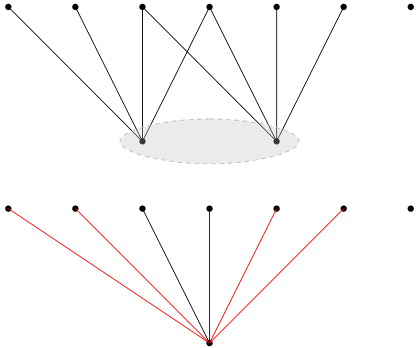
Computing (Oriented) Twin-width

Mathis Rocton

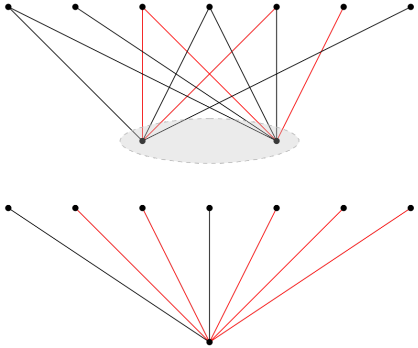
TU Wien

November 20, 2025

Contracting Vertices



Contracting Vertices



Contraction Sequences

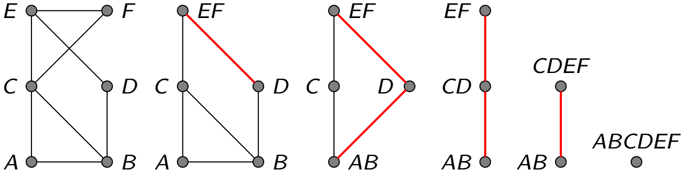


Figure from Jakub Balabán.

The Twin-Width of a Graph

The *twin-width* of a graph G is the minimum width of a contraction sequence, over all valid contraction sequences from G to K_1.

Using Twin-Width to Solve FO

Theorem (Bonnet, Kim, Thomassé, Watrigant; 2020)

*Provided a **contraction sequence** of G of width d, evaluating a formula φ expressible in **First Order Logic (FO)** on G can be done in time $f(d, |\varphi|) \cdot |V(G)|$ for a computable function f.*

Using Twin-Width to Solve FO

Theorem (Bonnet, Kim, Thomassé, Watrigant; 2020)

Provided a ***contraction sequence*** *of G of width d, evaluating a formula φ expressible in* ***First Order Logic (FO)*** *on G can be done in time $f(d, |\varphi|) \cdot |V(G)|$ for a computable function f.*

To this date, we do **not** have any efficient way of computing contraction sequences of (quasi) optimal width!

Using Twin-Width to Solve FO

Theorem (Bonnet, Kim, Thomassé, Watrigant; 2020)

Provided a ***contraction sequence*** *of G of width d, evaluating a formula φ expressible in* ***First Order Logic (FO)*** *on G can be done in time $f(d, |\varphi|) \cdot |V(G)|$ for a computable function f.*

To this date, we do **not** have any efficient way of computing contraction sequences of (quasi) optimal width!

Theorem (Bergé, Bonnet, Déprés; 2022)

Deciding whether the twin-width of a graph is at most 4 is NP-complete.

Computing Twin-Width

Is there an algorithm that given an n-vertex graph G and $k \in \mathbb{N}$, runs in time $f(k) \cdot n^{\mathcal{O}(1)}$ and either correctly reports that $\text{tww}(G) \geq k$ or outputs a contr. sequence of width at most $g(k)$?

