

A Congestion Parameter for Depth-First Graph Traversals

Codaline Bourotte — École Normale Supérieure de Lyon, France

Gwendal Ducloz — École Normale Supérieure de Lyon, France

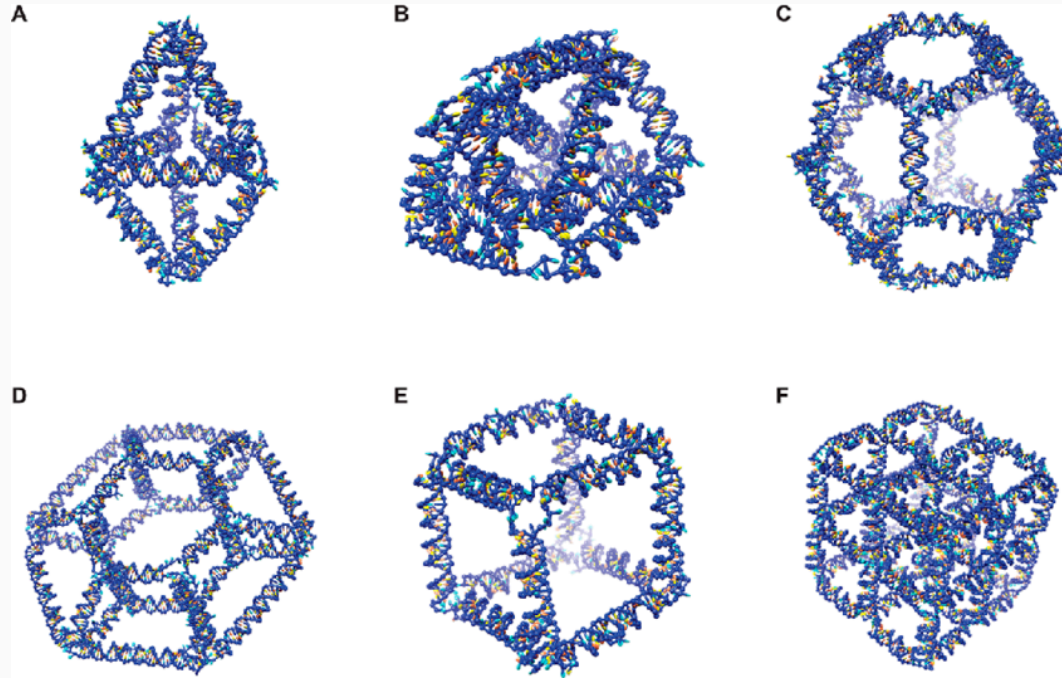
Pekka Orponen — Aalto University, Finland

Shinnosuke Seki — University of Electro-Communications, Japan

Tuesday 24 August — MFCS2026

Motivation 1

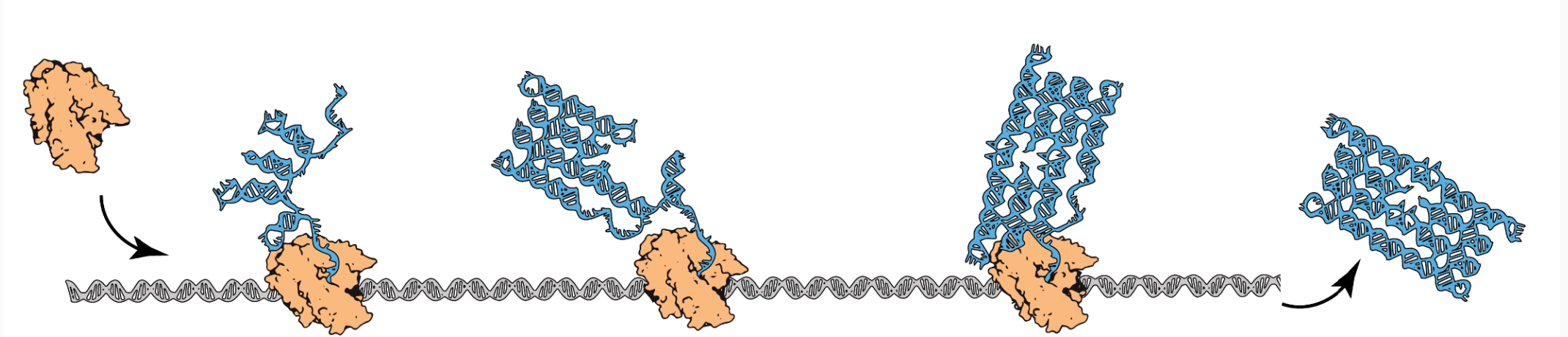
Motivation 1



© Li, M., Zheng, M., Wu, S. et al. In vivo production of RNA nanostructures via programmed folding of single-stranded RNAs;
licensed under Creative Commons License CC-BY

Motivation 2

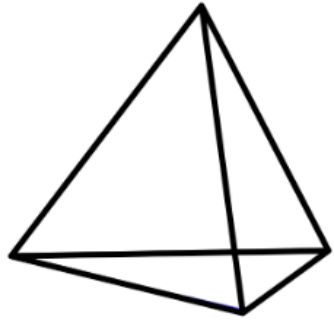
Motivation 2



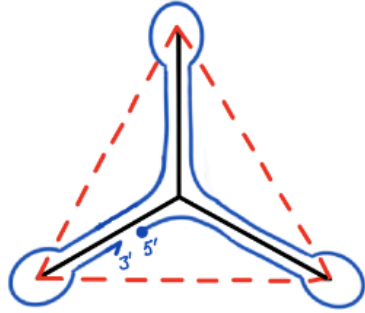
© Cody Geary, Pierre-Étienne Meunier, Nicolas Schabanel and Shinnosuke Seki; licensed under Creative Commons License CC-BY

The RNA Folding Procedure

The RNA Folding Procedure



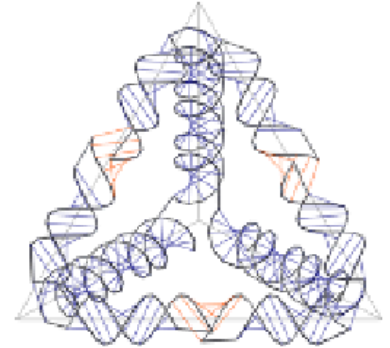
(a)



(b)

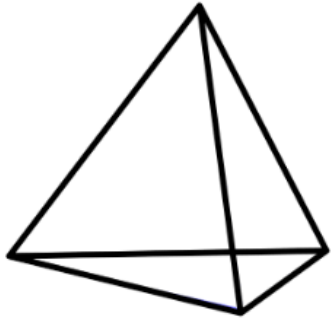


(c)

