Snakes and Ladders: a Treewidth Story

Steven Chaplick[1], Steven Kelk[1], Ruben Meuwese[1], Matúš Mihalák[1], and Georgios Stamoulis[1]

Dept. Advanced Computing Sciences, Maastricht University, the Netherlands

Abstract. Let G be an undirected graph. We say that G contains a ladder of length k if the $2 \times (k+1)$ grid graph is an induced subgraph of G that is only connected to the rest of G via its four cornerpoints. We prove that if all the ladders contained in G are reduced to length 4, the treewidth remains unchanged (and that this bound is tight). Our result indicates that, when computing the treewidth of a graph, long ladders can simply be reduced, and that minimal forbidden minors for bounded treewidth graphs cannot contain long ladders. Our result also settles an open problem from algorithmic phylogenetics: the common chain reduction rule, used to simplify the comparison of two evolutionary trees, is treewidth-preserving in the display graph of the two trees.

1 Introduction

This is a story about treewidth, but it starts in the world of biology. A phylogenetic tree on a set of leaf labels X is a binary tree representing the evolution of X. These are studied extensively in computational biology [16]. Given two such trees a natural aim is to quantify their topological dissimilarity [12]. Many such dissimilarity measures have been devised and they are often NP-hard to compute, stimulating the application of techniques from parameterized complexity [7]. Recently there has been a growing focus on treewidth. This is because, if one takes two phylogenetic trees on X and identifies leaves with the same label, we obtain an auxiliary graph structure known as the *display graph* [6]. Crucially, the treewidth of this graph is often bounded by a function of the dissimilarity measure that we wish to compute [13]. This has led to the use of Courcelle's Theorem within phylogenetics (see e.g. [13,11]) and explicit dynamic programs running over tree decompositions; see [10] and references therein. In [14] the spin-off question was posed: is the treewidth of the display graph actually a meaningful measure of phylogenetic dissimilarity *in itself* - as opposed to purely being a route to efficient algorithms? A closely-related question was whether parameter-preserving reduction rules, applied to two phylogenetic trees to shrink them in size, also preserve the treewidth of the display graph? The well-known *subtree reduction rule* is certainly treewidth preserving [14]. However, the question remained whether the *common chain reduction rule* [2] is treewidth-preserving. A common chain is, informally, a sequence of leaf labels $x_1, \ldots, x_k$ that has the same order in both trees. Concretely, the question arose [14]: is it possible to truncate a common chain to *constant length* such that the treewidth of the display graph is preserved? Common chains form ladder-like structures in the display graph, i.e., this question is about how far ladders can be reduced in length without causing the treewidth to decrease.

In this article we answer this question affirmatively, and more generally. Namely, we do not restrict ourselves to display graphs, but consider arbitrary graphs. A *ladder L of length $k \geq 1$* of a graph G is a $2 \times (k+1)$ grid graph such that L induces (only) itself and that L is only connected to the rest of the graph by its four cornerpoints. First, we prove that a ladder L can be reduced to length 4 without causing the treewidth to decrease, and that this is best possible: reducing to length 3 sometimes causes the treewidth to decrease. We also show that if $tw(G) \geq 4$ then reduction to length 3 is safe and, again, best possible. These tight examples are also shown to exist for higher treewidths. Returning to phylogenetics, and thus when G is a display graph, we leverage the extra structure in these graphs to show that common chains can be reduced to 4 leaf labels (and thus the underlying ladder to length 3) without altering the treewidth: this result is thus slightly stronger than on general G.

Our proofs are based on first principles: we directly modify a tree decomposition to get what we need. In doing so we come across the problem that, unless otherwise brought under control, the set of bags that contain a given ladder vertex of G can wind and twist through the tree decomposition in very pathological ways. Getting these *snakes* under control is where much of the hard work and creativity lies, and is the inspiration for the title of this paper.

From a graph-theoretic perspective our results have the following significance. First, it is standard folklore that shortening paths (i.e. suppressing vertices of degree 2) is treewidth-preserving, but there is seemingly little in the literature about shortening recursive structures that are slightly more complex than paths, such as ladders. (Note that Sanders [15] did consider ladders, but only for recognizing graphs of treewidth at most 4, and in such a way that the reduction destroys the ladder topology). Second, our results imply a new safe reduction rule for the computation of treewidth; a survey of other reduction rules for treewidth can be found in [1]. Third, we were unable to find sufficiently precise machinery, characterisations of treewidth or restricted classes of tree decomposition in the literature that would facilitate our results. Perhaps most closely related to our ladders are the more general *protrusions*: low treewidth subgraphs that "hang" from a small boundary [9, Ch. 15-16]. There are general (algorithmic) results [5] wherein one can safely cut out a protrusion and replace it with a graph of parameter-proportional size instead – these are based on a problem having *finite integer index* [4]. Such techniques might plausibly be used to prove that there is *some* constant to which ladders might safely be shortened, but our tight bounds seem out of their reach. Finally, the results imply that minimal forbidden minors for bounded treewidth cannot have long ladders.

The above context leads us to ask whether our results on ladders can be (i) (elegantly) generalized to more complex structures than ladders; (ii) can they be made constructive; (iii) can recognition of such structures be performed efficiently? We hope that our results will stimulate new research in this direction.

2 Preliminaries

We follow [14] for notation. A *tree decomposition* of an undirected graph $G = (V, E)$ is a pair $(\mathcal{B}, \mathbb{T})$ where $\mathcal{B} = \{B_1, \ldots, B_q\}$, $B_i \subseteq V(G)$, is a multiset of *bags* and $\mathbb{T}$ is a tree whose q nodes are in bijection with $\mathcal{B}$, and

(tw1) $\cup_{i=1}^{q} B_i = V(G)$;
(tw2) $\forall e = \{u, v\} \in E(G), \exists B_i \in \mathcal{B}$ s.t. $\{u, v\} \subseteq B_i$;
(tw3) $\forall v \in V(G)$, all the bags B_i that contain v form a connected subtree of $\mathbb{T}$.

The *width* of $(\mathcal{B}, \mathbb{T})$ is equal to $\max_{i=1}^{q} |B_i| - 1$. The *treewidth* of G, denoted $tw(G)$, is the smallest width among all tree decompositions of G. Given a tree decomposition $\mathbb{T}$ of a graph G, we denote by $V(\mathbb{T})$ the (multi)set of its bags and by $E(\mathbb{T})$ the set of its edges. Property (tw3) is also known as *running intersection property*. Without loss of generality, we consider only connected graphs G.

Note that subdividing an edge $\{u, v\}$ of G with a new degree-2 vertex uv does not change the treewidth of G. In the other direction, suppression of degree-2 vertices is also treewidth preserving *unless* it causes the only cycle in a graph to disappear (e.g. if G is a triangle); unlike [14] we will never encounter this boundary case. An equivalent definition of treewidth is based on chordal graphs. Recall that a graph G is chordal if every induced cycle in G has exactly three vertices. The treewidth of G is the minimum, ranging over *all* chordal completions $c(G)$ of G (we add edges until G becomes chordal), of the size of the maximum clique in $c(G)$ minus one. Under this definition, each bag of a tree decomposition of G naturally corresponds to a maximal clique in a chordal completion of G [3].

We say that a graph H is a *minor* of another graph G if H can be obtained from G by deleting edges and vertices and by contracting edges.