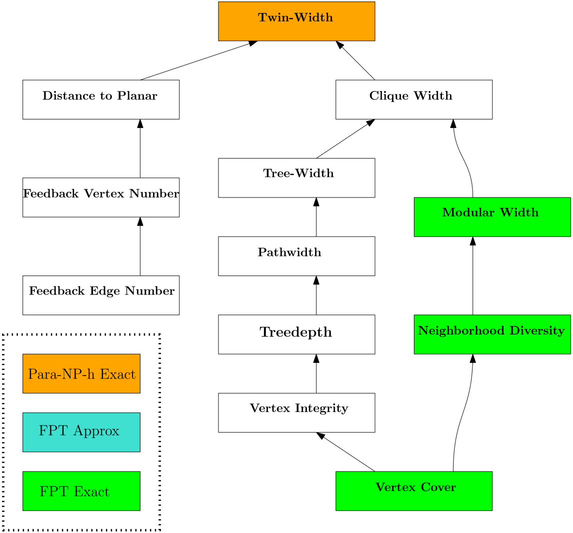
- At this moment wide open!

Computing Twin-Width

Is there an algorithm that given an n-vertex graph G and $k \in \mathbb{N}$, runs in time $f(k) \cdot n^{\mathcal{O}(1)}$ and either correctly reports that $\text{tww}(G) \geq k$ or outputs a contr. sequence of width at most $g(k)$?

- At this moment wide open!
- What about using more restrictive parameters?
- Then maybe even *exact* FPT algorithms would be possible!

State of the Art



State of the Art

Theorem (+1-approximation for FEN)

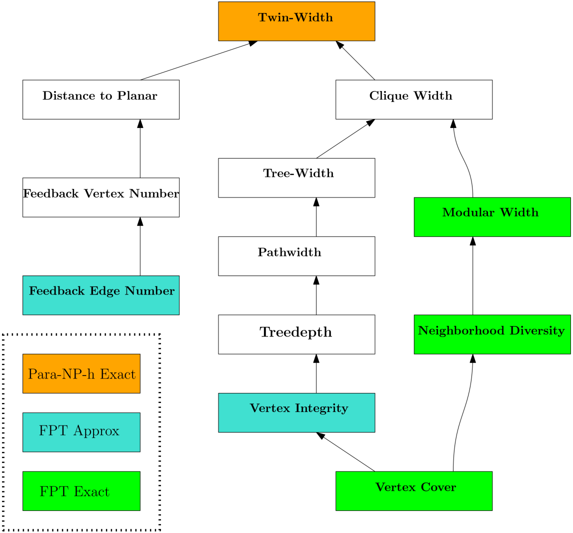
It is FPT to compute a contraction sequence for G of width at most $\mathrm{tww}(G) + 1$*, parameterized by the Feedback Edge Number.*

Theorem (2-approximation for VI)

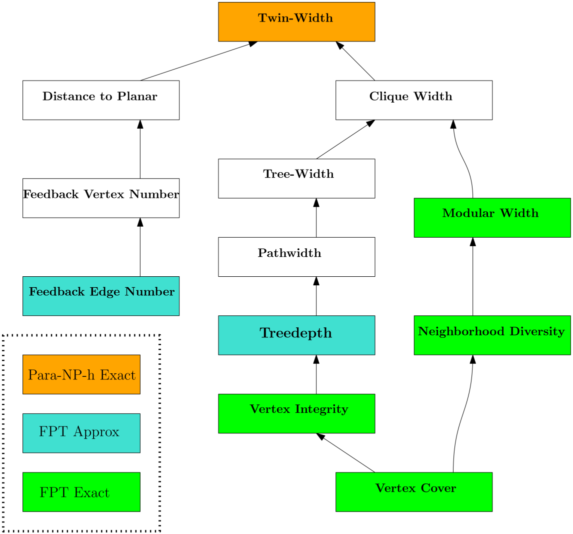
It is FPT to compute a contraction sequence for G of width at most $2 \cdot \mathrm{tww}(G)$*, parameterized by the Vertex Integrity.*

Balabán, Ganian, R., *SIDMA* (2025)