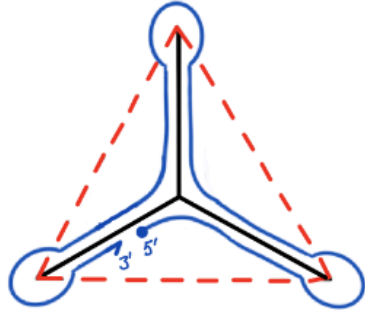


(d)

The RNA Folding Procedure



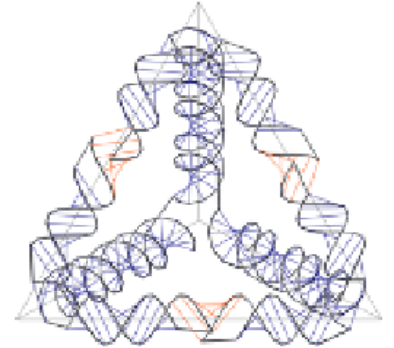
(a)



(b)



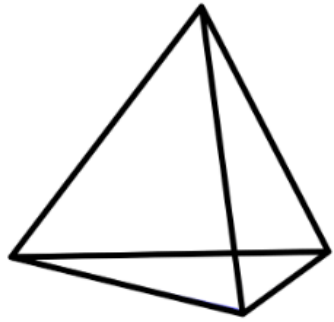
(c)



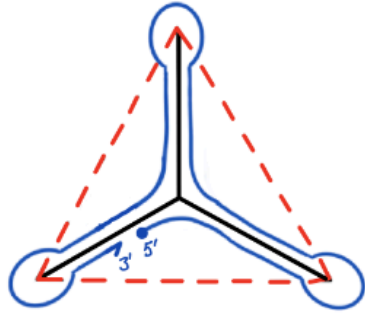
(d)

There are two problems:

The RNA Folding Procedure



(a)



(b)



(c)

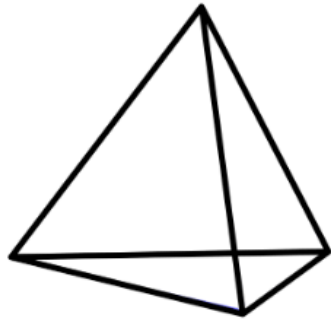


(d)

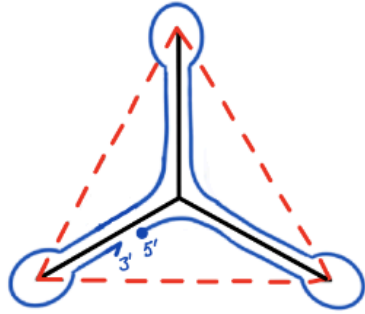
There are two problems:

- **Trapping:** The tree must be a DFS tree
-

The RNA Folding Procedure



(a)



(b)



(c)

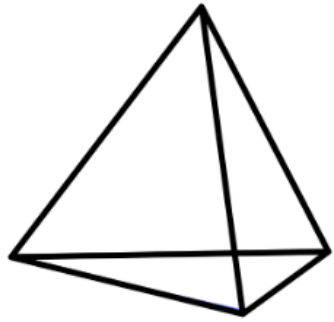


(d)

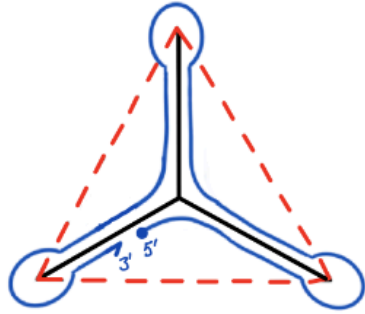
There are two problems:

- **Trapping:** The tree must be a DFS tree
- **Low KLX:** Only 6 half-edges can be open at the same time

The RNA Folding Procedure



(a)



(b)



(c)



(d)

There are two problems:

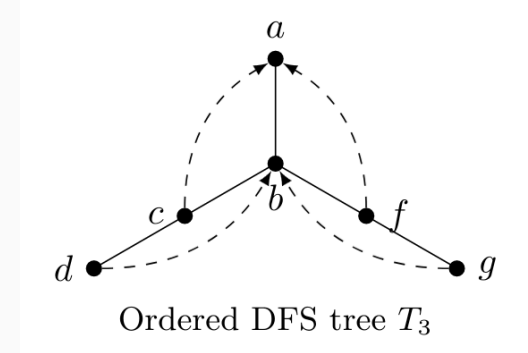
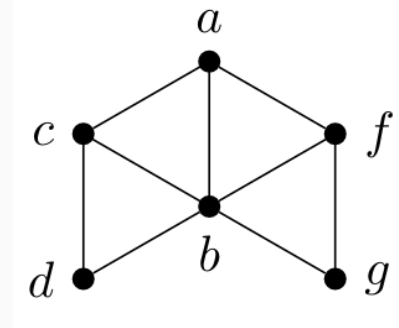
- **Trapping:** The tree must be a DFS tree
- **Low KLX:** Only 6 half-edges can be open at the same time

This leads to the definition of the **KLX number**

The KLX number

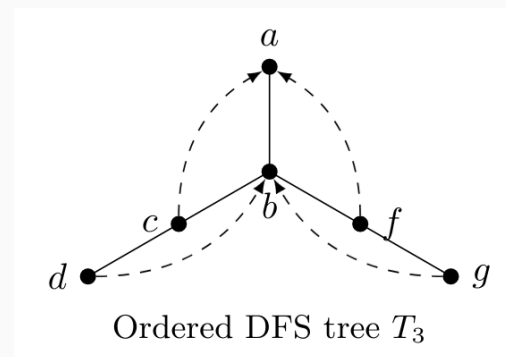
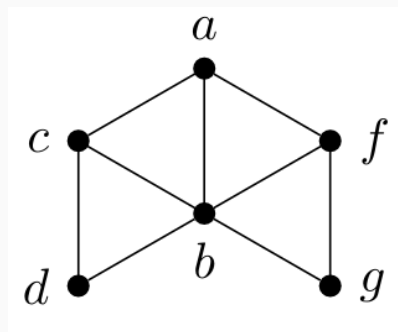
The KLX number

Let $G = (V_G, E_G)$ be a graph and $T = (V_T, E_T)$ an ordered DFS Spanning Tree.

