valley node. Write $f(i,j) = \alpha$, if the node i'' as above is of the form $h_{\alpha(lr^2+1)+1}$. It is easy to see that α does not depend on the choice of i'.

Note that $f(i,j) \leq f(i,j+1)$; if not, then the string path between blob E^i_j and blob E^i_{j+1} must go back over a hill: that hill then gets too many string path vertices in its node sets. We also must have $f(i,0) = 0$, and $f(i,l_i) = m$. So, for each i, there are exactly m values $\delta^i_1, \delta^i_2, \ldots, \delta^i_m$, with $\delta^i_1 < \delta^i_2 < \cdots < \delta^i_m$, and for all p, $1 \leq p \leq m$: $f(i, \delta^i_p - 1) = p - 1$ and $f(i, \delta^i_p) = p$.

We now claim, that all subsequences $s^i_{\delta^i_1} s^i_{\delta^i_2} \cdots s^i_{\delta^i_m}$ are equal. Consider the two characters $s^i_{\delta^i_p} = \sigma_t$ and $s^j_{\delta^j_p} = \sigma_{t'}$. We must show these are equal.

Write $\rho = (p-1)(lr^2+1)+1$, and $\rho' = p(lr^2+1)+1$. h_ρ is the valley node, just before the pth hill, and $h_{\rho'}$ is the valley node just after the pth hill. X_{h_ρ} must contain a vertex $d^i_{(\delta^i_p-1)(lr^2+1)+1+\epsilon}$ for some $\epsilon \geq 0$. $X_{h_{\rho'}}$ must contain a vertex $d^i_{\delta^i_p(lr^2+1)-\epsilon'}$ for some $\epsilon' \geq 0$. Note that the distance in T between h_ρ and $h_{\rho'}$ equals the length of the path between these vertices for $\epsilon = \epsilon' = 0$, plus one. From dominoness, it follows, that for all β, $0 \leq \beta \leq lr^2$, we must have that $d^i_{(\delta^i_p-1)(lr^2+1)+1+\beta} \in X_{h_{\rho+\beta}} \cap X_{h_{\rho+\beta+1}}$. A similar analysis holds for the nodes on the jth string path.

In particular, we have that $i''' = h_{\rho+(i-1)rl+(j-1)l+t}$ is the unique node that contains $d^i_{(\delta^i_p-1)(lr^2+1)+(i-1)rl+(j-1)l+t}$ and $d^i_{(\delta^i_p-1)(lr^2+1)+(i-1)rl+(j-1)l+t+1}$, and is the unique node that contains $d^j_{(\delta^j_p-1)(lr^2+1)+(i-1)rl+(j-1)l+t}$ and $d^j_{(\delta^j_p-1)(lr^2+1)+(i-1)rl+(j-1)l+t+1}$. Both $f^i_{\delta^i_p,(i-1)rl+(j-1)l+t}$ and $f^j_{\delta^j_p,(i-1)rl+(j-1)l+t}$ when existing, must belong to i'''. As i''' is a hill node, they cannot exist both, otherwise $X_{i'''}$ contains too many vertices. As $f^i_{\delta^i_p,(i-1)rl+(j-1)l+t}$ exists (we assumed $\sigma_t = s^i_{\delta^i_p}$), we have that $f^j_{\delta^j_p,(i-1)rl+(j-1)l+t}$ does not exist, and hence that $s^j_{\delta^j_p} = \sigma_t = s^i_{\delta^i_p}$.

It follows that all subsequences $s^i_{\delta^i_1} s^i_{\delta^i_2} \cdots s^i_{\delta^i_m}$ are equal, hence the strings s^i, $1 \leq i \leq r$ have a common subsequence of length m.

$\Leftarrow$: Suppose subsequences $s^i_{\delta^i_1} s^i_{\delta^i_2} \cdots s^i_{\delta^i_m}$ of strings s^i are all equal. We show how to construct a domino tree-decomposition of G of width k. First, we take nodes h_0, $\ldots$, h_{q+2}, forming a path. Put all vertices of A_1 in X_{h_0}, all vertices of A_2 in $X_{h_{q+2}}$, and all vertices of the form b^α_i ($0 \leq \alpha \leq q+1$, $1 \leq i \leq \frac{2}{5}(k+1)$) in X_{h_α} and $X_{h_{\alpha+1}}$.