A *ladder L of length* $k \geq 1$ is a $2 \times (k + 1)$ grid graph. A *square* of L is a set of vertices of L that induce a 4-cycle in L. We call the endpoints of L, i.e., the degree-2 vertices of L, the *cornerpoints* of L. We say that a graph G *contains* L if the following holds (see Fig. 1 for illustration):

1. The subgraph induced by vertices of L is L itself.
2. Only cornerpoints of L can be incident to an edge with an endpoint outside L.

Observe that a ladder of length k is a minor of the ladder of length $(k + 1)$. Treewidth is non-increasing under the action of taking minors, so reducing the length of a ladder in a graph cannot increase the treewidth of the graph.

Suppose G contains a ladder L. We say that L *disconnects* G if L contains a square $\{u, v, w, x\}$ such that the two horizontal edges of the square (following Fig. 1, these are the edges $\{u, w\}$ and $\{v, x\}$) form an edge cut of the entire graph G. Note that a square of L has this property if and only if all squares of L do. Also, if we reduce the length of a ladder L to obtain a shorter ladder L', L' disconnects G if and only if L does. We recall a number of results from Section 5.2 of [14]; these will form the starting point for our work.

Lemma 1 ([14]). *Suppose G contains a disconnecting ladder L. The ladder L can be increased arbitrarily in length without increasing the treewidth of G.*

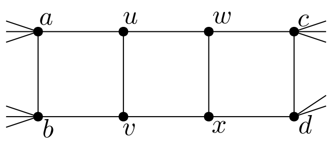

Fig. 1. A ladder L of length 3 with corner points a, b, c, d.

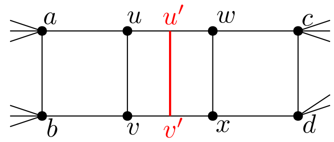

Fig. 2. Inserting a new edge $\{u', v'\}$ into ladder L results in ladder L' of length 4.

For the more general case, the following weaker result is known.

Lemma 2 ([14]). *Suppose G has $tw(G) \geq 3$ and contains a ladder. If the ladder is increased arbitrarily in length, the treewidth of G increases by at most one.*

We now make the following (new) observation.

Observation 1 *Suppose G contains a ladder L of length 2 or longer. If L is not disconnecting, then $tw(G) \geq 3$.*

We can leverage Observation 1 to reformulate Lemma 2 without the $tw(G) \geq 3$ assumption. However it then only applies to ladders of size at least two.

Lemma 3. *Suppose G contains a ladder L with length at least 2. If L is increased arbitrarily in length, the treewidth of the graph increases by at most one.*

If we start from a sufficiently long ladder, can the ladder be increased in length without increasing the treewidth? Past research has the following partial result.

Theorem 1 ([14]). *Let G be a graph with $tw(G) = k$. There is a value $f(k)$ such that if G contains a ladder of length $f(k)$ or longer, the ladder can be increased in length arbitrarily without altering (in particular: increasing) the treewidth.*

Ideally we would like a single, universal value *that does not depend on k*. In this article we will show that such a single, universal constant does exist.

3 Results

We first consider graphs of treewidth at least 4; we later remove this restriction.

Theorem 2. *Let G be a graph with $tw(G) \geq 4$. If G has a ladder L of length 3 or higher, the ladder can be lengthened arbitrarily without changing the treewidth.*

Proof. Due to Lemma 1 we can assume that L is not disconnecting. Our general strategy is to show that if G contains the ladder L shown in Fig. 1, we can insert an extra 'rung' in the ladder without increasing the treewidth, thus obtaining a ladder with one extra square (see Fig. 2). The extension of the ladder by one square can then be iterated to obtain an arbitrary length ladder.

Let L be the ladder shown in Fig. 1, and assume that G contains L. Let $(\mathcal{B}, \mathbb{T})$ be a minimum-width tree decomposition for G. We proceed with a case analysis. The cases are cumulative: we will assume that earlier cases do not hold.

Case 1. Suppose that $\mathcal{B}$ contains a bag B such that all four vertices from one of the squares of L are in B. Let $\{u, v, w, x\}$, say, be the square of L contained in bag B, where the position of the vertices is as in Fig. 1. We prolong the ladder as in Fig. 2 and create a valid tree decomposition for the new graph as follows: we introduce a new size-5 bag $B' = \{u', u, v, w, x\}$ which we attach pendant to B in the tree decomposition, and a new size-5 bag $B'' = \{u', v', v, w, x\}$ which we attach pendant to B'. Observe that this is a valid tree decomposition for the new graph. Due to the fact that $tw(G) \geq 4$, the treewidth does not increase, and the statement follows. Note that in this construction B'' contains all four of $\{u', w, v', x\}$, which is a square of the new ladder, so the construction can be applied iteratively many times as desired to produce a ladder of arbitrary length.

Case 2. Suppose that $\mathcal{B}$ contains a bag B such that $|B \cap \{a, u, w, c\}| \geq 2$ and $|B \cap \{b, v, x, d\}| \geq 2$. Let h_1, h_2 be two distinct vertices from $B \cap \{a, u, w, c\}$ and l_1, l_2 be two distinct vertices from $B \cap \{b, v, x, d\}$.

Observe that it is possible to partition the sequence a, u, w, c into two disjoint intervals H_1, H_2, and the sequence b, v, x, d into two disjoint intervals L_1, L_2 such that $h_1 \in H_1$, $h_2 \in H_2$, $l_1 \in L_1$ and $l_2 \in L_2$. If we contract the edges and vertices in each of H_1, H_2, L_1, L_2 we obtain a new graph G' which is a minor of G. Note that G' is similar to G except that the ladder now has two fewer squares – the three original squares have been replaced by a square whose corners correspond to H_1, H_2, L_1, L_2. This square might contain a diagonal but we simply delete this. We have $tw(G') \leq tw(G)$ because treewidth is non-increasing under taking minors. Now, by projecting the contraction operations onto $(\mathcal{B}, \mathbb{T})$ in the usual way[1], we obtain a tree decomposition $(\mathcal{B}', \mathbb{T}')$ for G' such that the width of $\mathbb{T}'$ is less than or equal to the width of $\mathbb{T}$. The bag in $(\mathcal{B}', \mathbb{T}')$ corresponding to B, let us call this B', contains all four vertices H_1, H_2, L_1, L_2. Clearly, $\mathbb{T}'$ is a valid tree decomposition for G'. We distinguish two subcases.

1. If $\mathbb{T}'$ has width at least 4, we can repeatedly apply the Case 1 transformation to B' to produce an arbitrarily long ladder without raising the width of $\mathbb{T}'$. The resulting decomposition will thus have width no larger than $\mathbb{T}$.
2. Suppose $\mathbb{T}'$ has width strictly less than 4, and thus strictly less than the width of $\mathbb{T}$. The width of $\mathbb{T}'$ is at least 3 because of the bag containing H_1, H_2, L_1, L_2. Case 1 introduces size-5 bags and can thus raise the width of the decomposition by at most 1. Hence we again obtain a decomposition whose width is no larger than $\mathbb{T}$ for a graph with an arbitrarily long ladder.