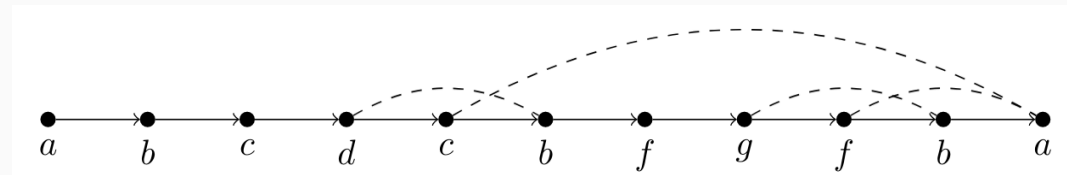


The KLX number

Let $G = (V_G, E_G)$ be a graph and $T = (V_T, E_T)$ an ordered DFS Spanning Tree.

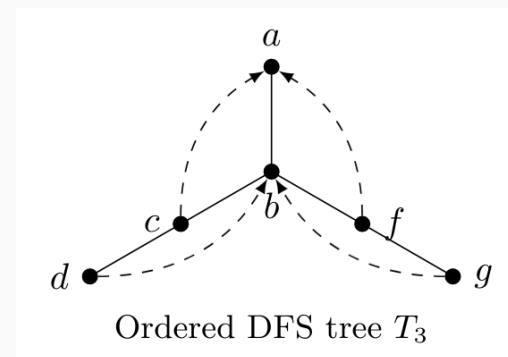
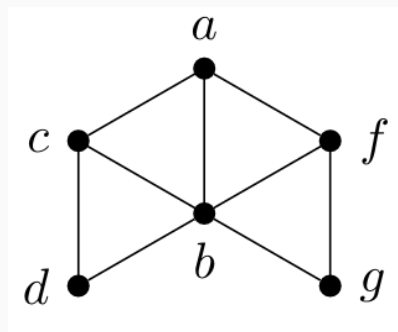


$\mathcal{O}(e)$ is the set of **open edges** (u, v) for $e \in E_T$, meaning that v has not been seen after coming back from fully visiting u .

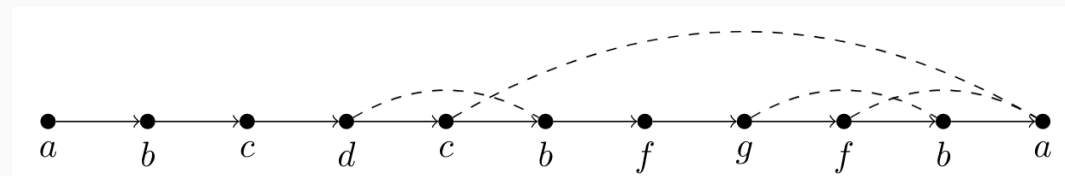


The KLX number

Let $G = (V_G, E_G)$ be a graph and $T = (V_T, E_T)$ an ordered DFS Spanning Tree.



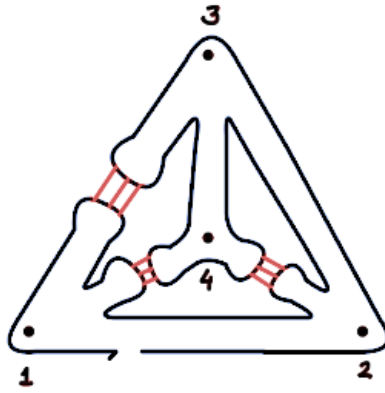
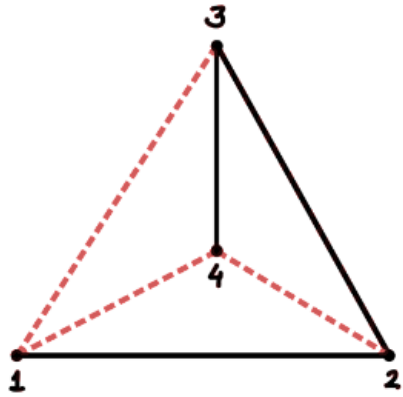
$\mathcal{O}(e)$ is the set of **open edges** (u, v) for $e \in E_T$, meaning that v has not been seen after coming back from fully visiting u .



$$\text{KLX}(G, T) = \max_{e \in E_T} |\mathcal{O}(e)|, \quad \text{and} \quad \text{KLX}(G) = \min_{T \text{ DFS of } G} \text{KLX}(G, T)$$

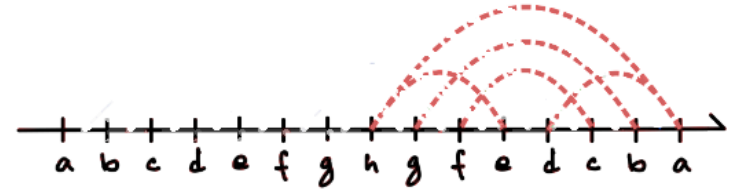
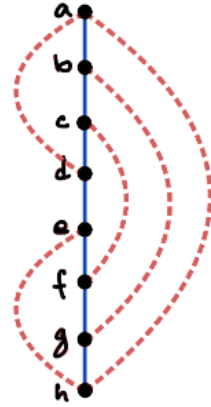
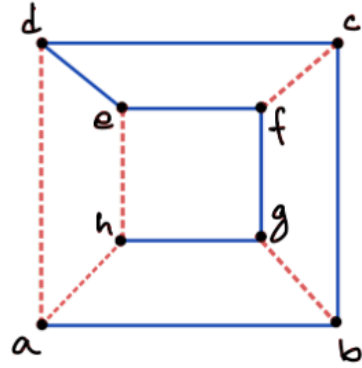
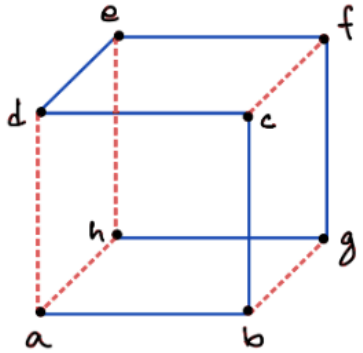
Example 1