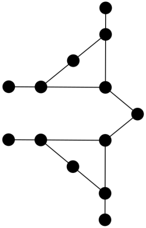
State of the Art



State of the Art

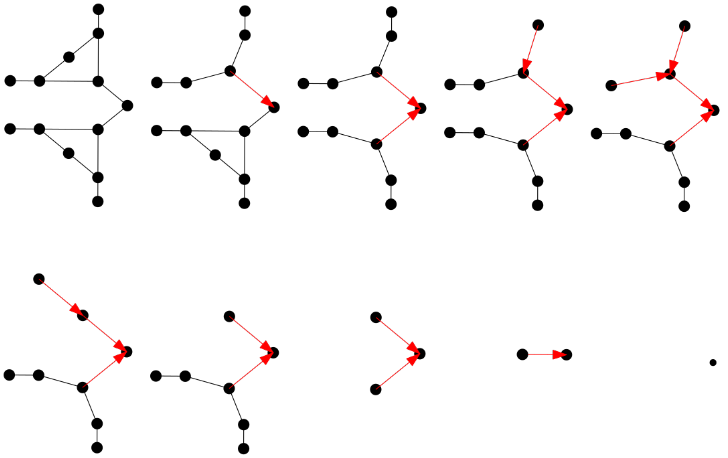


Oriented Twin-Width



- Related to Twin-Width
- Refines the error edges with orientation
- Always smaller than Twin-Width

Oriented Twin-Width



A Twin to Twin-width?

Theorem (Bonnet, Kim, Reinald, Thomassé, 2022)

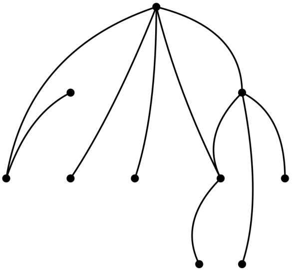
For every graph G, $\mathrm{otww}(G) \leq \mathrm{tww}(G) \leq 2^{2^{\mathcal{O}(\mathrm{otww}(G))}}$.

Theorem (Combining results from Twin-Width I, IV and VI)

There is an FPT algorithm that takes as input a graph G together with a contraction sequence of oriented width $f(\mathrm{otww}(G))$ and outputs a contraction sequence of width at most $2^{2^{2^{\mathcal{O}(f(\mathrm{tww}(G)))}}}$.

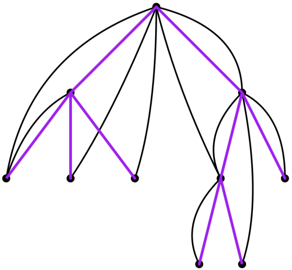
Treedepth

The treedepth of G is the minimum height of a forest $\mathcal{F}$ on V(G), such that each edge of G connects two vertices with a ancestor/descendant relation in $\mathcal{F}$.



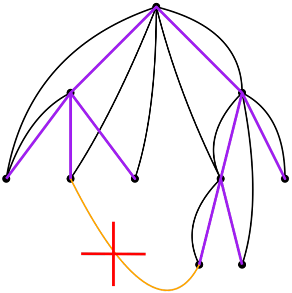
Treedepth

The treedepth of G is the minimum height of a forest $\mathcal{F}$ on V(G), such that each edge of G connects two vertices with a ancestor/descendant relation in $\mathcal{F}$.