This concludes Case 2. Moving on, any chordalization of G must add the diagonal $\{w, v\}$ and/or the diagonal $\{u, x\}$. Hence we can assume that there is a bag

[1] In every bag of the decomposition vertices from H_1 all receive the vertex label H_1, and similarly for the other subsets H_2, L_1, L_2.

containing $\{u, w, v\}$ and another bag containing $\{v, w, x\}$. (If the other diagonal is added we can simply flip the labelling of the vertices in the horizontal axis i.e. $a \Leftrightarrow b, u \Leftrightarrow v$ and so on). As Case 1 does not hold we can assume that the bag containing $\{u, w, v\}$ is distinct from the bag containing $\{v, w, x\}$.

For the benefit of later cases we impose extra structure on our choice of minimum-width tree decomposition of G. The *distance* of decomposition $(\mathcal{B}, \mathbb{T})$ is the minimum, ranging over all pairs of bags B_1, B_2 such that B_1 contains $\{u, w, v\}$ and B_2 contains $\{v, w, x\}$, of the length of the path in $\mathbb{T}$ from B_1 to B_2.

We henceforth let $(\mathcal{B}, \mathbb{T})$ be a minimum-width tree decomposition of G such that, ranging over all minimum-width tree decompositions, the distance is minimized. Clearly such a tree decomposition exists.

Let B_1, B_2 be two bags from $\mathcal{B}$ with $\{u, w, v\} \subseteq B_1$, $\{v, w, x\} \subseteq B_2$ which achieve this minimum distance. Let P be the path of bags from B_1 to B_2, including B_1 and B_2. We assume that P is oriented left to right, with B_1 at the left end and B_2 on the right. As Case 2 does not hold, we obtain the following.

Observation 2 *B_1 does not contain b, x or d, and B_2 does not contain a, u, c.*

Case 3. B_1 and B_2 are adjacent in P. Although this could be subsumed into a later case it introduces important machinery; we therefore treat it separately.

Subcase 3.1: Suppose $a \in B_1$ (or, completely symmetrically, $d \in B_2$). Note that in this case all the edges in G incident to u are covered by B_1. Hence, we can safely delete u from all bags except B_1. Next, we create a new bag $B^* = \{a, u, w, v\}$ and attach it pendant to B_1, and finally we replace u with x in B_1. It can be easily verified that this is a valid tree decomposition for G and that the width is not increased, so it is still a minimum-width tree decomposition. However, B_1 is now a candidate for Case 2, and we are done. Note that replacing u with x in B_1 is only possible because B_1 is next to B_2 in P.

Subcase 3.2: Suppose Subcase 3.1 does not hold. Then $a \notin B_1$ (and, symmetrically, $d \notin B_2$). Putting all earlier insights together, we see $a, b, x, d \notin B_1$ and $a, u, c, d \notin B_2$. Observe that a, which is not in B_2, is not in any bag to the right of B_2. If it was, then the fact that some bag contains the edge $\{a, u\}$, and the running intersection property, entails that B_2 would contain at least one of a and u, neither of which is permitted. Hence, if a appears in bags other than B_1, they are all in the left part of the decomposition. Completely symmetrically, if d is in bags other than B_2, they are all in the right part of the decomposition. Because of this, b can only appear on the left of the decomposition (because the edge $\{a, b\}$ has to be covered) and c can only be on the right of the decomposition (because of the edge $\{c, d\}$). Summarising, B_1 (respectively, B_2) does not contain a or b (respectively, c or d) and all bags containing a or b (respectively, c or d) are in the left (respectively, right) part of the decomposition. Note that $c \notin B_1$. This is because edge $\{c, d\}$ has to be in some bag, and this must necessarily be to the right of B_2: but then running intersection puts at least one of c, d in B_2, contradiction. Symmetrically, $b \notin B_2$. So $a, b, c, d, x \notin B_1$ and $a, b, c, d, u \notin B_2$.

We now describe a construction that we will use extensively: *reeling in (the snakes) a and b*. Observe that, due to coverage of the edge $\{a, u\}$, and running intersection, there is a simple path of bags p_{ua} starting at B_1 that all contain u such that the endpoint of the path also contains a. The path will necessarily be entirely on the left of the decomposition. Due to coverage of the edge $\{b, v\}$ there is an analogously-defined simple path p_{vb}. (Note that p_{ua} and p_{vb} both exit B_1 via the same bag B'. If they exited via different bags, coverage of the edge $\{a, b\}$ would force at least one of a, b to be in B_1, yielding a contradiction). Now, in the bags along p_{ua}, except B_1, we relabel u to be a, and in the bags along p_{vb}, except B_1, we relabel v to be b. This is no longer necessarily a valid tree decomposition, because coverage of the edges $\{u, a\}$ and $\{v, b\}$ is no longer guaranteed, but we shall address this in due course. Next we delete the vertices u, w, v from all bags on the left of the decomposition, except B_1; they will not be needed. (The only reason that w would be in a bag on the left, would be to meet c, since B_1 and B_2 already cover the edges $\{u, w\}$ and $\{w, x\}$. But then, due to coverage of the edge $\{c, d\}$ and the fact that d only appears on the right of the decomposition, running intersection would put at least one of c, d in B_1, contradiction.) Observe that B' contains $\{a, b\}$. We replace B_1 with 5 copies of itself, and place these bags in a path such that the leftmost copy is adjacent to B', the rightmost copy is adjacent to B_2, and all other bags that were originally adjacent to B_1 can (arbitrarily) be made adjacent to the leftmost copy. In the 5 copied bags we replace $\{u, w, v\}$ respectively with: $\{a, u', b\}$, $\{u', b, v'\}$, $\{u', u, v'\}$, $\{v', u, v\}$ and $\{u, w, v\}$. It can be verified that this is a valid tree decomposition for G', and our construction did not inflate the treewidth - we either deleted vertices from bags or relabelled vertices that were already in bags - so we are done. The operation can easily be telescoped, if desired, to achieve an arbitrarily long ladder.

Case 4. P contains at least one bag other than B_1 and B_2.

Observation 3 *All bags in P contain v, w, by the running intersection property.*

We partition the bags of the decomposition into (i) B_1, (ii) bags *left* of B_1, (iii) B_2, (iv) bags *right* of B_2, (v) all other bags (which we call the *interior*).

Recall that $b, d, x \notin B_1$, $a, c, u \notin B_2$ (because Case 2 does not hold).

Observation 4 *No bag in the interior contains u or x. B_1 does not contain x, and no bag on the left contains x. Symmetrically, B_2 does not contain u, and no bag on the right contains u.*

Observation 5 *At least one of the following is true: $a \in B_1$, a is in a bag on the left. Symmetrically, at least one of the following is true: $d \in B_2$, d is in a bag on the right.*

Now, suppose w is somewhere on the left. We will show that then either w can be deleted from the bags on the left, or Case 2 holds. A symmetrical analysis will hold if v is somewhere on the right. Specifically, the only possible reason for w to be on the left would be to cover the edge $\{w, c\}$ – all other edges incident to w are already covered by B_1 and B_2. If no bags on the left contain c, we can simply delete w from all bags on the left. On the other hand, if some bag on the left contains c, then $c \in B_1$, because: $d \notin B_1$, the need to cover the edge $\{c, d\}$, the presence of d on the other 'side' of the decomposition (Observation 5), and running intersection. So we have that $c, u, w, v \in B_1$. This bag already covers all edges incident to w, except possibly the edge $\{w, x\}$. To address this, we replace w everywhere in the tree decomposition with x - this is a legal tree decomposition because some bag contains $\{w, x\}$ - and then add a bag $B' = \{u, w, x, c\}$ pendant to B. This new bag serves to cover all edges incident to w. But B_1 now contains u, v, c, x, so Case 2 applies, and we are done! Hence, we can assume that w is nowhere on the left, and, symmetrically, that v is nowhere on the right. In fact, the above argument can, independently of w, be used to trigger Case 2 whenever $c \in B_1$ or $b \in B_2$. So at this stage of the proof we know: $b, c, d, x \notin B_1$ (and c is not on the left) and $a, b, c, u \notin B_2$ (and b is not on the right).