Example 1

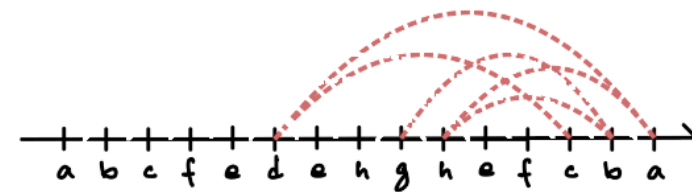
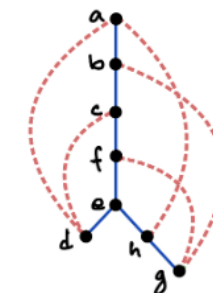
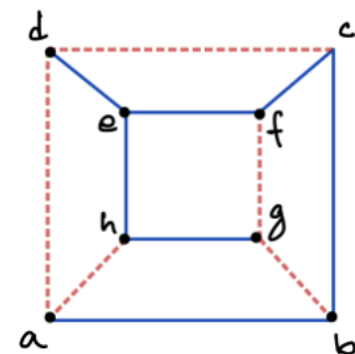
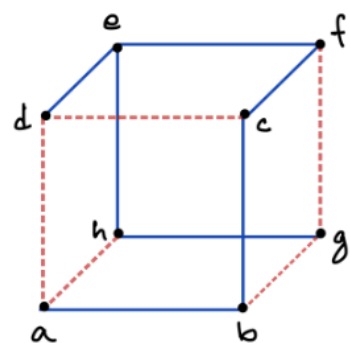
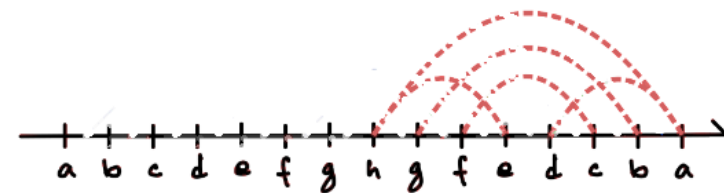
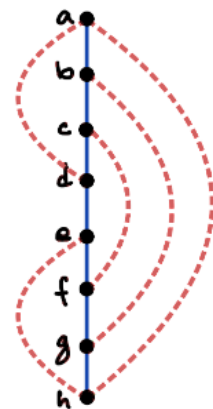
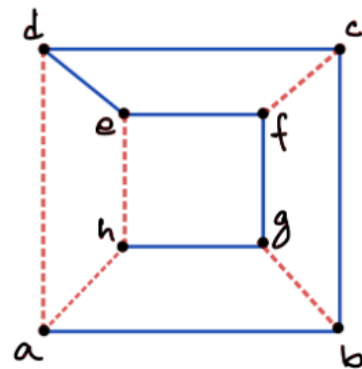
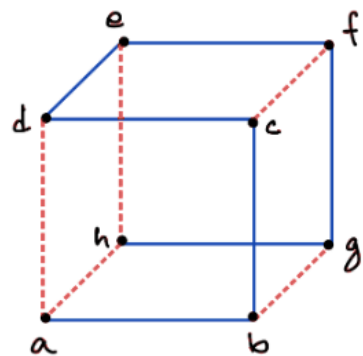


Example 2

Example 2



Example 2



Links with connectedness

Links with connectedness

A graph is k -connected if removal of at most $k - 1$ vertices leaves it connected. We denote by $\kappa(G)$ the biggest k such that G is k -connected.

Links with connectedness

A graph is k -connected if removal of at most $k - 1$ vertices leaves it connected. We denote by $\kappa(G)$ the biggest k such that G is k -connected.

2 connected component Lemma

If B is a 2 connected component of G then $\text{KLX}(B) \leq \text{KLX}(G)$.

Links with connectedness

A graph is k -connected if removal of at most $k - 1$ vertices leaves it connected. We denote by $\kappa(G)$ the biggest k such that G is k -connected.

2 connected component Lemma

If B is a 2 connected component of G then $\text{KLX}(B) \leq \text{KLX}(G)$.

Proof. For T a DFS tree of G of minimal KLX, then $T[V_B]$ is a DFS tree of B of lower KLX.

Links with connectedness

A graph is k -connected if removal of at most $k - 1$ vertices leaves it connected. We denote by $\kappa(G)$ the biggest k such that G is k -connected.

2 connected component Lemma

If B is a 2 connected component of G then $\text{KLX}(B) \leq \text{KLX}(G)$.

Proof. For T a DFS tree of G of minimal KLX, then $T[V_B]$ is a DFS tree of B of lower KLX.

k -connectedness Lemma

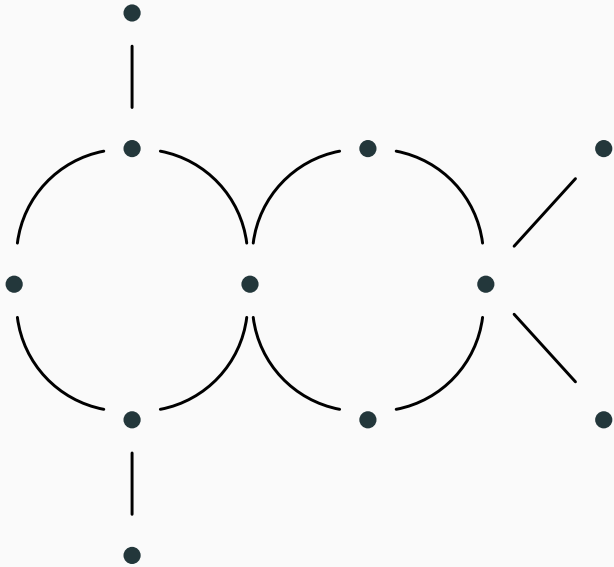
If G is k -connected for $k \geq 2$, then $\text{KLX}(G) \geq k - 2$

Characterisation of $KLX \leq 1$

A graph G has $KLX(G) \leq 1$ if and only if it is a cactus graph (connected without two edge-touching cycles).

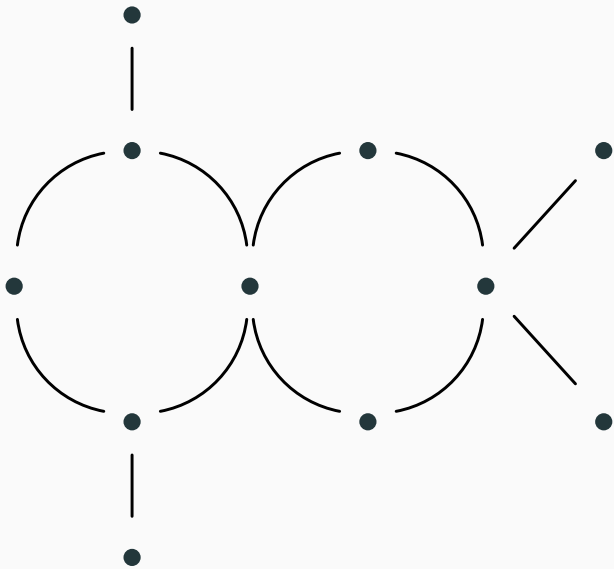
Characterisation of $KLX \leq 1$

A graph G has $KLX(G) \leq 1$ if and only if it is a cactus graph (connected without two edge-touching cycles).



