A Congestion Parameter for Depth-First Graph Traversals

Codaline Bourotte[1] @ ORCID
École Normale Supérieure de Lyon, France

Gwendal Ducloz @ ORCID
École Normale Supérieure de Lyon, LIP (UMR 5668, Équipe MC2), France

Pekka Orponen @ ORCID
Aalto University, Finland

Shinnosuke Seki @ ORCID
University of Electro-Communications, Japan

Abstract

We explore a new graph parameter, *KLX number*, which quantifies the minimum edge congestion of depth-first search (DFS) traversals of a given graph. Originally motivated by a problem in RNA nanostructure design, this parameter is also of independent theoretical interest. Informally, the KLX number of a graph is defined as the minimum, over all its DFS traversals, of the maximum number of back edges that are simultaneously open during the traversal.

We provide full characterisations and linear-time recognition algorithms for graphs with KLX numbers 0, 1 and 2. We also relate KLX to tree-width, proving that any graph satisfies TW $\leq$ KLX+1. Furthermore, we show that the property KLX $\leq k$ is MSO_2-expressible for every fixed k. Combined with the tree-width bound, this result implies that determining whether a graph has KLX number at most k can be achieved in linear time for any constant k.

2012 ACM Subject Classification Mathematics of computing → Graph algorithms; Theory of computation → Fixed parameter tractability; Applied computing → Computational biology

Keywords and phrases KLX, depth-first search, DFS trees, k-connectedness, tree-width, parameterised complexity, monadic second-order logic, Courcelle's theorem, RNA nanotechnology

Digital Object Identifier 10.4230/LIPIcs.MFCS.2026.1

Related Version *Full Version*: `https://arxiv.org/abs/2606.24675` [5]

Funding *Codaline Bourotte* : Work supported in part by the grant "Bourse Région Mobilité Internationale Etudiants de la région Auvergne-Rhône-Alpes" and by Kubota Information Technology
Gwendal Ducloz: Work supported in part by the French government under the "France 2030" programme ANR-24-RRII-0001, ANR-22-CE09-0034, and ANR-22-CE45-0020.
Shinnosuke Seki: Work supported in part by Ambition Internationale de la Région Auvergne-Rhône-Alpes No. 00284230-1 (Rokam) and by the Aalto Science Institute visiting researcher programme.

Acknowledgements A major part of this work was done during an internship of C.B. at the University of Electro-Communications, Japan. Further work was pursued during a visit of S.S. and G.D. to Aalto University, Finland, and during an internship of G.D. at the University of Electro-Communications.

[1] Corresponding author

1 Introduction

1.1 Background

In this paper we investigate the minimum "congestion" of depth-first graph traversals, in the sense of how many back edges are simultaneously open (discovered but not concluded) at any point during the traversal. This concept is analogous to the notion of *spanning tree congestion* introduced by Ostrovskii [12], but is specific to ordered DFS trees, and addresses the dynamic traversal process rather than the static DFS tree.

Our interest in this concept was spurred by a practical problem in RNA nanotechnology [13], viz. minimising the number of kissing-loop connector motifs in the algorithmic design of co-transcriptionally folding RNA wireframe nanostructures [7, 8, 11].

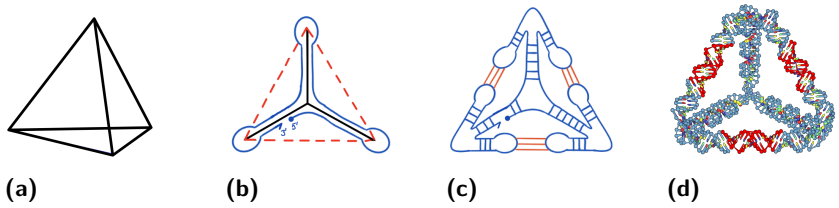


Figure 1 A spanning-tree based design scheme for RNA wireframe nanostructures. (a) Targeted wireframe model. (b) A spanning tree and strand routing of the wireframe graph. (c) Routing-based helical and kissing-loop pairings. (d) Nucleotide-level model. (Adapted with permission from [7].)

In this design task, one starts with a graph model G that represents the targeted wireframe nanostructure (Figure 1(a)), and then designs an initial route for the RNA strand by a traversal around some spanning tree T of G (blue curve in Figure 1(b)). This establishes an RNA base-pairing scheme for creating the tree T, but omits the co-tree edges (dashed red lines in Figure 1(b)). These are grafted to the design by extending the strand routing by two half-edges from both end-vertices of each co-tree edge, and terminating these extensions in "kissing loop" (KL) pairs with pairwise matching base sequences (Figure 1(c)). Figure 1(d) presents a helix-level model of the resulting nanostructure.

In nature, RNA is transcribed from a DNA template by a *polymerase enzyme* molecule and folds *co-transcriptionally*, that is, already while being transcribed. This opens up the possibility of reusing the KL base-pair sequences, because once a transcribed KL pair has closed, there is no longer a risk of mispairings from introducing another copy of the same sequence pair. This opportunity is potentially significant for large structures and motivates interest in cotranscriptional designs with a minimum number of simultaneously open KL pairs, that is, *graph traversals with a minimum number of simultaneously open co-tree edges*.

Co-transcriptional folding also has a downside: if the scheduling of the KL pairings relative to the helix transcriptions is bad, it is possible that the polymerase molecule becomes "trapped" by a KL pair that closes too early, and the winding of the downstream helices cannot be completed [9]. Surprisingly, this kinetic folding trap can be decisively avoided in wireframe designs *if and only if the spanning tree used for routing is a DFS tree* [11].

Accordingly, and following [11], we define the KLX ("kissing loop crossing") number of a graph G as the *minimum number of simultaneously open back edges in any DFS traversal of* G, and aim to characterise and recognise those graphs that can be constituted with a small KLX number. Even though this parameter thus has an application-driven origin, it connects to the broad literature on edge congestion problems, e.g. [3, 10, 12].

1.2 Structure of the paper

After introducing some preliminaries in Section 2, we provide characterisations of graphs with small KLX values in Section 3. Specifically, we show that: graphs with KLX ≤ 1 are cactus graphs, and graphs with KLX ≤ 2 admit an explicit, linear-time decidable characterisation. Computing the KLX value is known to be NP-hard [11], but in Section 4 we establish the existence of a linear-time fixed-parameter tractable (FPT) algorithm to test whether KLX $\leq k$ for any constant k. Our approach is based on Courcelle's Theorem [6], and consists of two key steps: we prove that the tree-width TW of any graph satisfies TW $\leq$ KLX $+ 1$, and we demonstrate that the condition KLX $\leq k$ is MSO_2-expressible. All proofs that have been omitted or abridged in this paper are available in the full version of the article at [5].

2 Preliminaries

All graphs $G = (V, E)$ considered in this paper are connected and simple. We denote by $\deg(x)$ the degree of a vertex $x \in V$, and by $\delta(G)$ and $\Delta(G)$ the minimum and maximum degrees of vertices in G. For $S \subseteq V$, we denote by $G[S]$ the subgraph of G induced by the vertices in S. We use the notation $|C|$ for the length of a sequence, path or cycle C.