Subcase 4.1: Suppose $a \notin B_1$. Then, a must only be on the left. It cannot be in the interior (or on the right) because the edge $\{u, a\}$ must be covered, $a \notin B_1$, $u \in B_1$, and u is not in the interior (Observation 4). Because a is on the left, and because some bag must contain the edge $\{a, b\}$, b must also be on the left. In fact b is only on the left. The presence of b both on the left and in the interior (or on the right) would force b into B_1 by running intersection, contradicting the fact that $b \notin B_1$. So a, b are only on the left. We are now in a situation similar to Subcase 3.2. We use the same *reeling in a and b* construction and we are done.

Subcase 4.2: Suppose $a \in B_1$. Note that here u has all its incident edges covered by B_1, so u can be deleted from all other bags. Recall that $b \notin B_1$. Due to edge $\{b, v\}$ some bag must contain both v and b. Suppose there is such a bag on the left. We attach a new bag $\{a, u, w, v\}$ pendant to B_1 and delete u from B_1. We put x in B_1 and to ensure running intersection we replace v with x in all bags anywhere to the right of B_1. This is safe, because in the part of the decomposition right of B_1, v only needs to meet x (and not b, because v meets b on the left). Thus, B_1 now contains $\{a, v, w, x\}$ and Case 2 can be applied.

Hence, we conclude that $\{v, b\}$ is not in a bag on the left. Because of this v can safely be deleted from all bags on the left. That is because any path p_{vb} that starts at B_1 and finishes at a bag containing b must go via the interior. In fact, such a path must avoid B_2, and is thus entirely contained in the interior. It avoids B_2 because $a, b \notin B_2$ and $\{v, b\}$ cannot be in a bag to the right: if it was, coverage of edge $\{a, b\}$, the fact that $a \in B_1$ and running intersection would mean that at least one of a and b is in B_2, yielding a contradiction.

The only case that remains is $a \in B_1$, $\{v, b\}$ is not in a bag on the left and thus p_{vb} is in the interior. By symmetry, we assume that $d \in B_2$, $\{w, c\}$ is not in a bag on the right and thus p_{wc} is in the interior. Consider any path p_{ab} starting at B_1, defined in the now familiar way. Note that no bag on the left of B_1 can contain b. This is because $\{v, b\}$ is in a bag in the interior: hence if b was also on the left, b would then by running intersection be in B_1 and we would be in an earlier case. This means that p_{ab} must go via the interior. Suppose the following operation gives a valid tree decomposition: delete u from B_1, attach a new bag $B^* = \{a, u, w, v\}$ pendant to B_1, and relabel all occurrences of a along the path

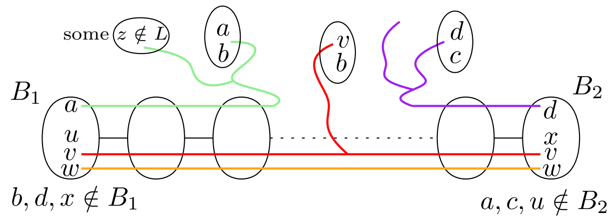

Fig. 3. Path p_{ab} goes via the interior, but it cannot be relabelled to b because it is used by other paths p_{az} to some neighbour z of a that does *not* lie on the ladder.

p_{ab} (except in B_1) with b. Then we are done, because we are back in Case 2. A symmetrical situation holds for the path p_{dc}.

Assume therefore that this transformation does not give a valid tree decomposition. This is the most complicated case to deal with. It is depicted in Fig. 3. The issue here is that the path p_{ab} (respectively, p_{dc}) necessarily goes via the interior, but cannot be relabelled with b (respectively, c) because the path is also part of p_{az} (respectively, p_{dz}) where z is some non-ladder vertex that is adjacent to a (respectively d). We deal with this as follows. We argue that some bag in the decomposition *must* contain a, b, v (and possibly other vertices). Suppose this is not the case. By standard chordalization arguments, every chordalization adds at least one diagonal edge to every square of the ladder. If $\{a, b, v\}$ are not together in a bag, then this is because the corresponding chordalization did not add the diagonal $\{a, v\}$ to square $\{a, u, b, v\}$. Hence, the chordalization must have added the diagonal $\{u, b\}$. This would in turn mean that some bag contains $\{a, u, b\}$. Such a bag must be on the interior, because this is the only place that b can be found. However, no bags in the interior contain u – contradiction.

Hence, some bag B' indeed contains $\{a, b, v\}$. Again, because b is only on the interior, B' must be in the interior. There could be multiple such bags, but this does not harm us. Let $B_{v\text{-done}}$ be the rightmost bag on the path P that is part of a path, starting from B_1, from a to some bag B' containing $\{a, b, v\}$. Let $B_{a\text{-done}}$ be the rightmost bag on P that contains a. Note that $B_{v\text{-done}}$ contains a (because of running intersection: $a \in B_1$ and $a \in B'$) and v, w (because it lies on P). We also have $a, v, w \in B_{a\text{-done}}$. By construction, $B_{v\text{-done}}$ is either equal to $B_{a\text{-done}}$ or left of it. This is important because it means that the only reason v might need to be in bags to the right of $B_{a\text{-done}}$ is to reach a bag containing x (i.e. to cover the edge $\{v, x\}$) – all other edges are already covered elsewhere in the decomposition; in particular, edge $\{v, b\}$ is covered by the bag B' containing $\{a, b, v\}$. See Fig. 4 for clarification. Recall that none of the bags on the path P contain both a and d. (If they did, there would be a bag containing $\{a, d, v, w\}$ and we would be in Case 2, done.) We also know that some path p_{dc} goes via the interior and thus that the penultimate bag on P (i.e. the one before B_2) thus definitely contains d. (To clarify: $c \notin B_2$, $d \in B_2$, the edge $\{c, d\}$ must be covered, and c is only in the interior). Combining these insights tells us that this penultimate bag definitely does *not* contain a, and hence $B_{a\text{-done}}$ is *not* equal to

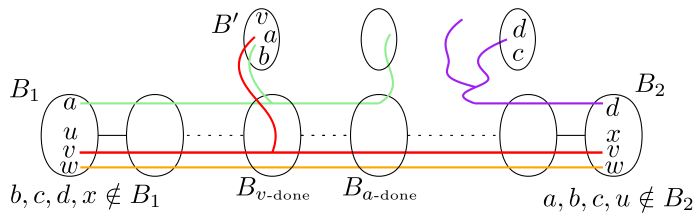

Fig. 4. The bags B', $B_{v\text{-done}}$ and $B_{a\text{-done}}$ illustrated. Note that $B_{a\text{-done}}$ cannot be the penultimate bag on the path P from B_1 to B_2, due to the presence of d in that bag.

the penultimate bag; it is further left. This fact is crucial. Consider $B_{a\text{-done}}$ and the bag immediately to its right on P. Between these two bags we insert a copy of $B_{a\text{-done}}$, call it B_r, remove a from B_r (i.e. forget it), and add the element x to it instead. Finally, we switch v to x in all bags on P right of B_r, including B_2 itself, and delete v from all bags in the tree decomposition that are anywhere to the right of B_r; there is no point having them there. It requires some careful checking but this is a valid (minimum-width) tree decomposition. Moreover, B_r contains w, v, x. The fact that $B_{a\text{-done}}$ was not the penultimate bag of P, means that the length of the path from B_1 to B_r is strictly less than the length of the path from B_1 to B_2: contradiction on the assumption that these were the closest bags containing $\{u, w, v\}$ and $\{w, v, x\}$ respectively. We are done. ∎

We now deal with the situation when the $tw(G) \geq 4$ assumption is removed.