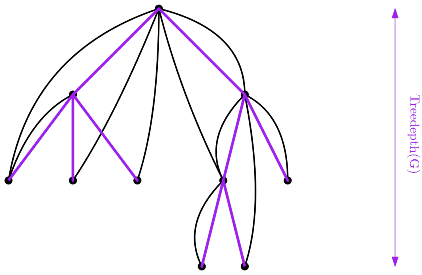


Treedepth

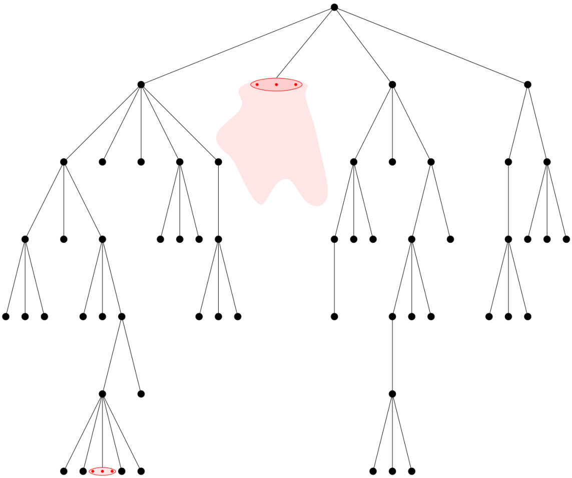
The treedepth of G is the minimum height of a forest $\mathcal{F}$ on V(G), such that each edge of G connects two vertices with a ancestor/descendant relation in $\mathcal{F}$.



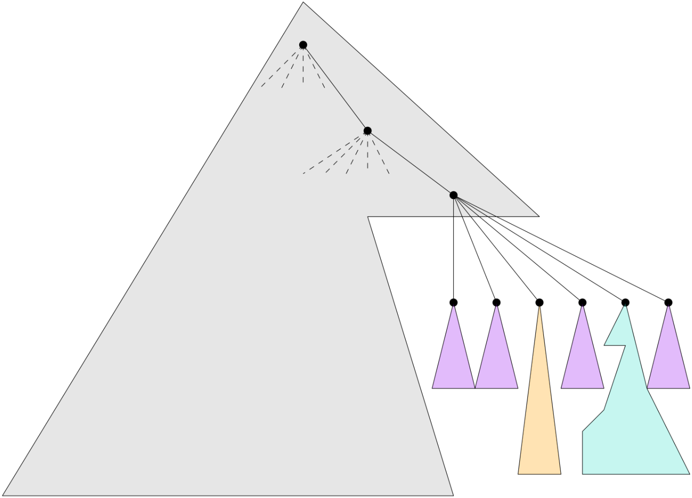
The treedepth of G is the minimum height of a forest $\mathcal{F}$ on V(G), such that each edge of G connects two vertices with a ancestor/descendant relation in $\mathcal{F}$.