A graph $G = (V, E)$ is *k-(vertex-)connected* if $|V| > k$ and G cannot be disconnected by removing fewer than k vertices. A 2-connected graph is also called *biconnected*. By $\kappa(G)$, we denote the largest integer k such that G is k-connected. The *k-edge-connectivity* and the notation $\lambda(G)$ are the edge analogues. By k-connectedness, we always mean k-vertex-connectedness. It is well known that $\kappa(G) \leq \lambda(G) \leq \delta(G)$. A *biconnected component* or *block* of G is a maximal biconnected subgraph. Any graph decomposes into a tree of biconnected components, known as the *block-cut tree* of the graph [4].

Starting from a vertex r and traversing G in a depth-first manner yields a spanning tree T for G of a special kind, called a *DFS tree* or a *Trémaux tree*, rooted at r. The unique path between two vertices x and y in T is denoted by xTy, and the subtree rooted at x is denoted by $T(x)$. The root r induces a *partial tree order* $\prec$ on the vertices of G, where $u \prec v$ if $u \in rTv$ (equivalently if $v \in T(u)$). The root is the unique minimum element of this order.

A vertex that is neither a leaf nor the root is called *internal*, and an internal vertex of degree at least three in T is called a *branch vertex*. We note the following simple lemma.

▶ **Lemma 1.** *The root of a DFS tree of a biconnected graph is incident to only one tree edge and is thus not a branch vertex.*

The edges of G are classified in two types: *tree edges*, which belong to T, and *back edges*, which do not. This nomenclature is justified by the property that, in a DFS tree, any back edge connects a vertex v to one of its ancestors, i.e., a vertex in rTv. We consider back edges as oriented correspondingly and write (v, u) for a back edge where $u \prec v$. In contrast, we use curly brackets for tree edges, such as $\{x, y\}$.

A DFS tree T is *ordered* if an ordering is specified for the children of each vertex. Each such ordering corresponds to a unique DFS traversal process that results in the tree T. Throughout the paper, when depicting an ordered tree, we assume that the children of a vertex are visited from left to right.

Given an ordered DFS tree T, its *traversal* or *(DFS-)contour* is the list of occurrences of its vertices on the tree tour. Formally, the traversal $\tau(T)$ of an ordered DFS tree T with root r is defined recursively as $\tau(r)$, where $\tau(u) = u$ if u is a leaf, and $\tau(u) = u\,\tau(v_1)\,u \cdots u\,\tau(v_k)\,u$, if u is a vertex with children $v_1, \ldots, v_k$ ordered left to right. Note that in a traversal, every tree edge $\{x, y\}$ occurs twice, once in orientation $x \to y$ and once in orientation $y \to x$.

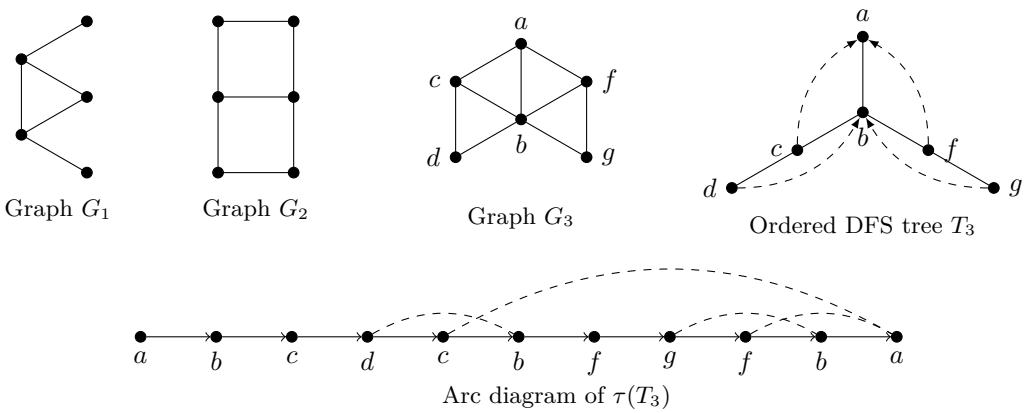


Figure 2 Three graphs G_1, G_2, and G_3, along with an ordered DFS tree T_3 of G_3 rooted at a and the corresponding arc diagram.

Given a DFS traversal $\tau(T)$, the corresponding *arc diagram* is constructed by laying out all the vertex occurrences of $\tau(T)$ in an oriented line, and representing each back edge (v, u) as an arc from the last occurrence of v in $\tau(T)$ to the next occurrence of u after the last occurrence of v—which exists as $u \prec v$ in T. An arc diagram is illustrated in Figure 2.

▶ **Definition 2** (KLX). *Let $e = \{x, y\}$ be a tree edge of an ordered DFS tree T, with $x \prec y$. The set $\mathcal{O}(e)$ of **open (back) edges for e** is defined as the set of back edges (v, u) for which the oriented edge $y \to x$ is contained in the arc diagram interval between the respective occurrences of v and u.*

We define $\mathrm{KLX}(T) = \mathrm{KLX}(G, T) = \max_{e \in E_T} |\mathcal{O}(e)|$ *and* $\mathrm{KLX}(G) = \min_T \mathrm{KLX}(G, T)$, *where the minimisation is over all ordered DFS trees T of G. An (ordered) DFS tree T^* that satisfies* $\mathrm{KLX}(G, T^*) = \mathrm{KLX}(G)$ *is a **KLX-minimum tree** of G.*

For instance, in the arc diagram of Figure 2, we have $\mathcal{O}(\{c, d\}) = \{(d, b)\}$ and $\mathcal{O}(\{b, f\}) = \{(c, a), (g, b), (f, a)\}$. In this case, $\mathrm{KLX}(T_3) = 3$.

▶ Remark 3. A graph has $\mathrm{KLX} = 0$ if and only if it is a tree, as in this case there are no back edges.

▶ **Example 4.** Figure 2 presents three graphs G_1, G_2, and G_3, with KLX numbers 1, 2, and 3, respectively. The KLX values can be verified by full enumerations of the respective DFS trees, but the present paper also provides simpler techniques. Graph G_1 is not a tree but a "cactus," and hence has $\mathrm{KLX} = 1$ (Theorem 12). Graph G_2 admits a path-like DFS tree that fits a characterisation of $\mathrm{KLX} = 2$ graphs in Theorem 15. Graph G_3 does not satisfy the conditions of Theorem 15, and thus has $\mathrm{KLX}(G_3) \geq 3$. Since we have just observed that $\mathrm{KLX}(G_3, T_3) = 3$, we can conclude that in fact $\mathrm{KLX}(G_3) = 3$.

The back edges in $\mathcal{O}(e)$ can be classified in two types, based on their relation to the tree edge e. We say that a back edge (v, u):

(i) *crosses* the edge e if $e \in uTv$;

(ii) *envelops* the edge e if there exists a branch vertex $x \in uTv$ with $x \neq u, v$, and with children w_1, w_2 such that w_1 precedes w_2, $v \in T(w_1)$, and either $e \in T(w_2)$ or $e = \{x, w_2\}$.