Lemma 4. *If G has a ladder L of length 5 or longer, the ladder can be increased in length arbitrarily without altering (in particular: increasing) the treewidth. This holds irrespective of the treewidth of G.*

Proof. Let L be a ladder of length 5 or longer. We can assume that L is not disconnecting and $tw(G) \leq 3$. We select the three most central squares and label these as in Fig. 1. These are flanked on both sides by at least one other square. Hence, a, b, c, d each has exactly one neighbour outside the 3 squares, let us call these a', b', c', d' respectively, where $\{a', b'\}$ is an edge and $\{c', d'\}$ is an edge. Now, $tw(G) = 3$ because L is not disconnecting. The only part of the proof of Theorem 2 that does not work for $tw(G) = 3$ is Case 1 and (indirectly) Case 2 because these create size-5 bags. We show that neither case can hold.

Consider Case 1. Let B be a bag containing one of the three most central squares S of the ladder (these are the only squares to which Case 1 is ever applied). A *small* tree decomposition is one where no bag is a subset of another. If a tree decomposition is not small, then by running intersection it must contain two adjacent bags $B^\dagger, B^\ddagger$ such that $B^\dagger \subseteq B^\ddagger$. The two bags can then be safely merged into $B^\ddagger$. By repeating this a small tree decomposition can be obtained without raising the width of the original minimum-width decomposition. Furthermore, some bag B will still exist containing S. If B has five or more vertices we immediately have $tw(G) \geq 4$ and we are done. Otherwise, let B' be any bag adjacent to B; such a bag must exist because G has more than 4 vertices. Due to the smallness of the decomposition we have $B \not\subseteq B'$ and $B' \not\subseteq B$. Hence, $B \cap B' \subset B$ and $B \cap B' \subset B'$. A *separator* is a subset of vertices whose deletion disconnects the graph. Now, $B \cap B'$ is by construction, and the definition of tree decompositions a separator of G. However, due to our use of the three central squares, S is not a separator, and no subset of it is a separator either; the inclusion of a', b', c', d' and the edges $\{a', b'\}$ and $\{c', d\}$, alongside the fact that L is not disconnecting, ensure this. This yields a contradiction. Hence Case 1 implies $tw(G) \geq 4$ i.e. it cannot happen when $tw(G) = 3$.

We are left with Case 2. This case replaces the three centremost squares with a single square, and deletes any diagonals that this single square might have, to obtain a new graph G'. We have $tw(G') \leq tw(G)$, by minors. Note that $tw(G') \geq 3$ because the shorter ladder in G' (which has length at least 3) is still disconnecting. Hence, $tw(G') = tw(G) = 3$. The decomposition $\mathbb{T}'$ of G' obtained by projecting the contraction operations onto the tree decomposition $\mathbb{T}$ of G, is a valid tree decomposition (as argued in Case 2) with the property that the width of $\mathbb{T}'$ is less than or equal to the width of $\mathbb{T}$. $\mathbb{T}'$ cannot have width less than 3, so it must have width 3. Hence it is a tree decomposition of G' in which all bags have at most four vertices. We then transform $\mathbb{T}'$ into a small tree composition: this does not raise the width of the decomposition, and every bag prior to the transformation either survives or is absorbed into another. Consider the bag B' containing H_1, H_2, L_1, L_2. The presence of $a', b', c,', d'$ in G' and the fact that the ladder in G' is not disconnecting, means that H_1, H_2, L_1, L_2 is not a separator for G', and neither is any subset of those four vertices. But the intersection of B' with any neighbouring bag *must* be a separator. Hence B' must contain a fifth vertex, contradiction. So Case 2 cannot happen when $tw(G) = 3$. ■

We can, however, still do better. The following proof is in the appendix.

Theorem 3. *If G has a ladder L of length 4 or longer, the ladder can be increased in length arbitrarily without altering (in particular: increasing) the treewidth. This holds irrespective of the treewidth of G.*

Tightness. The constant 4 in the statement of Theorem 3 is equal to the constant obtained for the 'bottleneck' case $tw(G) = 3$. An improved constant 3 for this case is not possible, as Fig. 5 shows. It is natural to ask whether, when $tw(G) \geq 4$, we can start from ladders of length 2, rather than 3. This is also not possible. See Fig. 6. In fact, we have examples up to treewidth 20. These can be found at `https://github.com/skelk2001/snakes_and_ladders`. We conjecture that this holds for all treewidths, but defer this to future work.

Implications for phylogenetics. A phylogenetic tree is a binary tree whose leaves are bijectively labelled by a set X. The *display graph* of T_1, T_2 on X is obtained by identifying leaves with the same label. In [14] it was asked whether *common chains* could be truncated to constant length without lowering the treewidth of the display graph. Theorem 3 establishes that the answer is *yes*. In fact, due to the restricted structure of display graphs, we can prove a stronger result: truncation to 4 leaves (i.e. 3 squares) is safe, and this is best possible. Theorem 4 summarizes this. We provide full details in the appendix.

Theorem 4. *Let T_1, T_2 be two unrooted binary phylogenetic trees on the same set of taxa X, where $|X| \geq 4$ and $T_1 \neq T_2$. Then exhaustive application of the subtree reduction and the common chain reduction (where common chains are reduced to 4 leaf labels) does not alter the treewidth of the display graph. This is best possible, because there exist tree pairs where truncation of common chains to length 3 does reduce the treewidth of the display graph (see Fig 7).*

Acknowledgements. We thank Hans Bodlaender and Bart Jansen for insightful feedback. We also thank the members of our department for useful discussions.

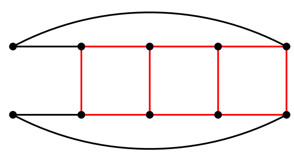

Fig. 5. A graph of treewidth 3 that contains a ladder with 3 squares, shown in red. Increasing the length of the ladder by 1 square increases the treewidth to 4.