Characterisation of $KLX \leq 1$

A graph G has $KLX(G) \leq 1$ if and only if it is a cactus graph (connected without two edge-touching cycles).

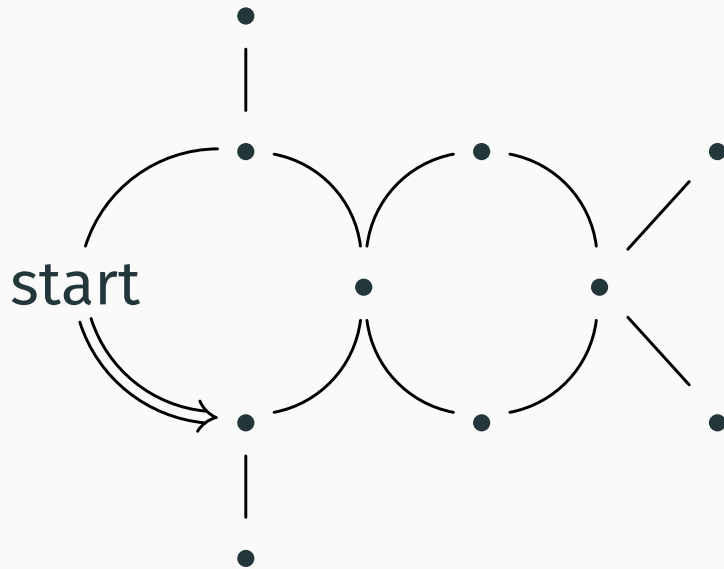


Proof.

- $(\implies)$ By the above lemmas
- $(\impliedby)$ We can form a DFS by always exploring outgoing edges before finishing a cycle

Characterisation of $KLX \leq 1$

A graph G has $KLX(G) \leq 1$ if and only if it is a cactus graph (connected without two edge-touching cycles).

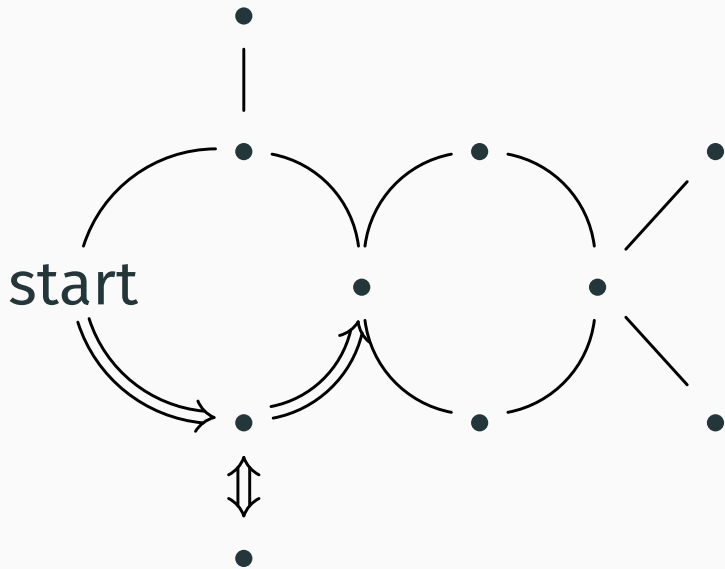


Proof.

- ($\implies$) By the above lemmas
- ($\impliedby$) We can form a DFS by always exploring outgoing edges before continuing a cycle

Characterisation of $KLX \leq 1$

A graph G has $KLX(G) \leq 1$ if and only if it is a cactus graph (connected without two edge-touching cycles).

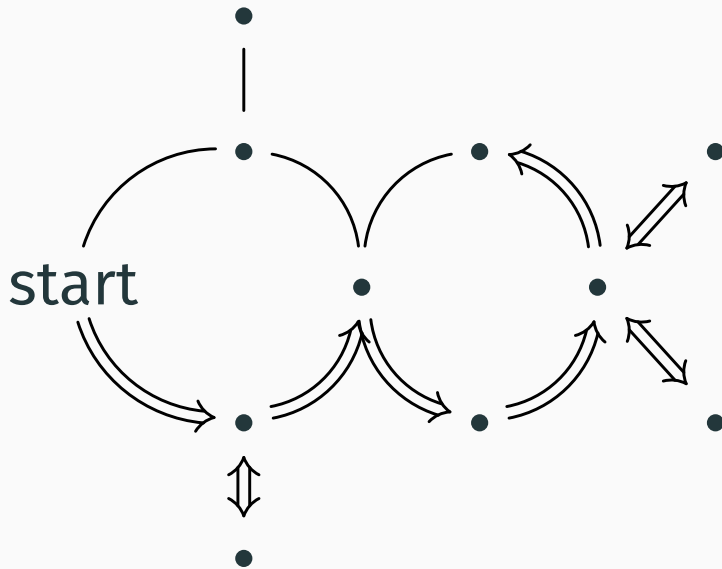


Proof.

- ($\implies$) By the above lemmas
- ($\impliedby$) We can form a DFS by always exploring outgoing edges before continuing a cycle

Characterisation of $KLX \leq 1$

A graph G has $KLX(G) \leq 1$ if and only if it is a cactus graph (connected without two edge-touching cycles).

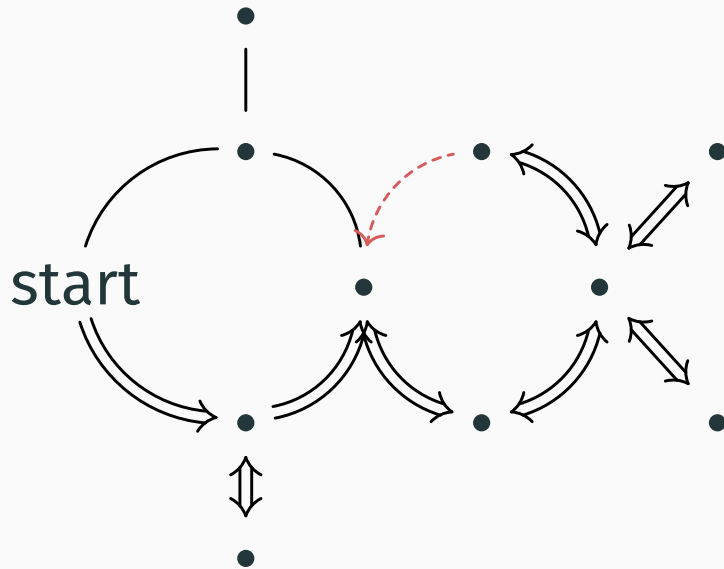


Proof.