Similarly, we say that a back edge (v, u) *crosses a vertex* x if $x \in uTv$ and $x \neq u, v$.

For example, in the ordered DFS tree T_3 and its arc diagram in Figure 2, among the three back edges in $\mathcal{O}(\{b, f\})$, the edges (g, b) and (f, a) are crossing, and the edge (c, a) is enveloping.

This classification of back edges suggests the following simple lower bound for the KLX number, obtained by taking into account only the crossing edges.

▶ **Definition 5** (DFS tree congestion). *For a tree edge e in a DFS tree T of a graph G, we define its **(DFS-tree) congestion** $\mathrm{dtc}(e)$ as the number of back edges (v, u) that cross e. The congestion of a DFS tree T is then defined as $\mathrm{DTC}(T) = \mathrm{DTC}(G, T) = \max_{e \in T} \mathrm{dtc}(e)$, and that of a graph G as $\mathrm{DTC}(G) = \min_T \mathrm{DTC}(G, T)$.*

We observe that the value of $\mathrm{DTC}(T)$ is independent of the traversal order of the tree T, and thus the quantity $\mathrm{DTC}(G)$ matches Ostrovskii's spanning tree congestion parameter *stc* [12], when the latter is restricted to DFS trees and adjusted by -1.

▶ **Proposition 6.** *For any graph G, $\mathrm{DTC}(G) \leq \mathrm{KLX}(G)$. Moreover, there exist graphs for which this inequality is strict.*

The strictness of the lower bound is demonstrated e.g. by the biconnected graph G_3 in Figure 2, where from Example 4 we know that $\mathrm{KLX}(G_3) = 3$, and DFS tree T_3 witnesses that $\mathrm{DTC}(G) \leq 2$.

3 Characterisations of graphs with KLX ≤ 1 and KLX ≤ 2

In this section we provide characterisations of graphs with KLX number less than or equal to two. We begin by establishing some foundational lemmas, and then build on these to obtain the targeted characterisations. Finally, we present linear-time algorithms to decide whether a given graph satisfies $\mathrm{KLX} \leq 2$.

3.1 Some preliminary lemmas

▶ **Lemma 7.** *A block (biconnected component) B of a graph G satisfies $\mathrm{KLX}(B) \leq \mathrm{KLX}(G)$.*

Proof. Let T be a KLX-minimum tree of G, and let B be a block of G. Consider the subtree $T[V_B]$ of T induced by the vertices of B. Since B is biconnected, all paths in G between vertices of B lie entirely within B. Consequently, $T[V_B]$ is connected and acyclic, and thus forms a spanning tree (and hence an ordered DFS tree) of B. By the definition of KLX, the inequality $\mathrm{KLX}(B) \leq \mathrm{KLX}(B, T[V_B]) \leq \mathrm{KLX}(G)$ holds. ◀

▶ **Lemma 8.** *Let $k \geq 2$, and let T be a DFS tree of a k-connected graph, rooted at a vertex r. Any subtree of T rooted at a child of a vertex v is then connected to the path rTv by at least $k - 1$ back edges. Moreover, unless $v = r$, at least one back edge from each subtree crosses the tree edge between v and its parent.*

Proof. Suppose, for a contradiction, that for some subtree of T rooted at a child of v there are at most $k-2$ such back edges. Cutting all these back edges, along with the tree edge between v and the respective child, would disconnect G. This would imply $\kappa(G) \leq \lambda(G) \leq k - 1$, contradicting the k-connectivity of G.

Regarding the second statement, without such a crossing back edge, removing vertex v would disconnect the subtree from the rest of G, contradicting the biconnectivity of G. ◀

▶ **Corollary 9.** *In a DFS tree T of a biconnected graph G every vertex has at most $\mathrm{DTC}(T) \leq \mathrm{KLX}(T)$ children.*

▶ **Lemma 10.** *Let G be a biconnected graph and T a DFS tree of G. Then any vertex of G has degree at most* $\mathrm{KLX}(T)+1$ *if it is the root or a leaf of T, and degree at most* $2\cdot\mathrm{KLX}(T)+1$ *if it is an internal and not a branch vertex of T.*

Proof. If a vertex is a leaf, only one of its incident edges belongs to T. All others incident edges are back edges, and are crossing the tree edge adjacent to the leaf. Thus, the degree of a leaf cannot exceed $\mathrm{KLX}(T)+1$. The biconnectivity of G implies that the root is not a branching vertex (as noted in Lemma 1), allowing the same argument to be applied to the root.

Let us consider an internal vertex u that is not a branch vertex. In order for u not to be an articulation point, some back edge e must cross both the tree edges above and below u. If u had $2\cdot\mathrm{KLX}(T)+1$ or more incident edges, of which only two belong to T, one of the tree edges above or below u would be crossed by at least $\mathrm{KLX}(T)$ back edges. Adding the back edge e, this would contradict the definition of $\mathrm{KLX}(T)$. Hence, the degree of an internal vertex that is not a branch vertex is at most $2\cdot\mathrm{KLX}(T)+1$. ◀

▶ **Lemma 11.** *Let G be a biconnected graph with $\Delta(G)\geq\frac{k(k+1)}{2}+2$ for some $k\geq 1$. Then* $\mathrm{KLX}(G)\geq k+1$.

Proof. Suppose, for a contradiction, that there is a DFS tree T of G such that $\mathrm{KLX}(T)\leq k$. According to Lemma 10, any vertex that is the root or a leaf of T has degree at most $k+1$, and any internal vertex that is not a branch vertex of T has degree at most $2k+1$. It thus now suffices to show that any other (branch) vertex of T has degree at most $\frac{k(k+1)}{2}+1$ to get a contradiction with the assumption on $\Delta(G)$.

By Corollary 9 the number of children of any branch vertex of T is bounded by $\mathrm{KLX}(T)$. Let b be a branch vertex with $p\leq k$ children. Since T cannot branch at its root (Lemma 1), b has a parent. Let $e\in T$ be the edge connecting b to its parent, and let $e_1,\cdots,e_p\in T$ be the edges connecting b with its children. Without loss of generality, assume that the traversal of T explores e_1 first, e_2 second, and so on.

The remaining edges incident to b, if any, are back edges and must cross e, e_1, …, or e_p. As G is biconnected, for all $1\leq i\leq p$, there exists a back edge spanning from the subtree below the i-th child to crossing e (Lemma 8). Moreover, this i-th back edge envelops the edges $e_{i+1},\cdots,e_p$ and all edges below them. Therefore, e_i and e can be crossed by at most $k-i$ and $k-p$ more back edges, respectively. Hence, the degree of the branch vertex b is at most

$$p+1+k-p+\sum_{i=1}^{p}(k-i)=-\frac{1}{2}\left(p-\left(k-\frac{1}{2}\right)\right)^2+\frac{k(k+1)}{2}+\frac{9}{8}\leq\frac{k(k+1)}{2}+1$$

◀

3.2 Characterisation of graphs with KLX ≤ 1

A graph G is a *cactus graph* if every block of G is either an edge or a cycle [1].