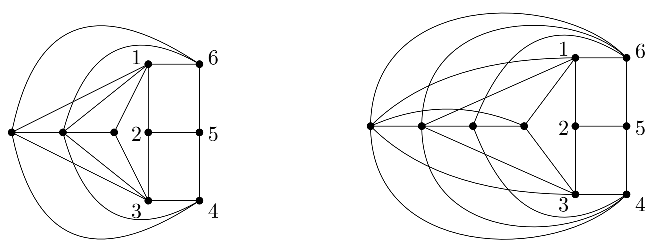

Fig. 6. These graphs have treewidth 4 (left) and 5 (right) and each one has a length 2 ladder, induced by $\{1, 2, 3, 4, 5, 6\}$. In each case increasing the length of the ladder to length 3 increases the treewidth of the graph by one.

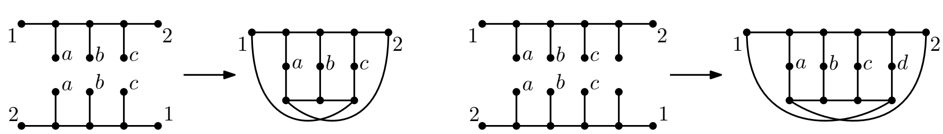

Fig. 7. Lengthening the common chain $\{a, b, c\}$ to $\{a, b, c, d\}$ in these phylogenetic trees causes the treewidth of the display graph to increase. Equivalently: shortening common chains to 3 leaf labels is not guaranteed to preserve treewidth in the display graph.

Ruben Meuwese was supported by the Dutch Research Council (NWO) KLEIN 1 grant *Deep kernelization for phylogenetic discordance* (OCENW.KLEIN.305).

References

1. Abu-Khzam, F.N., Lamm, S., Mnich, M., Noe, A., Schulz, C., Strash, D.: Recent Advances in Practical Data Reduction, pp. 97–133. Springer Nature Switzerland, Cham (2022)
2. Allen, B., Steel, M.: Subtree transfer operations and their induced metrics on evolutionary trees. Annals of Combinatorics **5**, 1–15 (2001)
3. Blair, J., Peyton, B.: Graph Theory and Sparse Matrix Computation, chap. An Introduction to Chordal Graphs and Clique Trees, pp. 1–29. Springer New York, New York, NY (1993)
4. Bodlaender, H.L., van Antwerpen-de Fluiter, B.: Reduction algorithms for graphs of small treewidth. Information and Computation **167**(2), 86–119 (2001). https://doi.org/10.1006/inco.2000.2958
5. Bodlaender, H.L., Fomin, F.V., Lokshtanov, D., Penninkx, E., Saurabh, S., Thilikos, D.M.: (meta) kernelization. Journal of the ACM **63**(5), 44:1–44:69 (2016). https://doi.org/10.1145/2973749
6. Bryant, D., Lagergren, J.: Compatibility of unrooted phylogenetic trees is FPT. Theoretical Computer Science **351**(3), 296–302 (2006)
7. Bulteau, L., Weller, M.: Parameterized algorithms in bioinformatics: An overview. Algorithms **12**(12), 256 (2019)
8. Fernández-Baca, D., Vakati, S.R.: On compatibility and incompatibility of collections of unrooted phylogenetic trees. Discrete Applied Mathematics **245**, 42–58 (2018)
9. Fomin, F.V., Lokshtanov, D., Saurabh, S., Zehavi, M.: Kernelization: Theory of Parameterized Preprocessing. Cambridge University Press (2019). https://doi.org/10.1017/9781107415157
10. van Iersel, L., Jones, M., Weller, M.: Embedding phylogenetic trees in networks of low treewidth. In: Chechik, S., Navarro, G., Rotenberg, E., Herman, G. (eds.) 30th Annual European Symposium on Algorithms, ESA 2022, September 5-9, 2022, Berlin/Potsdam, Germany. LIPIcs, vol. 244, pp. 69:1–69:14. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2022). https://doi.org/10.4230/LIPIcs.ESA.2022.69, arXiv preprint arXiv:2207.00574
11. Janssen, R., Jones, M., Kelk, S., Stamoulis, G., Wu, T.: Treewidth of display graphs: bounds, brambles and applications. Journal of Graph Algorithms and Applications **23**(4), 715–743 (2019)
12. John, K.S.: The shape of phylogenetic treespace. Systematic Biology **66**(1), e83 (2017)
13. Kelk, S., van Iersel, L., Scornavacca, C., Weller, M.: Phylogenetic incongruence through the lens of monadic second order logic. Journal of Graph Algorithms and Applications **20**(2), 189–215 (2016)
14. Kelk, S., Stamoulis, G., Wu, T.: Treewidth distance on phylogenetic trees. Theoretical Computer Science **731**, 99–117 (2018)
15. Sanders, D.: On linear recognition of tree-width at most four. SIAM Journal on Discrete Mathematics **9**(1), 101–117 (1996)
16. Steel, M.: Phylogeny: Discrete and Random Processes in Evolution. SIAM (2016)

A Omitted proofs

Observation 1. *Suppose G contains a ladder L of length 2 or longer. If L is not disconnecting, then $tw(G) \geq 3$.*

Proof. Let a, b, u, v, w, x be the six vertices in the ladder L with edges as shown in Fig. 1. Observe that G contains a K_4 minor. Specifically, take $\{a, b\}, u, v$ and $\{w, x\}$ as the four corners of the minor pre-contraction. The fact that L is not disconnecting means that there is a path from $\{a, b\}$ to $\{w, x\}$ that leaves the ladder at one end and re-enters it via the other, inducing the minor edge between $\{a, b\}$ and $\{w, x\}$. K_4 is the (unique) forbidden minor for graphs of treewidth at most 2, so $tw(G) > 2$. ■

Observation 4. *No bag in the interior contains u or x. B_1 does not contain x, and no bag on the left contains x. Symmetrically, B_2 does not contain u, and no bag on the right contains u.*

Proof. Recall Observation 3. If some bag in the interior contained u or x, we could due to running intersection (in particular: due to $u \in B_1, x \in B_2$) find two bags on P containing $\{u, v, w\}$ and $\{v, w, x\}$ that were closer than B_1 and B_2, contradiction. Next, we have already established that $x \notin B_1$ and $u \notin B_2$. If x was on the left, then running intersection would put $x \in B_1$ (because $x \in B_2$), contradiction. If u was on the right, then running intersection would put $u \in B_2$ (because $u \in B_1$), contradiction. ■

Observation 5. *At least one of the following is true: $a \in B_1$, a is in a bag on the left. Symmetrically, at least one of the following is true: $d \in B_2$, d is in a bag on the right.*

Proof. The edge $\{a, u\}$ (respectively, the edge $\{d, x\}$) needs to be in a bag, and from Observation 4 u and x are restricted in their possible locations. ■

Lemma 5. *If G has $tw(G) \geq 3$ and a ladder L of length 1 or longer whereby at least one of the four cornerpoints of the ladder has degree 2, the ladder can be increased in length arbitrarily without altering (in particular: increasing) the treewidth.*

Proof. Let $\{a, b, c, d\}$ be the four cornerpoints of the ladder and let c be a degree-2 cornerpoint. Assume as usual that c and d are part of square $\{w, x, c, d\}$. If we suppress c and relabel vertex d as cd we create a triangle $\{w, x, cd\}$ in G without altering its treewidth. Take any minimum-width decomposition. This triangle must be contained in some bag B of the decomposition. Pendant to B we attach a new chain of bags $\{w, x, w', cd\}$, $\{x, w', x', cd\}$. This is a valid tree decomposition for the graph obtained from G by inserting a new rung in the ladder $\{w', x'\}$ parallel to edge $\{w, x\}$. The construction can be iterated if desired to insert more rungs in the ladder, by attaching bags pendant to bag $\{x, w', x', cd\}$. Once completed the degree 2 vertex can be re-introduced via subdivision, if desired. ■