- ($\implies$) By the above lemmas
- ($\impliedby$) We can form a DFS by always exploring outgoing edges before continuing a cycle

Characterisation of $KLX \leq 1$

A graph G has $KLX(G) \leq 1$ if and only if it is a cactus graph (connected without two edge-touching cycles).

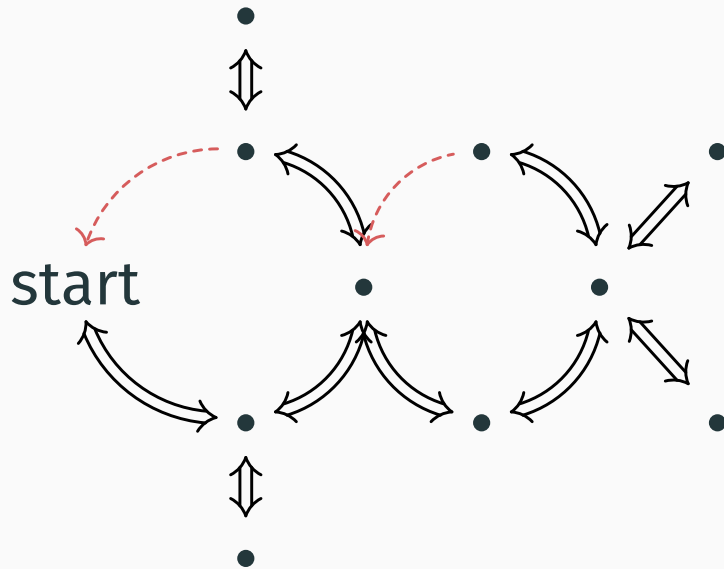


Proof.

- ($\implies$) By the above lemmas
- ($\impliedby$) We can form a DFS by always exploring outgoing edges before continuing a cycle

Characterisation of $KLX \leq 1$

A graph G has $KLX(G) \leq 1$ if and only if it is a cactus graph (connected without two edge-touching cycles).



Proof.

- ($\implies$) By the above lemmas
- ($\impliedby$) We can form a DFS by always exploring outgoing edges before continuing a cycle

Computing the KLX number

Computing the KLX number is **NP-Hard** [Orponen et al. 2025]

Computing the KLX number is **NP-Hard** [Orponen et al. 2025]

Is testing the KLX in P? Is it FPT?

Computing the KLX number is **NP-Hard** [Orponen et al. 2025]

Is testing the KLX in P? Is it FPT?

Our results:

Computing the KLX number is **NP-Hard** [Orponen et al. 2025]

Is testing the KLX in P? Is it FPT?

Our results:

- An explicit algorithm to test if the KLX-number is ≤ 2 ,
-
-

Computing the KLX number is **NP-Hard** [Orponen et al. 2025]

Is testing the KLX in P? Is it FPT?

Our results:

- An explicit algorithm to test if the KLX-number is ≤ 2 ,
- Using MSO_2 , testing if the KLX-number is $\leq k$ is P for fixed $k \in \mathbb{N}$,
-

Computing the KLX number is **NP-Hard** [Orponen et al. 2025]

Is testing the KLX in P? Is it FPT?

Our results:

- An explicit algorithm to test if the KLX-number is ≤ 2 ,
- Using MSO_2 , testing if the KLX-number is $\leq k$ is P for fixed $k \in \mathbb{N}$,
- There is a FPT algorithm parametrized by the KLX number.

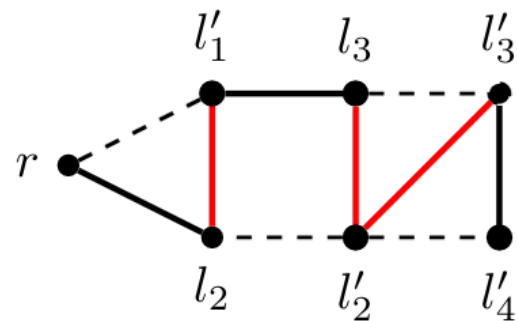
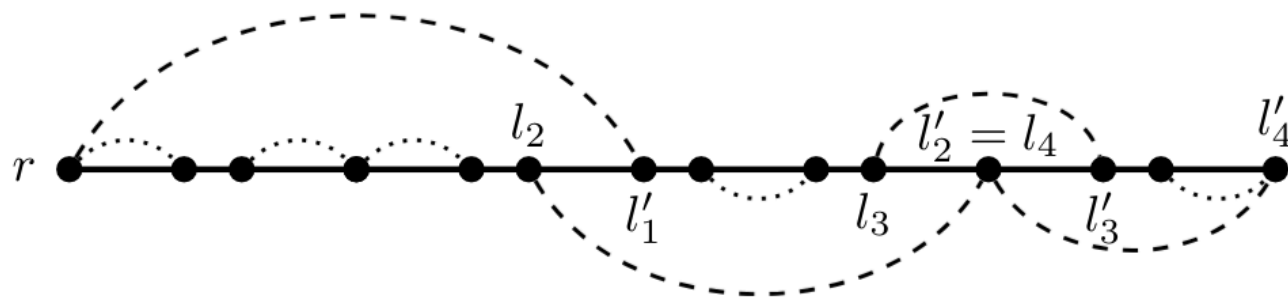
Explicit algorithm for $KLX \leq 2$

Explicit algorithm for $KLX \leq 2$

A biconnected component has $KLX = 2$ iff it admits a **ladder pattern** along with extra cycles placed in an alternating way on this ladder.

Explicit algorithm for $KLX \leq 2$

A biconnected component has $KLX = 2$ iff it admits a **ladder pattern** along with extra cycles placed in an alternating way on this ladder.