▶ **Theorem 12.** *A graph G has* $\mathrm{KLX}(G)\leq 1$ *if and only it is a cactus graph.*

Proof. For the only-if direction, Lemma 7 implies that every block B of G satisfies $\mathrm{KLX}(B)\leq 1$. Applying Lemma 11 with $k=1$, no block involves a vertex with degree 3 or greater. Therefore, each block B must be either an edge or a simple cycle. Thus, G is a cactus graph.

For the other direction, given a cactus graph G, we construct a DFS traversal of G in which any tree edge is crossed by at most one back edge and enveloping never occurs. The

traversal can start at an arbitrary vertex. After the traversal enters a cycle block C via some articulation point u, it prioritises visiting the neighbour blocks of C (if any) in the block-cut tree before continuing along the cycle C. By this strategy, the only back edge ever open in C is the one connecting the final vertex of C back to u, which closes when the traversal returns back to u. Thus, at most one back edge in G is open at any time, and hence $\mathrm{KLX}(G) \leq 1$. ◀

▶ **Corollary 13.** *There is an $O(|V| + |E|)$ algorithm to test whether a graph has* $\mathrm{KLX} \leq 1$.

3.3 Characterisation of graphs with KLX ≤ 2

In the following, we first provide a structural characterisation of biconnected graphs with $\mathrm{KLX} = 2$, and then generalise this result to general graphs.

▶ **Lemma 14.** *Let T be a DFS tree of a biconnected graph G with* $\mathrm{KLX}(T) \leq 2$. *Then any branch vertex b has only two children in T, and at least one of the subtrees rooted at them has the following properties:*

1. *The subtree is a path (contains no further branch vertices).*
2. *All of the subtree is crossed by a single back edge that connects its (unique) leaf to an ancestor of b. No other back edges are incident to the subtree.*

Proof. Firstly, b is not the root of T (Lemma 1) and has only two children (Corollary 9). Let T_1, T_2 denote the two subtrees rooted at these children and assume, without loss of generality, that in the ordering of T that yields $\mathrm{KLX}(T) \leq 2$, tree T_1 precedes tree T_2.

By Lemma 8, there exists at least one back edge from T_1 to an ancestor of b that envelops all edges in T_2. This entails that no edge in T_2 can be crossed by more than one back edge. As a result, T_2 can not contain a branch vertex, as the edge from the branch vertex to its parent would be crossed by two back edges (Lemma 8). Consequently, T_2 must be a path.

Secondly, the unique leaf ℓ of T_2 must be incident to a back edge lest its parent be an articulation point. And the other endpoint w of this back edge must lie above the branch vertex b; otherwise, to prevent w from being an articulation point, another back edge would need to cross w. But then the tree edge connecting w to its only child would be crossed by two back edges, contradicting our earlier conclusion. Consequently, the back edge from ℓ to w crosses all edges in T_2, and no other back edges are incident to T_2. ◀

A new perspective: a spanning path $\tilde{T}$ with back paths. Let us note that a biconnected graph with $\mathrm{KLX} = 1$ is a cycle, by Theorem 12. From now on, we exclude this case, and focus on characterising biconnected graphs with KLX exactly equal to 2.

Let G be such a graph and T a KLX-minimum tree of G. By Lemma 14, every branch vertex b in T has at least one path-like subtree $b \to \cdots \to \ell$, with a unique back edge (ℓ, v) that crosses the whole subtree. We refer to such a path $b \to \cdots \to \ell \to v$ as a *back path* from b to v, and denote it by $[b, v]$. At the lowermost branch vertex — which exists as the graph is biconnected and not a cycle — both subtrees become back paths, and we consider also the other back edges, that do not participate in back paths, as special cases of this concept.

Contracting the path-like subtrees to back paths yields now a reduced spanning tree, in fact a spanning path $\tilde{T}$. The notions of edge crossings and the KLX number can be extended to $\tilde{T}$, resulting in $\mathrm{KLX}(G, \tilde{T}) = \mathrm{KLX}(G, T)$. Note also that there are now no enveloping back paths, as the tree $\tilde{T}$ no longer contains branch vertices.

Long back paths. We first remark that, as $\mathrm{KLX}(\tilde{T}) \leq 2$, no tree edge of $\tilde{T}$ can be crossed by more than two back paths. Consequently, the root r is incident to at most two back paths.

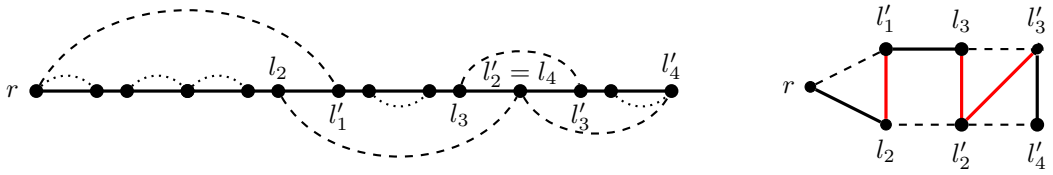


Figure 3 (Left) A DFS tree of a biconnected graph with KLX = 2, consisting of four long back paths $[r, l'_1], [l_2, l'_2], [l_3, l'_3], [l_4, l'_4]$ (drawn in dashed lines) and short back paths (dotted lines). (Right) A structural version of the corresponding graph, where all edges are paths, and black edges may contain cactus chains.

Let us then define a notion of *long back paths* by induction.

Initialisation. The first long back path $[l'_1, l_1]$ is defined as follows:

(i) If r is incident to only one back path $[u, r]$, we set $[l'_1, l_1] = [u, r]$.

(ii) If r is incident to two back paths $[u_1, r]$ and $[u_2, r]$, with $u_1 \preceq u_2$, we set $[l'_1, l_1] = [u_2, r]$.

Induction. Let $[l'_i, l_i]$ be lastly defined long back path, with $l_i \prec l'_i$.

(i) If l'_i is the single leaf of $\tilde{T}$, then all long back paths have been defined.

(ii) Otherwise, to avoid being an articulation point, the branch vertex l' must be crossed by a back path, but no more than one, as $\mathrm{KLX}(\tilde{T}) \leq 2$. This unique back path $[l'_{i+1}, l_{i+1}]$ is defined as the next long back path. Moreover, $l_i \prec l_{i+1} \prec l'_i \prec l'_{i+1}$. For $i = 1$, it comes from the definition of the first long back path as the "longest" one. For $i \geq 2$, the relation holds as $[l'_i, l_i]$ is the only back path to cross l'_{i-1}, hence, $l_i \prec l'_{i-1} \preceq l_{i+1}$

All long back paths, hence, satisfy the following relation:

$$r = l_1 \prec l_2 \prec l'_1 \preceq l_3 \prec l'_2 \preceq l_4 \prec \cdots \prec l'_{p-2} \preceq l_p \prec l'_{p-1} \prec l'_p, \tag{1}$$

Note that if $\Delta(G) = 4$, then l_{i+2} can be equal to l'_i for all $1 \leq i \leq p-2$ (for instance, in Figure 3, $l'_2 = l_4$). Note however, that $\Delta(G) < 5$ according to Lemma 11.