Next, put each hill vertex in the unique node set that contains the neighbors of the hill vertex, i.e., put $c^\alpha_{i,j}$ in $X_{h_{(\alpha-1)(lr^2+1)+i+1}}$.

Let $1 \leq i \leq r$. We now consider the ith string component. For all p, $1 \leq p \leq m$, β, $0 \leq \beta \leq lr^2$, put $d^i_{(\delta^i_p-1)(lr^2+1)+1+\beta}$ in $X_{h_{(p-1)(lr^2+1)+1+\beta}}$, and in $X_{h_{(p-1)(lr^2+1)+2+\beta}}$. Put the vertices $\{e^i_{\delta^i_p-1,j} \mid 1 \leq j \leq 2r+2\}$ in $X_{h_{(p-1)(lr^2+1)+1}}$, and put the vertices

$\{e^i_{\delta^i_p,j} \mid 1 \leq j \leq 2r+2\}$ in $X_{h_{p(lr^2+1)+1}}$. If $\delta^i_p + 1 \neq \delta^i_{p+1}$, we must 'fold' the part from the ith string component between what is put on top of the pth hill and what is on top of the $(p+1)$st hill. First, we define sets Y_j, $\delta^i_p \cdot (lr^2+1) \leq j \leq \delta^i_{p+1} \cdot (lr^2+1)$, and put (for j in this range), d^i_j and d^i_{j+1} in Y_j. Also, for each blob and symbol checker vertex v, if their neighbors belong to some set Y_j, put v in Y_j. Note that the maximum size of a set Y_j is $2r+4$. The sequence of sets $Y_{\delta^i_p \cdot (lr^2+1)}, \ldots, Y_{\delta^i_{p+1} \cdot (lr^2+1)}$ forms a domino tree-decomposition, with the tree being a path. Note that the first, and the last set of this domino tree-decomposition are contained in $X_{h_{p(lr^2+1)+1}}$. We now take $\lceil \frac{1}{2}(\delta^i_p \cdot (lr^2+1) - (\delta^i_{p+1} \cdot (lr^2+1)) - 1) \rceil$ new nodes, by identifying the second and one but last node of the node sets Y, the third and the second but last node of the node sets Y, etc, i.e., we take sets $Z_t = Y_{\delta^i_p \cdot (lr^2+1)+t} \cup Y_{\delta^i_{p+1} \cdot (lr^2+1)-t}$, and make the new node with set Z_t adjacent to the node with set Z_{t+1} in the tree T. Also, make Z_1 adjacent to $h_{p(lr^2+1)+1}$. Further, put vertices $e^i_{0,j'}$ and d^i_0 in X_{h_1}, put $e^i_{l_i,j'}$ and $d^i_{l_i(lr^2+1)+1}$ in $X_{h_{q+1}}$, and 'fold' the first and the last part of the ith string path, in case $\delta^i_1 \neq 1$, or $\delta^i_m \neq l_i$, respectively.

Finally, symbol checker vertices that are not yet placed are put in that set that contains both its neighbors.

It is a tedious but rather straightforward verification, that the construction above gives a domino tree-decomposition of G with width k. We just note that $X_{h_{p(lr^2+1)+1}}$ contains at most $\frac{4}{5}(k+1) + 2r(2r+4) = \frac{4}{5}(k+1) + \frac{1}{5}(k+1)$ vertices. □

The theorem now follows from the transformation, given above, and the fact that LONGEST COMMON SUBSEQUENCE is $W[t]$ hard for all $t \in \mathbf{N}$ [7, 8]. □

Theorem 22 DOMINO TREEWIDTH *is NP-complete.*

Proof: Membership in NP is trivial. Observe that the transformation, given in the proof of Theorem 20, is a polynomial time transformation from the NP-complete LONGEST COMMON SUBSEQUENCE problem to DOMINO TREEWIDTH. Hence, the latter is NP-complete. □

6 Conclusions

In this paper, we considered the notion of domino treewidth. We showed a correspondence between bounded domino treewidth, and bounded degree and treewidth, and obtained several results on the complexity of determining the domino treewidth of a given graph.