Links with tree-width

Links with tree-width

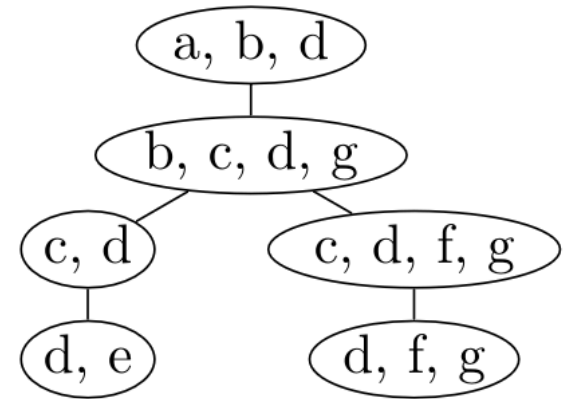
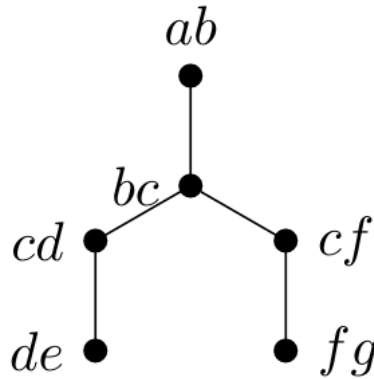
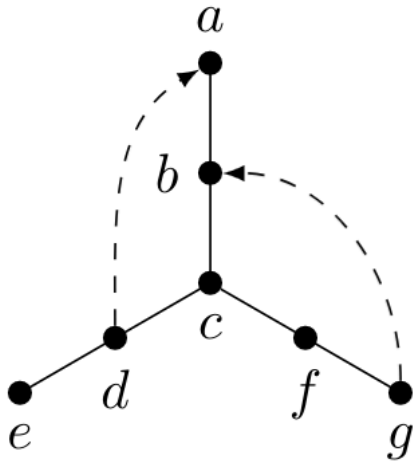
Theorem

For all graph G , $TW(G) \leq K LX(G) + 1$

Links with tree-width

Theorem

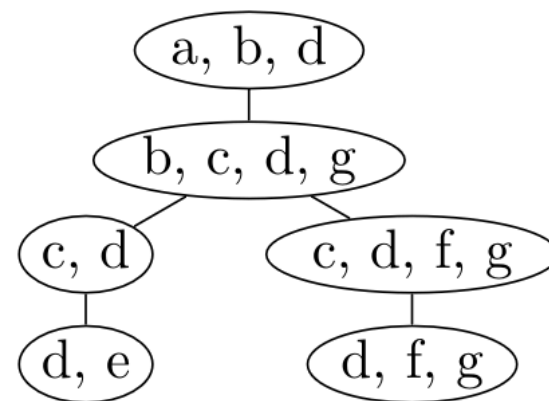
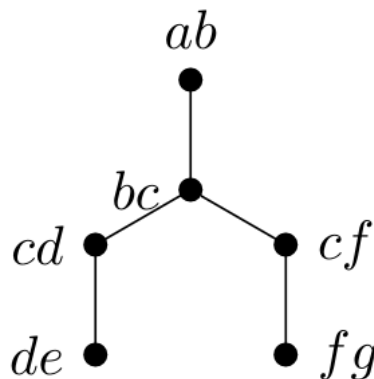
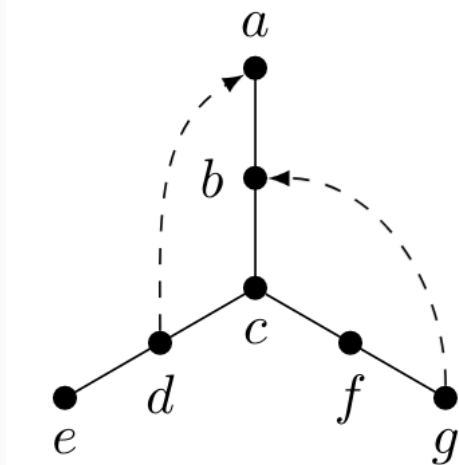
For all graph G , $TW(G) \leq K LX(G) + 1$



Links with tree-width

Theorem

For all graph G , $TW(G) \leq KLX(G) + 1$



Good news: We just need a MSO_2 formula for testing $KLX \leq k$!

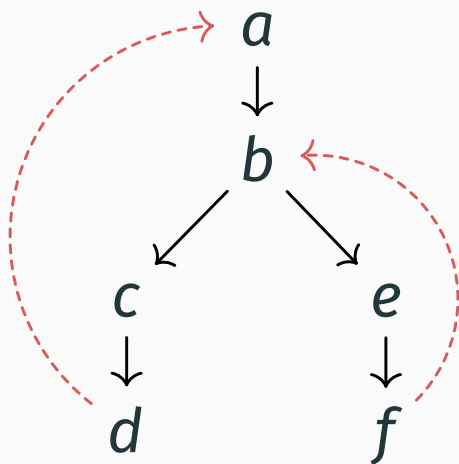
Problem for encoding MSO_2 ...

Problem for encoding MSO_2 ...

The KLX number of a DFS tree **depends on the order of visiting children**

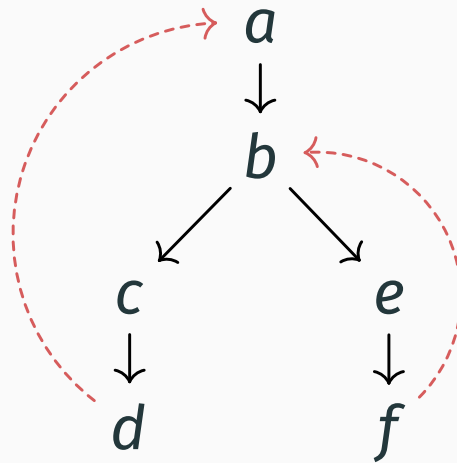
Problem for encoding MSO_2 ...

The KLX number of a DFS tree **depends on the order of visiting children**



Problem for encoding MSO_2 ...

The KLX number of a DFS tree **depends on the order of visiting children**

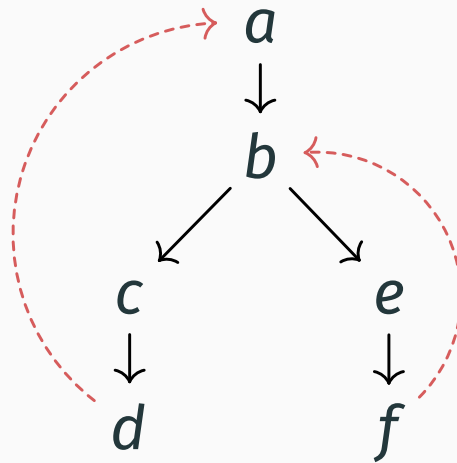


$\langle a, b, c, \overline{d, c, b}, \overline{e, f, e, b}, a \rangle$

$\langle a, b, e, \overline{f, e, b}, \overline{c, d, c, b}, a \rangle$

Problem for encoding MSO_2 ...