Short back paths. Let us now classify the remaining back paths, called *short back paths*. By construction, for each $1 \leq i \leq p-1$, the tree path between l_{i+1} and l'_i is crossed by two long back paths, hence, no other back path can cross one of these edges.

However, for each $2 \leq i \leq p-1$, if $l_{i+1} \neq l'_{i-1}$, all edges along the tree path $l'_{i-1}\tilde{T}l_{i+1}$ are crossed only by the long back path $[l'_i, l_i]$, allowing one more back path to cross them. By characterisation of graphs with KLX 1, the only remaining possibility at this location is arbitrarily many *short back paths* $[k'_{i,1}, k_{i,1}], \ldots, [k'_{i,m}, k_{i,m}]$ $(m \geq 0)$, aligned in series, as

$$l_i \prec l'_{i-1} \preceq k_{i,1} \prec k'_{i,1} \preceq k_{i,2} \prec k'_{i,2} \preceq \cdots \preceq k_{i,m} \prec k'_{i,m} \preceq l_{i+1} \prec l'_i \tag{2}$$

Finally, the only non-treated location are on the tree path between l_1 and l_2 and between l'_{p-1} and l'_p. Similarly, the only possibility at these locations are arbitrarily many short back paths, aligned in series. We spot these short back paths with indices 1 and p, respectively.

$$\text{If } p = 1\text{:} \quad l_1 \preceq k_{1,1} \prec k'_{1,1} \preceq k_{1,2} \prec k'_{1,2} \preceq \cdots \preceq k_{1,m} \prec k'_{1,m} \preceq l'_1 \tag{3}$$

$$\text{If } p \geq 2\text{:} \quad \begin{cases} l_1 \preceq k_{1,1} \prec k'_{1,1} \preceq k_{1,2} \prec k'_{1,2} \preceq \cdots \preceq k_{1,m} \prec k'_{1,m} \preceq l_2 \\ l'_{p-1} \preceq k_{p,1} \prec k'_{p,1} \preceq k_{1,2} \prec k'_{p,2} \preceq \cdots \preceq k_{p,m} \prec k'_{p,m} \preceq l'_p \end{cases}$$

▶ **Theorem 15** (Biconnected graph with KLX = 2)**.** *A biconnected graph G has* KLX $= 2$ *if and only if it admits a DFS path $\tilde{T}$, and a partition of its back paths into long and short ones such that:*

(i) *long back paths satisfy equation (1), and*

(ii) *short back paths satisfy equations (2) and (3).*

Proof. The previous paragraphs prove the forward direction. In the opposite direction, a graph with a DFS path rooted at r, with long and short back paths satisfying all previous equations can be traversed in a way that no edge is enveloped or crossed by two back edges. Starting from the root r, enveloping can be avoided by first going straight down to the leaf b_n and then visiting the back paths during the back traversal, as stated in the proof of Lemma 14. For example, Figure 3 illustrates that no tree edge is crossed by more than two back paths. Thus, this traversal results in a DFS tree with KLX $= 2$. Together, all arguments from this subsection prove Theorem 15. ◀

▶ **Corollary 16.** *A graph G has* KLX ≤ 2 *if and only if it admits a DFS tree T with the following structural properties:*

1. *For each biconnected component C of G, the restriction of T to C is a DFS tree T_C exhibiting the back-path structure characterised in Theorem 15.*
2. *The articulation points linking each component C to the rest of the graph satisfy the following:*
   - *Components with* KLX $= 2$ *are articulated only along the main tree path or along one of the last (either long or short) back paths of the corresponding DFS tree T_C.*
   - *All other articulation points connect exclusively to cactus subgraphs.*

Proof of Corollary 16. Let G be a graph with KLX ≤ 2, and T its associated KLX-minimum tree. Let C be such a biconnected component of G (if any).

The previous characterisation of biconnected graph with KLX ≤ 2 (see Theorem 15) guarantees that $T[C]$ has a specific structure of "spanning path", as illustrated in Figure 3. Let us note this restricted DFS tree as T_C.

A key observation is that the traversal T_C explores all tree edges before back paths, as established in Lemma 14. This lemma let only one non-defined order at the lowermost branching vertex. If this vertex is part of a short back path, then T_C could visit it first, before the long back path.

Let $x \in V_C$ be an articulation point, and let H_x denote the component attached at x. We analyse KLX(H_x, x) depending on the position of x in T_C:

1. ***x* is on the main tree edge of T_C.** Then H_x is explored first during the DFS, while no back edge in C is open. Hence KLX$(H_x, x) \leq 2$. After exploring H_x, all its back edges are closed (since x is an articulation point), and the remainder of C is unaffected.
2. ***x* is on a long back path (not last one).** By the key observation, H_x is enveloped by the back edge opened by the previous back path. Since G has KLX 2, KLX$(H_x, x) \leq 1$.
3. ***x* is on a short back path (not the last being crossed by the last long back path).** As such a short back path is not the last one, it is crossed by a long back path. Therefore, H_x is enveloped by an edge open in a long back path and KLX$(H_x, x) \leq 1$.
4. ***x* is on the last long back path.** there are two subcases, depending on the order of T.
   - The last short back path crossed by the last long back path contain an articulation point y such that KLX$(H_y, y) = 2$. In that case, the traversal should visit y first, as the final back path must not be open before exploring H_y. This means that H_x is enveloped, hence, KLX$(H_x, x) \leq 1$.

- Otherwise, the long back path can be explored first and $\mathrm{KLX}(H_x, x) \leq 2$.

5. **x is on the last short back path being crossed by the last long back path.** This case mirrors the previous one. $\mathrm{KLX}(H_x, x) \leq 1$ whether there is an articulation point y on the last long back path such that $\mathrm{KLX}(H_y, y) = 2$. Otherwise, if no such y exist, $\mathrm{KLX}(H_x, x) \leq 2$.

Therefore, connected components with KLX 2 are only articulated along tree paths and along either the last long or short back paths of the DFS tree T_C. All other articulation points must connect to KLX 1 components (i.e., only cactus subgraphs).

This concludes the direct sense of the equivalence, the other is verified by construction. ◀

3.4 Algorithms for recognising graphs with KLX ≤ 2

For an algorithm recognising graphs with $\mathrm{KLX} \leq 2$, we first need three structural definitions:

(i) A *2-high-degree cycle* is a simple cordless cycle containing exactly two vertices of degree greater than 2.

(ii) A *ladder pattern* is a sequence of simple cycles such that:
- No cycle contains a vertex of degree 5 or higher.
- Any two adjacent cycles in the sequence share exactly one common path.
- Non-adjacent cycles in the sequence share no common path.

(iii) Let $\mathcal{C}_1, \ldots, \mathcal{C}_p$ (with $p \geq 2$) be a ladder pattern, and let $(u_1, d_1), \ldots, (u_{p-1}, d_{p-1})$ be the paths common to each pair of neighbouring cycles, such that all u_i (respectively all d_i) lie on the same side of the ladder. This means that all paths from u_i to u_{i+1} belong only to $\mathcal{C}_{i+1}$. A *zigzag motif of the ladder pattern* is then a sequence of edges of the form:
- $\{(v, d_1), (u_1, u_2), (d_2, d_3), (u_3, u_4), \ldots, (\cdot, w)\}$, or
- $\{(v, u_1), (d_1, d_2), (u_2, u_3), (d_3, d_4), \ldots, (\cdot, w)\}$,

where v and w are vertices located on one of the paths composed of degree 2 vertices, either between u_1 and d_1 or between u_p and d_p, respectively.