Theorem 3. *If G has a ladder L of length 4 or longer, the ladder can be increased in length arbitrarily without altering (in particular: increasing) the treewidth. This holds irrespective of the treewidth of G.*

Proof. As usual, if $tw(G) \geq 4$ we can use Theorem 2, and if L is disconnecting then we are done thanks to Lemma 1. So, let G be a graph with ladder L with four squares that is not disconnecting. From Observation 1 we have $tw(G) = 3$. Let a', a, u, w, c be the vertices on the top of the ladder and b', b, v, x, d be the vertices on the bottom. Our goal as usual is to show that adding a square does not increase the treewidth of G. We henceforth assume that G is (vertex) biconnected. This is because the treewidth of a graph is the maximum treewidth ranging over all biconnected components of the graph. The ladder, both before and after lengthening, belongs to a single biconnected component. So we henceforth focus only on that component.

As in the proof of Theorem 2 we focus on the three squares defined by vertices a, u, w, c and b, v, x, d. We have an extra square on the left side - a', a, b', b - and this has the same 'buffer' role as in the proof of Lemma 4. However, there is no extra buffer square on the right of the ladder, and this causes some mild complications.

We begin with several observations. Whenever, in the proof of Theorem 2, Case 1 or Case 2 are shown to apply to four vertices from a, u, w, b, v, x (i.a. avoiding c and d) then we are already done[2]. That is because, using exactly the same argument as in Lemma 4, those four vertices or a subset thereof cannot (after contraction, when Case 2 applies) be a separator, so there must be an extra vertex in the bag: $tw(G) > 3$, and a contradiction on the assumption $tw(G) = 3$ is obtained. The absence of a separator is due to the vertices a', b', the edges $\{a', b'\}, \{a', a\}, \{b', b\}$ and the fact that we avoid vertices c, d, allowing them to assume the same role as c', d' in the proof of Lemma 4. Hence, the main headache is when Case 1 or Case 2 is applied to four vertices involving c and/or d. The problem is that, due to not having any knowledge about the non-ladder neighbours of c and d, we cannot guarantee that the four vertices (or a subset thereof) do not form a separator. Hence, it is not possible to directly derive a contradiction on $tw(G) = 3$. In such situations the size-5 bags introduced by Case 1 and Case 2 might, therefore, inflate the treewidth. However, it is possible to circumvent this, as we shall see.

The fact that G is biconnected and L is not disconnecting means that at least one of the following holds:

- There is a simple path that starts at c, avoids all other vertices on the ladder (in particular: d), and ends at a' or b';
- There is a simple path that starts at d, avoids all other vertices on the ladder (in particular: c), and ends at a' or b';

[2] When applying Case 2 in such a context, we leave c and d alone i.e. we only contract two squares of the ladder, not three.

(Note that it is permitted that these paths intersect, perhaps multiple times, at vertices distinct from c and d). Now, suppose *both* these paths exist. In this situation we are done, because Cases 1 and Case 2 can be applied in their unconstrained form i.e. they do not even need to avoid c and d. This is because, after contracting the three squares to one, the single square remaining (or a subset thereof) cannot be a separator. This is because c and d are independently of each other connected to the other side of the ladder (and as observed above the square a', a', b, b' basically has the same separator-preventing function at the other end of the ladder). Hence, the proof of Theorem 2 goes through essentially unchanged, the only difference being that Case 1 and Case 2 now generate contradictions on the assumption $tw(G) = 3$.

Hence, we assume that only one such path exists. For now, suppose this path starts at d. Due to biconnectivity, and the absence of the second path, there are exactly two possibilities:

1. c has degree 2, in which case its only neighbours are d and w.
2. The edge $\{c, d\}$ is a separator.

If $\{c, d\}$ is a separator, then deleting this edge splits G into G_1 and G_2, where G_1 is the connected component containing the ladder. The treewidth of G is equal to the maximum of $tw(G_1 \cup \{c, d\})$ and $tw(G_2 \cup \{c, d\})$. This is because $\{c, d\}$ is a clique separator. In particular, any tree decomposition of $G_1 \cup \{c, d\}$ (respectively, $G_2 \cup \{c, d\}$) must have $\{c, d\}$ together in some bag, so any two such tree decompositions can be linked together via a single extra bag containing $\{c, d\}$ in order to obtain a tree decomposition for G. Now, $G_2 \cup \{c, d\}$ (and thus its treewidth) is unchanged if the ladder is extended, so we can focus on the graph $G_1 \cup \{c, d\}$. This brings us back into the situation that c has degree 2. (An exactly symmetrical argument holds if the path had started at c.)

Hence, at this point we can assume without loss of generality that exactly one of c and d has degree 2. We can invoke Lemma 5 and we are done. ■

B Resolving an open problem from phylogenetics

B.1 The subtree and chain reduction rules are treewidth-preserving in the display graph

The research in this article was originally inspired by a question arising in *phylogenetics*, a subfield of bioinformatics. An (unrooted, binary) phylogenetic tree on a set of discrete labels X representing a set of species, is an undirected, connected, binary tree whose leaves are bijectively labelled by X. Due to this bijection we often refer to leaves and labels interchangeably. Two phylogenetic trees T_1, T_2, both on X, are defined to be equal if there is an isomorphism from one to the other that preserves the labels X.

The *display graph* $D = D(T_1, T_2)$ of two unrooted binary phylogenetic trees T_1, T_2 on X is obtained by identifying leaves with the same label. Display graphs have been quite intensively studied in recent years, see e.g. [6,13,8,11,10]. We note that the assumption $T_1 \neq T_2$ guarantees that $|X| \geq 4$ and that the display graph contains a K_4 minor, and thus has treewidth at least 3. For convenience we thus henceforth assume that $T_1 \neq T_2$. This is a very reasonable assumption because it is easy to check in polynomial time whether two phylogenetic trees are equal (equivalently, that the display graph has treewidth at most 2). The fact that the display graph has treewidth at least 3 is useful because it allows us to suppress degree-2 nodes in the display graph without altering the treewidth or worrying about the whole display graph vanishing. In particular, we can suppress the degree-2 nodes that are created in the formation of the display graph when vertices with the same leaf label are identified.

The *subtree reduction* is a data reduction rule very often used to simplify phylogenetic trees when computing a dissimilarity measure between them. Let x, y be distinct labels in X. If x, y have a common parent in T_1 and a common parent in T_2, then the *cherry reduction* deletes the leaves x and y from both trees and assigns label xy to the parent. The subtree reduction is simply when the cherry reduction is applied to exhaustion.

It was shown in [14] that if one applies the subtree reduction rule to T_1, T_2 to obtain new trees T'_1, T'_2 then $tw(D(T'_1, T'_2)) = tw(D(T_1, T_2))$. The question arose whether another frequently encountered data reduction rule, the *common chain reduction* rule, is also treewidth-preserving in the display graph. The definition of a common chain is rather technical[3], but in essence it is an uninterrupted sequence of leaves that exist in the same order in both trees. The main nuance is that the first two leaves in the sequence, and the last two leaves in the sequence, might be unordered in one or both trees. The common chain reduction rule simply reduces common chains to length k, for some given constant k. It is well known that reduction to length 3 preserves a number of commonly encountered phylogenetic dissimilarity measures. Is there a constant k such that the common chain reduction rule preserves the treewidth of the display graph?