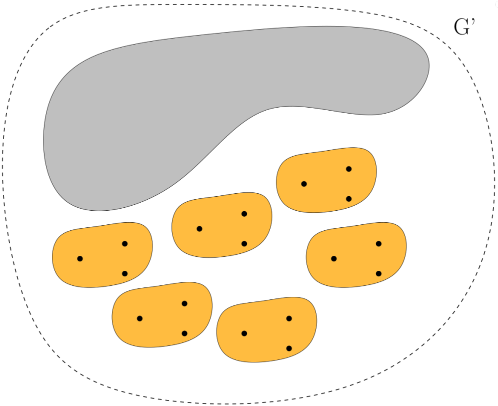
Pruning Principle



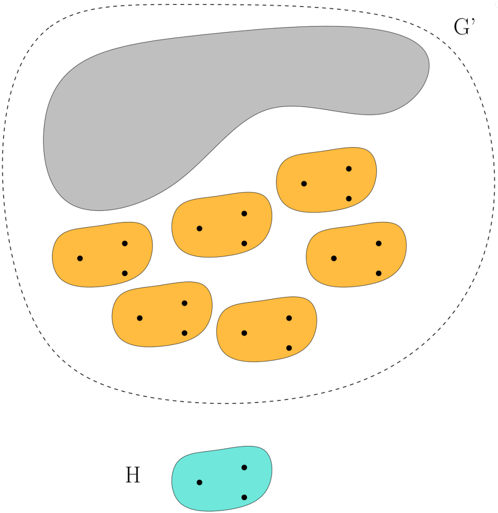
Pruning Principle



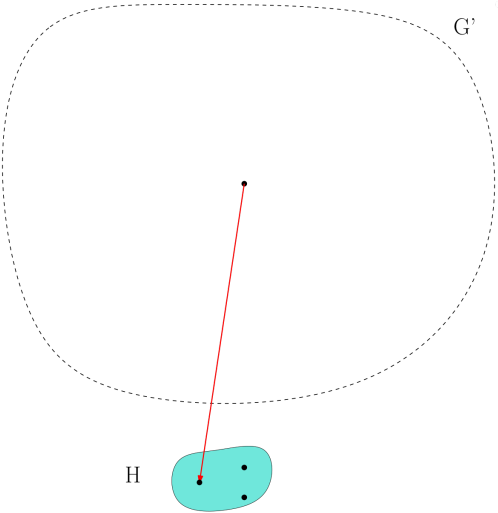
Inserting Back a Pruned Subgraph



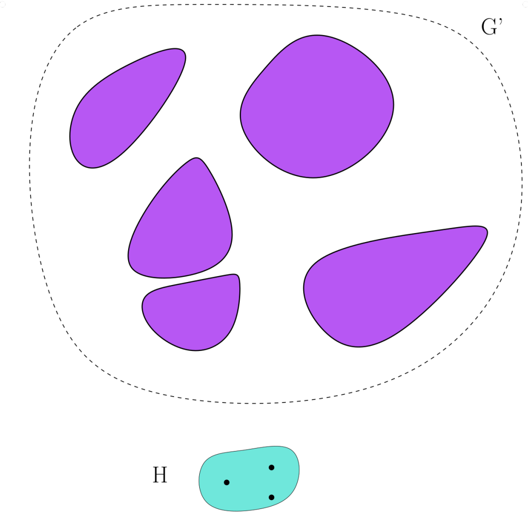
Inserting Back a Pruned Subgraph



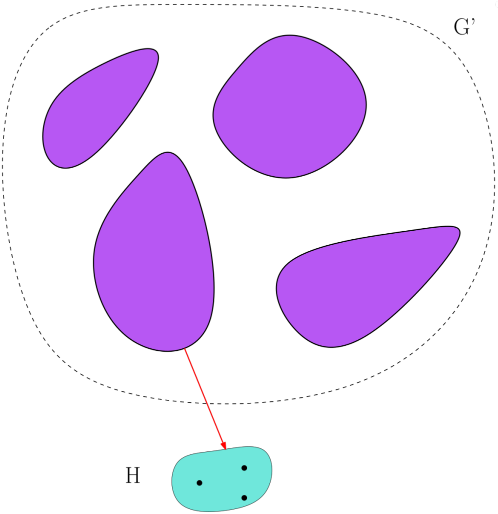
Inserting Back a Pruned Subgraph



Inserting Back a Pruned Subgraph



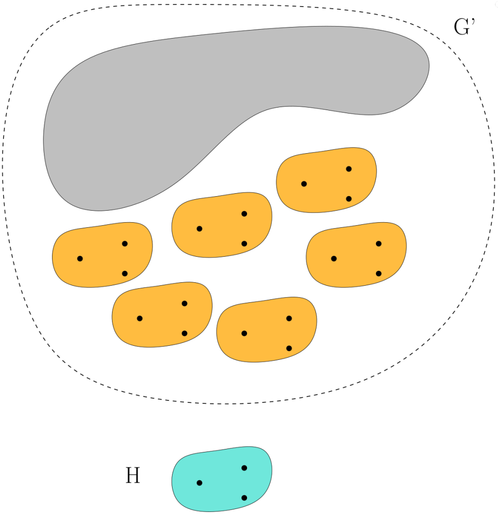
Inserting Back a Pruned Subgraph



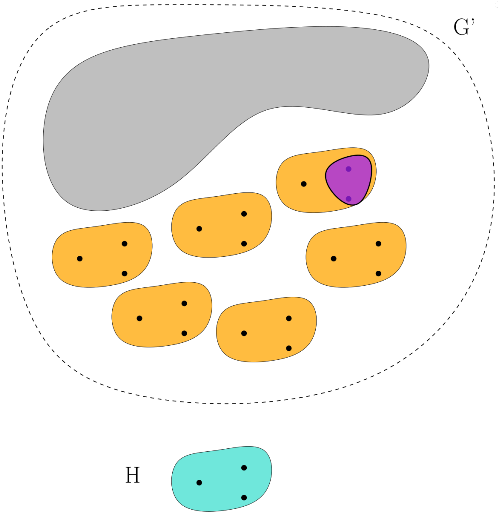
Inserting Back a Pruned Subgraph

We call **indifferent** all the trigraphs in the sequence such that no red arc goes to H.