Algorithm 1 An algorithm recognising biconnected graphs with $\mathrm{KLX} \leq 2$

1: Check that there is no vertex of degree at least 5, and test whether $\mathrm{KLX}(G) \leq 1$.
2: Replace every 2-high-degree cycle by a path of length 2 between the two high degree vertices. Proceed linearly, without using the "replaced" edges.
3: Verify that the resulting graph forms a ladder pattern.
4: Check the placement of the replaced cycles:
- If the ladder pattern consists of a single cycle, accept if there is a vertex of degree 3.
- Otherwise, verify that there exists a zigzag motif of the ladder pattern and that all replaced cycles are correctly positioned along this zigzag motif.

▶ **Proposition 17.** *A biconnected graph G passes all the steps of Algorithm 1 if and only if* $\mathrm{KLX}(G) \leq 2$. *Algorithm 1 has time complexity* $\mathcal{O}(|V| + |E|)$.

Proof overview (full proof at [5]). We first show that 2-high-degree cycles correspond to short back paths. Once these are ignored, the graph must contain only long back paths according to Theorem 15, which in turn form exactly a ladder pattern.

The remaining task is to ensure that the short back paths are properly located along the DFS path. If G passes all steps of Algorithm 1, we can reconstruct a DFS tree that satisfies Theorem 15. This DFS traversal follows the zigzag motif of the identified ladder pattern.

All steps run in linear time, as they rely primarily on DFS searches. ◀

Overall, Algorithm 1 provides a structural characterisation of graphs with KLX = 2. Such a structure is presented in Figure 3, where short back paths are omitted and the ladder pattern is made explicit.

▶ **Proposition 18.** *Based on Corollary 16, there exists an algorithm that accepts a graph G if and only if* $\mathrm{KLX}(G) \leq 2$. *Moreover, it runs in time* $\mathcal{O}(|V| + |E|)$.

Proof overview (full proof at [5]). We describe the main idea of the algorithm.

Our goal is to verify the existence of a DFS tree T satisfying Corollary 16. We begin by decomposing G into its block-cut tree $\mathcal{T}$. We then check that each biconnected component B satisfies $\mathrm{KLX}(B) \leq 2$ using Algorithm 1. As established in the proof of Proposition 17, passing Algorithm 1 ensures the existence, for each B, of a DFS tree T_B satisfying Theorem 15. We further verify that the articulation points of B can be embedded in such a tree (possibly after a slight modification) so as to "locally" satisfy the constraints stated in Corollary 16.

The remaining step is to combine the local DFS trees T_B into a global DFS tree T of G. To this end, we determine, for each block B, the set of feasible root positions, and orient the edges of the block-cut tree $\mathcal{T}$ according to compatibility constraints between adjacent components. If these constraints are consistent—i.e., they define a valid orientation of $\mathcal{T}$—then we can assemble the T_B into a DFS tree T satisfying Corollary 16, and hence $\mathrm{KLX}(G) \leq 2$.

All steps run in linear time. A key point is that only $\mathcal{O}(1)$ candidate DFS structures need to be considered for each block. This follows from the structural characterisation of Theorem 15, which restricts attention to a constant number of maximal zigzag patterns, together with the efficient handling of short back paths via articulation points. ◀

4 An FPT algorithm for testing KLX $\leq$ k.

Having designed linear-time algorithms for deciding whether a graph has KLX less than or equal to 0, 1, or 2, we now turn to testing whether KLX $\leq k$, for a fixed integer k. In particular, we present a fixed-parameter tractable (FPT) algorithm with running time $\mathcal{O}(f(k) \cdot (|V| + |E|))$ for this task.

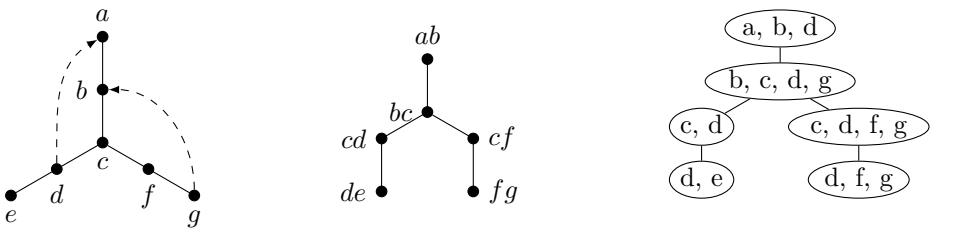
Algorithm 2 An FPT algorithm to test if a given graph is of KLX at most k.

Require: A graph G and an integer $k \in \mathbb{N}$.
1: Test that $\mathrm{TW}(G) \leq k + 1$. If that is not the case, return false.
2: Build $\mathrm{KLX}_{\leq k}$ as written in the proof in Section 4.2
3: Test if $G \vDash \mathrm{KLX}_{\leq k}$, and return the result.

Given a graph G as input, this algorithm first ensures that the tree width of G is at most $k + 1$. By the inequality $\mathrm{TW}(G) \leq \mathrm{KLX}(G) + 1$ shown in the following as Theorem 20, any graph with tree width greater than $k + 1$ must have $\mathrm{KLX}(G) \geq k + 1$, and can therefore be rejected. All graphs that pass Step 1 have bounded tree width, and hence, the following theorem by Courcelle applies.

▶ **Theorem 19** (Courcelle [6]). *If φ is an* MSO_2 *formula, then there exists an algorithm deciding whether a graph G of tree-width at most k satisfies φ in time $O(f(\varphi, k) \cdot (|V| + |E|))$ for some computable function f.*

It remains to prove Theorem 20 and to show that the property "$\mathrm{KLX}(G) \leq k$" can be expressed by an MSO_2 formula.



A DFS tree T with back edges | Its line graph | The associated tree decomposition

Figure 4 Conversion from a DFS tree to a tree-decomposition. Each bag in the tree decomposition is constructed from an edge of the DFS tree and augmented with vertices from back edges.

4.1 Tree-width as a lower bound on KLX

Given a graph $G = (V_G, E_G)$, a *tree decomposition of* G is a labelled tree $L = (V_L, E_L)$ such that the vertices of V_L, called *bags*, satisfy the following 3 properties:

- Every bag $B \in V_L$ is a subset of V_G.
- For every edge $\{a, b\} \in E_G$, there exists a bag $B \in V_L$ such that $a, b \in B$.
- For every vertex $v \in V_G$, the set of bags containing v induces a connected subtree of L. Formally, for every $v \in V_G$, we have that $L[\{B \in V_L \mid v \in B\}]$ is connected.

The *width* of a tree decomposition L is $\max_{B \in V_L} |B| - 1$. The **tree-width** of G, denoted by $\mathrm{TW}(G)$, is defined as the smallest width of a tree decomposition of G.