Theorem 3 shows that such a k definitely exists. Specifically, common chains with k leaves induce ladders in the display graph with $k-1$ squares. Hence, if we reduce common chains to length 5, we know that the corresponding ladder in the display graph is reduced to length 4, and thus (via Theorem 3) that the treewidth is preserved. The result can be summarized as follows:

Lemma 6. *Let T_1, T_2 be two unrooted binary phylogenetic trees on the same set of taxa X, where $|X| \geq 4$ and $T_1 \neq T_2$. Then exhaustive application of the subtree reduction and the common chain reduction (where common chains are reduced to 5 leaf labels) does not alter the treewidth of the display graph.*

The subtree and chain reductions are the centrepiece of many kernelization results in phylogenetics [7]. Now we have established that these two reduction

[3] For $n \geq 2$, let $C = (\ell_1, \ell_2 \ldots, \ell_n)$ be a sequence of distinct taxa in X. We call C an n-chain of T if there exists a walk $p_{\ell_1}, p_{\ell_2}, \ldots, p_{\ell_n}$ in T and the elements in $p_{\ell_2}, p_{\ell_3}, \ldots, p_{\ell_{n-1}}$ are all pairwise distinct. Note that ℓ_1 and ℓ_2 may have a common parent or ℓ_{n-1} and ℓ_n may have a common parent. Furthermore, if $p_{\ell_1} = p_{\ell_2}$ or $p_{\ell_{n-1}} = p_{\ell_n}$ holds, then C is said to be *pendant* in T. If a chain C exists in both phylogenetic trees T_1 and T_2 on X, we say that C is a *common chain* of T_1 and T_2.

rules *also* preserve treewidth in the display graph. We note that, when trying to compute phylogenetic distance measures or parameters by exploiting low treewidth in the display graph, this treewidth-preserving result does not help: it is actually more advantageous if the treewidth decreases. Yet, if we are using the treewidth of the display graph as a proxy for phylogenetic dissimilarity, as proposed in [14], these results show that these two reduction rules are safe.

B.2 One step further: leveraging the restricted structure of display graphs

Display graphs are a restricted subclass of graphs, and it is not obvious whether the tight examples we discuss at the end of the main article are display graphs, so it is natural to ask whether it is treewidth-preserving to reduce common chains to 4 or perhaps even fewer labels (rather than the 5 labels stated in Lemma 6). We note, by leveraging an example from [14], that truncation to 3 leaf labels (inducing the shortening of ladders to 2 squares in the display graph) is *not* treewidth preserving. If we take the display graph of the two phylogenetic trees shown in Fig. 7 (far left), which have a common chain of length 3 on the leaf labels $\{a, b, c\}$, we get a ladder in the display graph with 2 squares. The display graph has treewidth 3. However, if we take the display graph of the two trees shown in Fig. 7 (second from right), where the chain has been increased to length 4 (and the ladder thus to 3 squares), the treewidth of the display graph increases to 4.

The question thus arises whether truncation to 4 leaf labels is safe. It turns out that it is! With a little more effort we obtain the following theorem.

Theorem 4. *Let T_1, T_2 be two unrooted binary phylogenetic trees on the same set of taxa X, where $|X| \geq 4$ and $T_1 \neq T_2$. Then exhaustive application of the subtree reduction and the common chain reduction (where common chains are reduced to 4 leaf labels) does not alter the treewidth of the display graph. This is best possible, because there exist tree pairs where truncation of common chains to length 3 does reduce the treewidth of the display graph (see Fig 7).*

Proof. We prove this by showing that if T_1 and T_2 have a common chain C with 4 leaves (a, b, c, d), and that this chain is then made longer to obtain new trees T_1' and T_2', we have $tw(D) = tw(D')$ where $D = D(T_1, T_2)$ and $D' = D(T_1', T_2')$. Note firstly that the chain C induces a ladder L with 3 squares in D[4]. Hence, if $tw(D) \geq 4$ we can use Theorem 2 and we are done. Similarly, if L is disconnecting then we are done via Lemma 1. Hence, we can assume that L is not disconnecting, and $tw(D) < 4$, so $tw(D) = 3$.

[4] To remain consistent with the formal definition of a ladder - in particular to ensure that it has four cornerpoints - it might be necessary to leave some degree 2 vertices in the display graph unsuppressed, or even to introduce degree 2 vertices via subdivision. However, as discussed in the preliminaries this will not alter the treewidth.

Now, we argue that there must exist a leaf label $x \notin \{a, b, c, d\}$ such that in one of the two trees, say T_1, x is on the "a" side of the chain in the tree, and in the other T_2 on the "d" side of the tree. If this was not so then in D there would be no path from the left of the ladder to the right that does not pass through the chain i.e. L is disconnecting, contradiction. Now, suppose that C is pendant in T_1 and/or T_2; recall that a chain is *pendant* if two of its outermost leaves have a common parent (and this occurs if and only if the tree has no other leaf labels on that side of the chain). This induces at least one degree-2 cornerpoint in L, so by Lemma 5 we are done. Hence, assume that C is pendant in neither tree. This means $|X| \geq 6$ because each tree needs at least one leaf label on both sides of the chain. Observe that if there is a leaf label $y \notin \{a, b, c, d, x\}$ such that in T_1 y is on the d side of the chain and in T_2 on the a side of the chain, then $tw(D) \geq 4$, yielding a contradiction. To see that it has $tw(D) \geq 4$ observe that the graph shown in Fig. 7 (far right) will be a minor of it. So, let $y \notin \{a, b, c, d, x\}$ be any taxon that is on the d side of the chain in T_1; such a taxon must exist by virtue of the assumption that the chain is pendant in neither tree. Given that y cannot be on the a side of the chain in T_2, it must - like x - thus be on the d side of the chain in T_2. But then we are in the situation as shown in Fig. 8. The fact that the right end of the ladder is part of the yellow-highlighted cycle, has exactly the same separator-blocking effect as the buffer square $\{a', b', a, b\}$ used in Theorem 3. Hence, the same proof can be used as that theorem, and we are done. ■

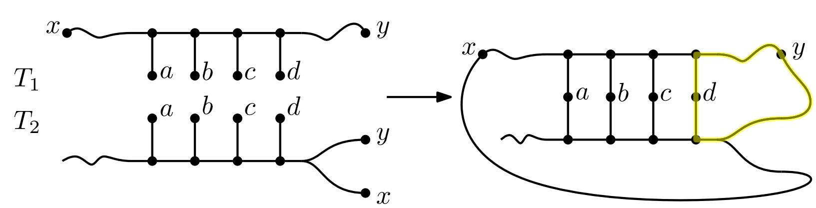

Fig. 8. If T_1 and T_2 have the shown structure, the 3 squares of the ladder induced in the display graph are flanked on the right by the yellow-highlighted cycle. This functions like the separator-blocking extra square in the proof of Theorem 3. This is one of the reason why chain-induced ladders can be safely reduced to 3 squares in display graphs (Theorem 4), rather than 4 as in general graphs.