The KLX number of a DFS tree **depends on the order of visiting children**



$\langle a, b, c, \overline{d, c, b, e, \overline{f, e, b}}, a \rangle$

$\langle a, b, e, \overline{f, e, b}, \overline{c, d, c, b}, a \rangle$

This means that we would need an **unbounded amount of information** to be stored at every vertex...

How to solve it

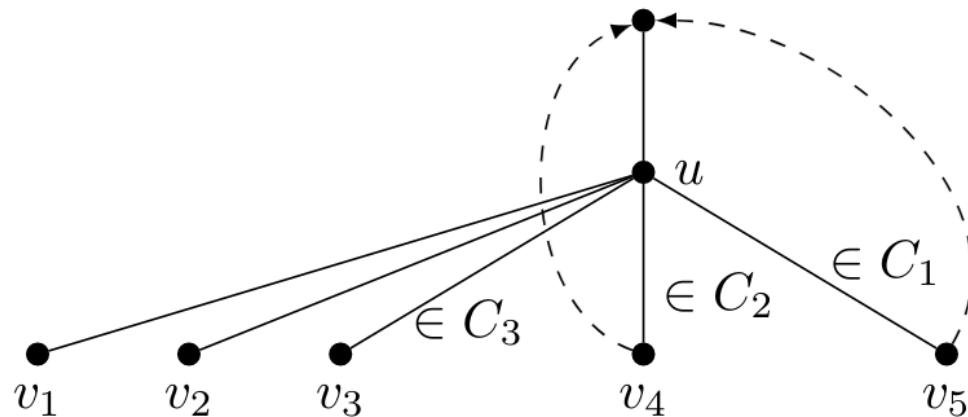
Theorem

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

How to solve it

Theorem

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



The MSO_2 reduction

The MSO_2 reduction

1. Create a formula $\Psi(T, C_1, \dots, C_k)$ that encodes a DFS tree T of E with, for every vertex, an order on k of the children.
- 2.
- 3.
- 4.

The MSO_2 reduction

1. Create a formula $\Psi(T, C_1, \dots, C_k)$ that encodes a DFS tree T of E with, for every vertex, an order on k of the children.
2. Create a formula $\text{IsOpen}(e, X)$ that tests if $X = \mathcal{O}(e)$ (this will use the DFS tree encoding), denoted X_e in the following steps.
- 3.
- 4.

The MSO_2 reduction

1. Create a formula $\Psi(T, C_1, \dots, C_k)$ that encodes a DFS tree T of E with, for every vertex, an order on k of the children.
2. Create a formula $\text{IsOpen}(e, X)$ that tests if $X = \mathcal{O}(e)$ (this will use the DFS tree encoding), denoted X_e in the following steps.
3. Create a cover of all edges $V_1, \dots, V_k$.
- 4.

The MSO_2 reduction

1. Create a formula $\Psi(T, C_1, \dots, C_k)$ that encodes a DFS tree T of E with, for every vertex, an order on k of the children.
2. Create a formula $\text{IsOpen}(e, X)$ that tests if $X = \mathcal{O}(e)$ (this will use the DFS tree encoding), denoted X_e in the following steps.
3. Create a cover of all edges $V_1, \dots, V_k$.
4. Verify that $(X_e)_{e \in T}$ is a disjoint spanning of $V_1, \dots, V_k$.

The MSO_2 reduction

1. Create a formula $\Psi(T, C_1, \dots, C_k)$ that encodes a DFS tree T of E with, for every vertex, an order on k of the children.
2. Create a formula $\text{IsOpen}(e, X)$ that tests if $X = \mathcal{O}(e)$ (this will use the DFS tree encoding), denoted X_e in the following steps.
3. Create a cover of all edges $V_1, \dots, V_k$.
4. Verify that $(X_e)_{e \in T}$ is a disjoint spanning of $V_1, \dots, V_k$.

Since a $e \in G_E \setminus T_E$ can be in at most k V_i , there are at most k open set while traversing it

The MSO_2 reduction

1. Create a formula $\Psi(T, C_1, \dots, C_k)$ that encodes a DFS tree T of E with, for every vertex, an order on k of the children.
2. Create a formula $\text{IsOpen}(e, X)$ that tests if $X = \mathcal{O}(e)$ (this will use the DFS tree encoding), denoted X_e in the following steps.
3. Create a cover of all edges $V_1, \dots, V_k$.
4. Verify that $(X_e)_{e \in T}$ is a disjoint spanning of $V_1, \dots, V_k$.

Since a $e \in G_E \setminus T_E$ can be in at most k V_i , there are at most k open set while traversing it

Therefore testing if $\text{KLX} \leq k$ is FPT

Conclusion

Our results:

Conclusion

Our results:

- An explicit algorithm to test if the KLX-number is ≤ 2 ,
-
-

Conclusion

Our results:

- An explicit algorithm to test if the KLX-number is ≤ 2 ,
- Using MSO_2 , testing if the KLX-number is $\leq k$ is P for fixed $k \in \mathbb{N}$,
-

Conclusion

Our results:

- An explicit algorithm to test if the KLX-number is ≤ 2 ,
- Using MSO_2 , testing if the KLX-number is $\leq k$ is P for fixed $k \in \mathbb{N}$,
- There is a FPT algorithm parametrized by the KLX number.

Conclusion

Our results:

- An explicit algorithm to test if the KLX-number is ≤ 2 ,
- Using MSO_2 , testing if the KLX-number is $\leq k$ is P for fixed $k \in \mathbb{N}$,
- There is a FPT algorithm parametrized by the KLX number.

Open Questions:

Conclusion

Our results:

- An explicit algorithm to test if the KLX-number is ≤ 2 ,
- Using MSO_2 , testing if the KLX-number is $\leq k$ is P for fixed $k \in \mathbb{N}$,
- There is a FPT algorithm parametrized by the KLX number.

Open Questions:

- Can we find explicit algorithm for testing the KLX number?
-

Conclusion

Our results:

- An explicit algorithm to test if the KLX-number is ≤ 2 ,
- Using MSO_2 , testing if the KLX-number is $\leq k$ is P for fixed $k \in \mathbb{N}$,
- There is a FPT algorithm parametrized by the KLX number.

Open Questions:

- Can we find explicit algorithm for testing the KLX number?
- How does it relate to other graph parameters?

Thanks for listening.

Full version available on arXiv.