▶ **Theorem 20.** *For a graph* G, $\mathrm{TW}(G) \leq \mathrm{KLX}(G) + 1$.

Proof. For this, we transform the ordered DFS into an edge-tree, somewhat reminiscent of a line graph. Let G be a graph such that $\mathrm{KLX}(G) = k$, and let T be a directed DFS such that $\mathrm{KLX}(T) \leq k$ and denote $v = \mathcal{C}(G)$ its contour. Consider the *line graph* of T, defined as the graph T' whose vertices are the edges of T, $V_{T'} = E_T$ and where two edges are adjacent if one is the direct child of the other, that is $E_{T'} = \big\{\{(x, y), (y, z)\} : (x, y) \in E_T \wedge (y, z) \in E_T\big\}$.

Recall how in a back edge (x, y), $x \prec y$. For an edge $e \in T_E$, denote by $\mathcal{O}^{\mathrm{deep}}(e) := \{x : (x, y) \in \mathcal{O}(e)\}$ the set of all the "deeper sides" of the back edges that are open when visiting e. This will define our bags, and the intuition is that we keep the deeper side of the back-edges in the bag until we reach the second endpoint of the backward edge. For instance, in Figure 4, g is added to $\{c, f\}$, due to (g, b) crossing the edge $\{c, f\}$ and d is also included because (d, a) envelops $\{c, f\}$, resulting in the bag $\{c, d, f, g\}$.

For every vertex $e = \{u, v\}$ of T', define the bag $B_e := \mathcal{O}^{\mathrm{deep}}(e) \cup \{u, v\}$. Therefore, $|B_e| \leq |\mathcal{O}(e)| + 2 \leq \mathrm{KLX}(G) + 2$. We now show that T' alongside the bags $(B_e)_e$ is a tree decomposition of G.

Property 1. By definition, every bag is a subset of V_G.

Property 2. For every edge $e \in E_G$, there are two cases:

- If $e \in E_T$, then the bag B_e contains both endpoints x and y.
- If $e = \{x, y\} \notin E_T$, then it is a back edge. Suppose $x \prec y$, and consider (c, y) the last edge of the path xTy. Since $c \neq x$ because $e \notin E_T$, we have $x \in \mathcal{O}^{\mathrm{deep}}(\{c, y\})$, and therefore $x, y \in B_{\{c,y\}}$.

Property 3. Remark that for all $u \in V_G$, the list of the moments t in the contour such that $u \in \mathcal{O}^{\mathrm{deep}}((\tau_t, \tau_{t+1}))$ is an interval: it is a union of intervals all starting at the same point. That is, a union of intervals of the form $[\![a; b_i]\!]$ with a the last occurrence of u in τ, common to all intervals, and a serie of $(b_i)_i$. Since an interval of the contour is necessary a connected subtree of T_E, this gives us the property.

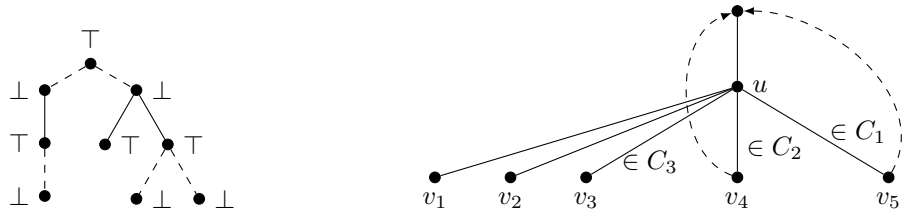


Figure 5 (Left) The encoding of a DFS tree by **R1**, **R2**, and **R3**. The solid edges represent $\top$ and the dotted ones represent $\bot$. (Right) The encoding of the order for $k = 3$. We only store the k rightmost children edges in sets C_1, C_2, and C_3. According to Lemma 22, if $\mathrm{KLX}(T) \leq k$, only v_3, v_4, and v_5 may contain back edges (dotted edges) crossing u.

Therefore, $\mathrm{TW}(G) \leq \max_e |B_e| - 1 \leq \mathrm{KLX}(G) + 1$. The inequality is strict in some cases. ◀

4.2 An MSO_2 formula of $\mathrm{KLX}(G) \leq k$

Monadic Second-Order logic (MSO_2) is a logical framework that extends classical first-order logic by allowing quantification not only over individual vertices and edges, but also over sets of vertices and edges. A graph property $P(G)$ is said to be *MSO_2-expressible* if there exists an MSO_2 formula φ such that $G \vDash \varphi$ if and only if $P(G)$ holds.

▶ **Theorem 21.** *The property* $\mathrm{KLX}_{\leq k}(G) :=$ "$\mathrm{KLX}(G) \leq k$" *is* MSO_2*-expressible.*

The full proof is given in the appendix of the full version of the article on arXiv [5]. We outline here the main ideas. A key difficulty is that MSO_2 cannot encode a total ordering of an unbounded number of children. However, the crucial observation is that only KLX children need to be ordered:

▶ **Lemma 22.** *Given a graph G with $\mathrm{KLX}(G) \leq k$, there exists an ordered DFS tree T of G with $\mathrm{KLX}(T) \leq k$ such that for every vertex x with children $y_1, ..., y_d$ ordered from left to right, only the rightmost k may contain a back edge crossing x.*

Let k be the KLX value we want to test. The proof is split in four parts, each showing how to encode a concept into MSO_2:

Encoding of the DFS Tree. We define an MSO_2 formula $\mathrm{DFS}(T, r, X)$ expressing that T is a DFS tree rooted at r. X is a "labelling" set used in the encoding to store for every vertex v the information of which neighbour is its parent. Specifically, we label every edge and vertex with either $\top$ or $\bot$ according to 3 rules **R1**, **R2**, and **R3**:

- **R1:** The root of T is labelled $\top$, and all edges incident to it are labelled $\bot$.
- **R2:** For every edge (u, v) of T, the vertex u is labelled $\top$ if and only if the vertex v is labelled $\bot$.
- **R3:** For every vertex, except the root, labelled with $x \in \{\top, \bot\}$, exactly one incident edge, namely the edge to its parent, is labelled x.

With this encoding, a path goes from a vertex to one of its ancestors if and only if it alternates between $\top$ and $\bot$, as depicted in Figure 5. We can then test that every back edge (y, x) corresponds to x being an ancestor of y. We also give a MSO_2 formula $\mathrm{Parent}(x, y)$ which tests if y is a parent of x.

Encoding of the order. According to Lemma 22, we don't need to encode the full order of the minimal-KLX tree, but only a part of it. We encode such an order of a DFS tree T using k sets $C_1, ..., C_k$, with C_i representing the i-th last visited child. Each set C_i is equipped with

a formula Choice(C_i) ensuring that for all vertices u at most one of its children is "chosen". We also require that all choices are distinct, which leads to a visiting order of the DFS tree: we first visit all non-chosen children before going into the children chosen by $C_k, ..., C_1$ in this order, as illustrated in Figure 5.

We also give the formula IsChoice such that IsChoice(C_i, u, v) tests if v is the child of u being "chosen" in C_i.