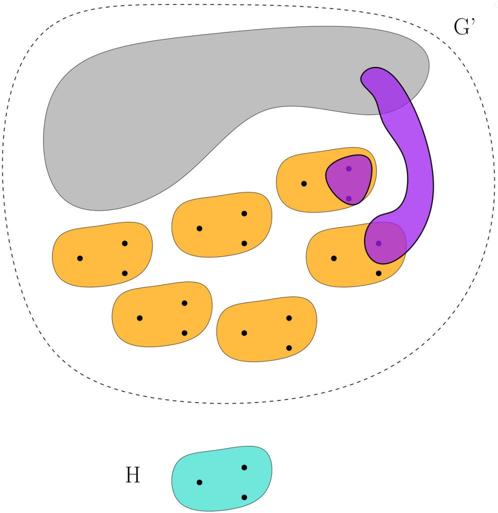
Inserting Back a Pruned Subgraph



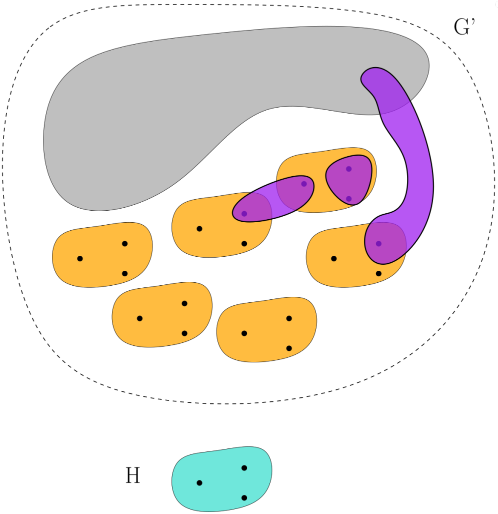
Inserting Back a Pruned Subgraph



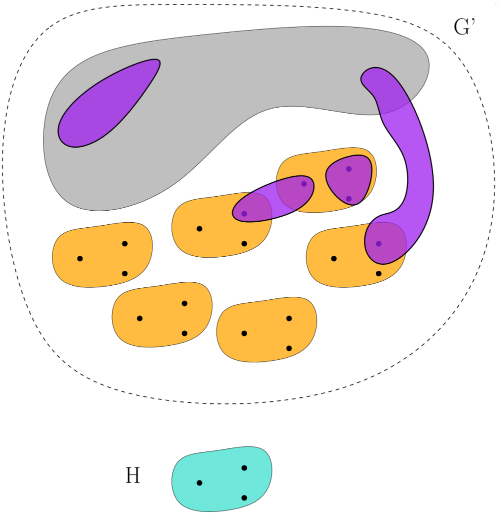
Inserting Back a Pruned Subgraph



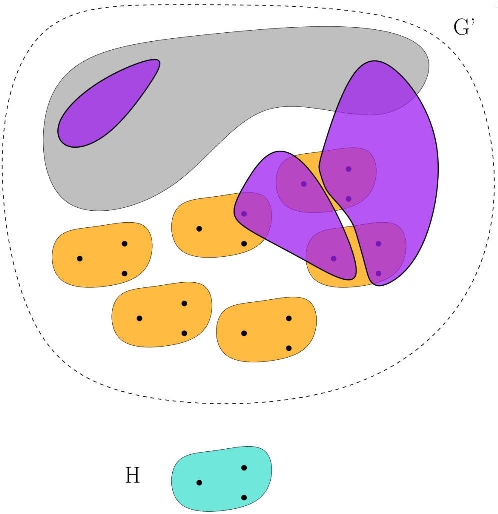
Inserting Back a Pruned Subgraph



Inserting Back a Pruned Subgraph



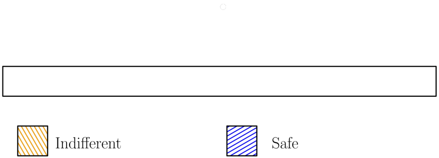
Inserting Back a Pruned Subgraph



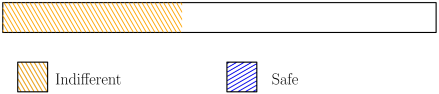
Inserting Back a Pruned Subgraph