We believe the notion of domino treewidth can be of use for other investigations in the (algorithmic) theory on the treewidth of graphs. For instance, having a domino tree-decomposition of logarithmic height, and of bounded width allows easy schemes for some problems on graphs that can be changed dynamically under the operations: delete an edge, change the weight or label of a vertex or edge, where

each such operation takes logarithmic time: for many problems, there are algorithms, solving these problems in linear time, of the type, where for each node 'something' is computed, and this information can be computed, given the information for the children of the node in constant time. For many problems, it is easy to see that when the graph is changed by an edge deletion or a weight or label change of an edge or vertex, for nodes that do not have an endpoint of the edge involved or the vertex involved in its node-set, and this holds also for all its descendants, the information to be computed is not changed. So, when we have a domino tree-decomposition, only $O(\log n)$ nodes must have their information recomputed, so the update can be done in $O(\log n)$ time. Unfortunately, the large constants involved make this scheme impractical. Other approaches for dynamic algorithms on graphs with bounded treewidth, which may be more practical, can be found e.g. in [6, 10].

References

[1] K. A. Abrahamson, R. G. Downey, and M. R. Fellows. Fixed-parameter tractability and completeness IV: On completeness for $W[P]$ and PSPACE analogues. Technical Report DCS-216-IR, Department of Computer Science, University of Victoria, Victoria, B.C., Canada, 1993.

[2] S. Arnborg, D. G. Corneil, and A. Proskurowski. Complexity of finding embeddings in a k-tree. *SIAM J. Alg. Disc. Meth.*, 8:277–284, 1987.

[3] H. L. Bodlaender. Some classes of graphs with bounded treewidth. *Bulletin of the EATCS*, 36:116–126, 1988.

[4] H. L. Bodlaender. A linear time algorithm for finding tree-decompositions of small treewidth. In *Proceedings of the 25th Annual Symposium on Theory of Computing*, pages 226–234, New York, 1993. ACM Press.

[5] H. L. Bodlaender. A tourist guide through treewidth. *Acta Cybernetica*, 11:1–23, 1993.

[6] H. L. Bodlaender. Dynamic algorithms for graphs with treewidth 2. To appear in proceedings WG'93, 1994.

[7] H. L. Bodlaender, R. G. Downey, M. R. Fellows, and H. T. Wareham. The parameterized complexity of sequence alignment and consensus (extended abstract). To appear in: proceedings Conference on Pattern Matching '94, 1993.

[8] H. L. Bodlaender, M. R. Fellows, and M. Hallett. Beyond *NP*-completeness for problems of bounded width: Hardness for the W hierarchy. Manuscript. To appear in: proceedings STOC'94, 1994.

[9] H. L. Bodlaender and R. H. Möhring. The pathwidth and treewidth of cographs. *SIAM J. Disc. Meth.*, 6:181–188, 1993.

[10] R. F. Cohen, S. Sairam, R. Tamassia, and J. S. Vitter. Dynamic algorithms for optimization problems in bounded tree-width graphs. In *Proceedings of the 3rd Conference on Integer Programming and Combinatorial Optimization*, 1993.

[11] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness I: Basic results. Manuscript, 1991. To appear in SIAM J. Comput.

[12] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness II: On completeness for $W[1]$. Manuscript, 1991. To appear in Theoretical Computer Science, Ser. A.

[13] R. G. Downey and M. R. Fellows. Fixed-parameter tractability and completeness III: Some structural aspects of the W hierarchy. Technical Report DCS-191-IR, Department of Computer Science, University of Victoria, Victoria, B.C., Canada, 1992.

[14] J. Engelfriet, L. M. Heyker, and G. Leih. Context-free graph languages of bounded degree are generated by Apex graph grammars. Technical Report 91-16, Department of Computer Science, Leiden University, Leiden, the Netherlands, 1991. To appear in Acta Informatica.

[15] E. Horowitz and S. Sahni. *Fundamentals of Computer Algorithms.* Pitman, London, 1978.

[16] B. Reed. Finding approximate separators and computing tree-width quickly. In *Proceedings of the 24th Annual Symposium on Theory of Computing*, pages 221–228, 1992.

[17] N. Robertson and P. D. Seymour. Graph minors. II. Algorithmic aspects of tree-width. *J. Algorithms*, 7:309–322, 1986.

[18] D. Seese. Tree-partite graphs and the complexity of algorithms. In L. Budach, editor, *Proc. 1985 Int. Conf. on Fundamentals of Computation Theory, Lecture Notes in Computer Science 199*, pages 412–421, Berlin, 1985. Springer Verlag.