Encoding of the open edges. We describe two rules **X1** and **X2** that define for every back edge (y, x) a set $X_{(y,x)}$ containing all tree edges where (y, x) is open. Specifically, **X1** refers to the edges being crossed by (y, x) and **X2** refers to the edges being enveloped by it. Formally, $X_{(y,x)}$ is the smallest set of edges satisfying:

- **X1.** All tree edges in the path from y to x along T are contained in $X_{(y,x)}$.
- **X2.** For every vertex u strictly inside the path from y to x, denote $c_k, ..., c_1$ the choices at u, there is a $(c_i, u) \in X_{(y,x)}$ and for all $j < i$, $T(c_j) \subseteq X_{(y,x)}$

We then check that our rules are consistent: if $T, r, X, C_1, ..., C_k$ is an encoding of a DFS tree respecting all previously mentioned rules, with C_i choosing the i-th last child, then $X_{(y,x)}$ corresponds exactly to the set of edges where (y, x) is open.

Encoding of the KLX. Finally, we need to count how many back edges are open at the same time. To that end, we define a series of k edge sets $E_1, ..., E_k \subseteq E_G$ such that every back edge e finds a unique i with $e \in E_i$. Intuitively, each E_i represents one "slot" for simultaneously open back edges. If a back edge $e = (y, x)$ belongs to E_i, we require that $X_{(y,x)} \subseteq E_i$. This guarantees that no two back edges assigned to the same E_i are open simultaneously, and thus enforces $\mathrm{KLX}(T, G) \leq k$.

Formally, we describe three rules **E1**, **E2** and **E3** that $E_1, ..., E_k$ must satisfy:

- **E1.** For all back edge e, there is an i with $e \in E_i$.
- **E2.** For all tree edges e, we have $e \in E_i$ if and only if there exists a back edge e' such that $e \in X_{e'}$ and $e' \in E_i$.
- **E3.** For all pair of back edges e, e', if $e, e' \in E_i$, then $X_e \cap X_{e'} = \emptyset$.

Putting everything together, we show that G satisfies $\mathrm{KLX}(G) \leq k$ if and only if there exist the variables $T, r, X, C_1, \ldots, C_k, E_1, \ldots, E_k$ satisfying all above constraints. This yields an MSO_2 formula of the proposition $\mathrm{KLX}_{\leq k}$.

4.3 Concluding remarks

Algorithm 2 serves as a proof for the following theorem.

▶ **Theorem 23.** *For all $k \in \mathbb{N}$, there exists an $\mathcal{O}(f(k) \cdot (|V| + |E|))$ algorithm that tests whether a graph G has KLX number at most k.*

Let us analyse the complexity of Algorithm 2. Testing $\mathrm{TW}(G) \leq k + 1$ (step 1) can be done in running time complexity $2^{O(k^3)} \cdot (|E| + |V|)$ [2]. Notice that depending on how one applies Courcelle's theorem (with or without a tree-decomposition), one may not even need to run Step 1. An approximation algorithm can also be used, as just testing that the tree-width is bounded is the only goal of this step. Step 2 is constant-time as the formula $\mathrm{KLX}_{\leq k}$ does not depend either on the graph or on the DFS tree. Due to Courcelle's Theorem, Step 3 can be done in $\mathcal{O}(f(\mathrm{KLX}_{\leq k}, k) \cdot (|V| + |E|))$. Therefore, the overall complexity is in $\mathcal{O}(g(k) \cdot (|V| + |E|))$ with $g(k) = 2^{O(k^3)} + f(\mathrm{KLX}_{\leq k}, k + 1)$, linear time.

Although this algorithm as such is unlikely to serve any practical use, due to the worse-than-exponential dependency on k, the result suggests that further work could uncover usable versions of the algorithm.

References

1. Julio Araujo, Frédéric Havet, Claudia Linhares Sales, and Ana Silva. Proper orientation of cacti. *Theoretical Computer Science*, 639:14–25, 2016. doi:https://doi.org/10.1016/j.tcs.2016.05.016.
2. Hans L. Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM Journal on Computing*, 25(6):1305–1317, 1996. doi:10.1137/S0097539793251219.
3. Hans L. Bodlaender, Fedor V. Fomin, Petr A. Golovach, Yota Otachi, and Erik Jan van Leeuwen. Parameterized complexity of the spanning tree congestion problem. *Algorithmica*, 64(1):85–111, September 2012. doi:10.1007/s00453-011-9565-7.
4. Béla Bollobás. *Modern Graph Theory*, volume 184. Springer Science & Business Media, New York, NY, 1998. doi:https://doi.org/10.1007/978-1-4612-0619-4.
5. Codaline Bourotte, Gwendal Ducloz, Pekka Orponen, and Shinnosuke Seki. A congestion parameter for depth-first graph traversals, 2026. Full version. arXiv:2606.24675, doi:https://doi.org/10.48550/arXiv.2606.24675.
6. Bruno Courcelle. The monadic second-order logic of graphs. I. Recognizable sets of finite graphs. *Information and Computation*, 85(1):12–75, 1990. doi:10.1016/0890-5401(90)90043-H.
7. Antti Elonen, Ashwin Karthick Natarajan, Ibuki Kawamata, Lukas Oesinghaus, Abdulmelik Mohammed, Jani Seitsonen, Yuki Suzuki, Friedrich C. Simmel, Anton Kuzyk, and Pekka Orponen. Algorithmic design of 3D wireframe RNA polyhedra. *ACS Nano*, 16(10):16608–16616, 2022. doi:10.1021/acsnano.2c06035.
8. Cody Geary, Paul W. K. Rothemund, and Ebbe S. Andersen. A single-stranded architecture for cotranscriptional folding of RNA nanostructures. *Science*, 345(6198):799, August 2014. doi:10.1126/science.1253920.
9. Cody W. Geary and Ebbe S. Andersen. Design principles for single-stranded RNA origami structures. In Satoshi Murata and Satoshi Kobayashi, editors, *DNA Computing and Molecular Programming: 20th International Conference*, volume 8727 of *Lecture Notes in Computer Science*, pages 1–19. Springer International Publishing, 2014. doi:10.1007/978-3-319-11295-4_1.
10. Huong Luu and Marek Chrobak. Better hardness results for the minimum spanning tree congestion problem. *Algorithmica*, 87(1):148–165, January 2025. doi:10.1007/s00453-024-01278-5.
11. Pekka Orponen, Shinnosuke Seki, and Antti Elonen. Secondary structure design for cotranscriptional 3D RNA origami wireframes. In Josie Schaeffer and Fei Zhang, editors, *31st International Conference on DNA Computing and Molecular Programming (DNA 31)*, volume 347 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 6:1–6:18, Dagstuhl, Germany, 2025. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. doi:10.4230/LIPIcs.DNA.31.6.
12. Mikhail I. Ostrovskii. Minimal congestion trees. *Discrete Mathematics*, 285(1):219–226, August 2004. doi:10.1016/j.disc.2004.02.009.
13. Jaimie Marie Stewart. RNA nanotechnology on the horizon: Self-assembly, chemical modifications, and functional applications. *Current Opinion in Chemical Biology*, 81:102479, 2024. doi:https://doi.org/10.1016/j.cbpa.2024.102479.