We call **safe** any trigraph in the sequence such that two *twin-blocks of H* are **merged together**.

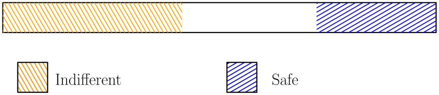
Inserting Back a Pruned Subgraph



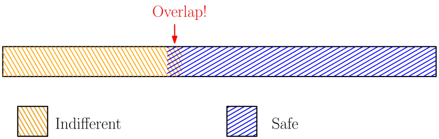
Inserting Back a Pruned Subgraph



Inserting Back a Pruned Subgraph



Inserting Back a Pruned Subgraph

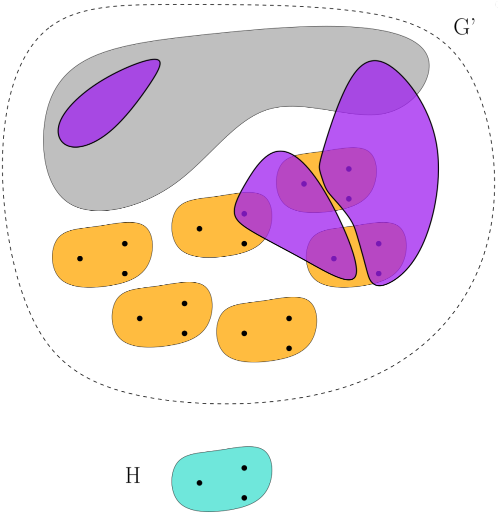


Inserting Back a Pruned Subgraph

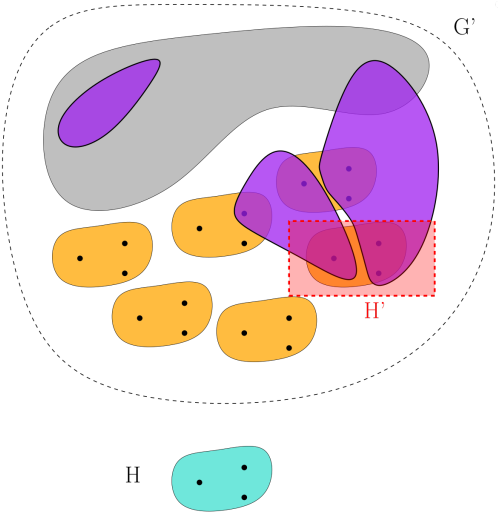
Lemma

There is always a trigraph which is both Indifferent and Safe.

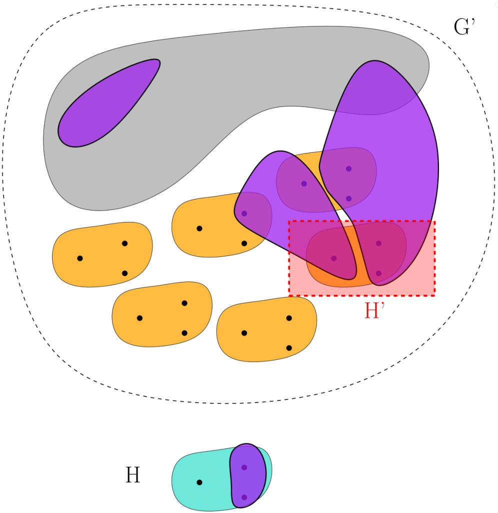
Inserting Back a Pruned Subgraph



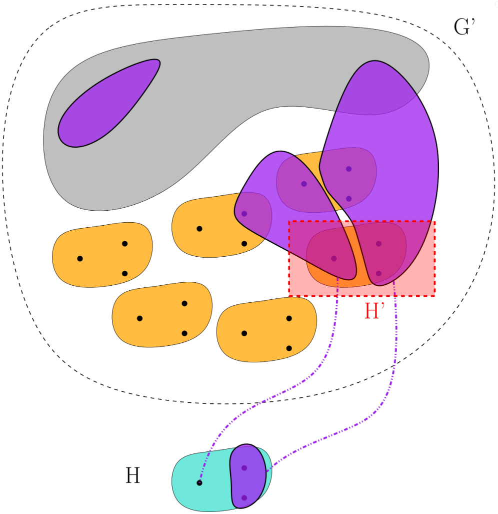
Inserting Back a Pruned Subgraph



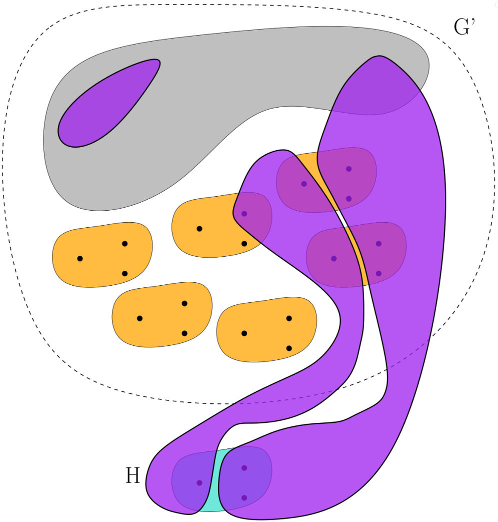
Inserting Back a Pruned Subgraph



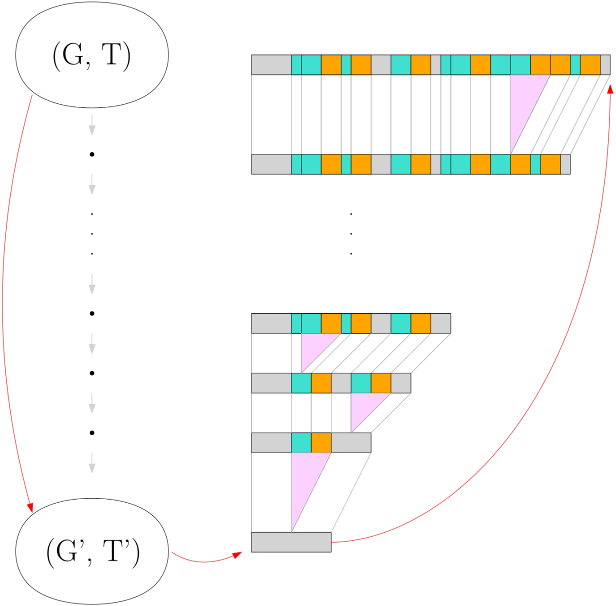
Inserting Back a Pruned Subgraph



Inserting Back a Pruned Subgraph



Dealing with Treedepth



Is oriented twin-width the "right parameter" for cracking the approximability of twin-width?

Thank you for your attention!
Questions?