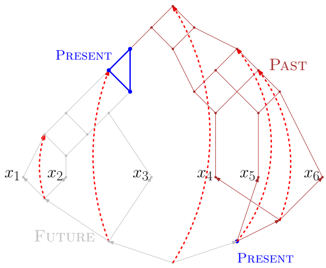
Treewidth and Scanwidth in Phylogenetics

Leo van Iersel[†] Niels Holtgrefe[†] Mark Jones[†] Mathias Weller[*]

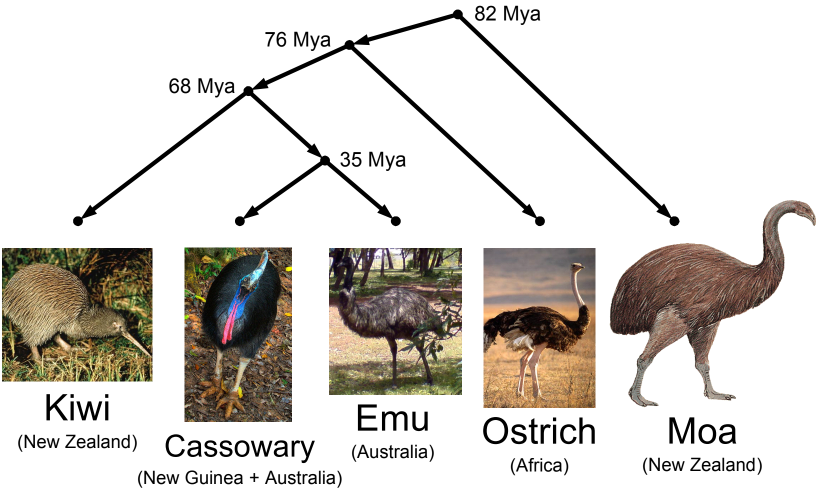
[†] TU Delft, [*] TU Berlin

Dutch Seminar on Optimization, 2023



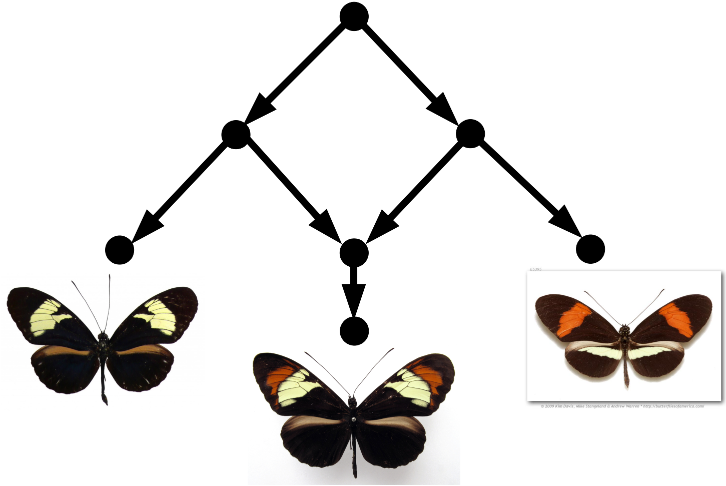
Definition

Let X be a finite set. A **rooted phylogenetic tree** on X is a rooted tree with no indegree-1 outdegree-1 vertices whose leaves are bijectively labelled by the elements of X.



Definition

Let X be a finite set. A **rooted phylogenetic network** on X is a rooted **directed acyclic graph** with no indegree-1 outdegree-1 vertices whose leaves are bijectively labelled by the elements of X.



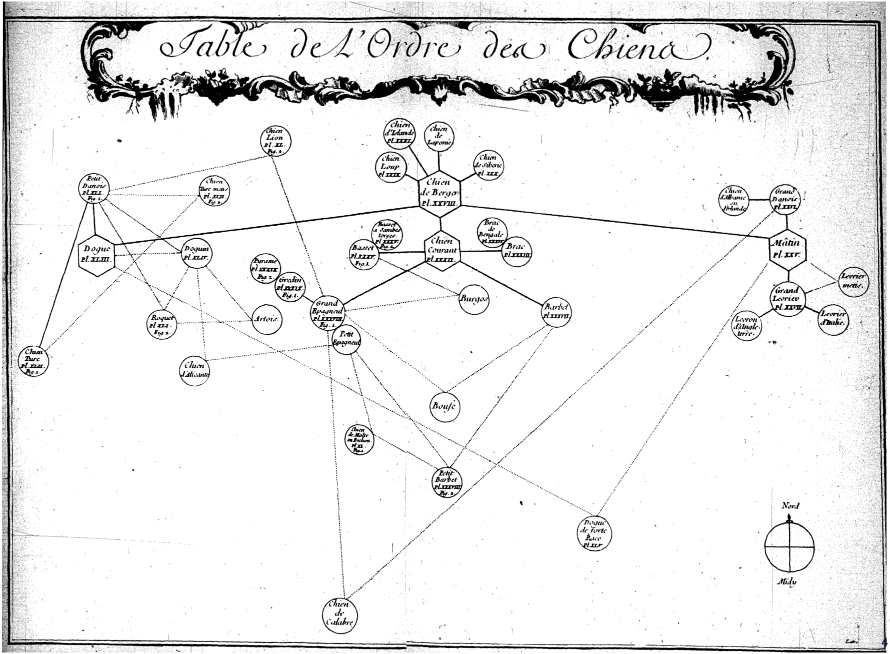
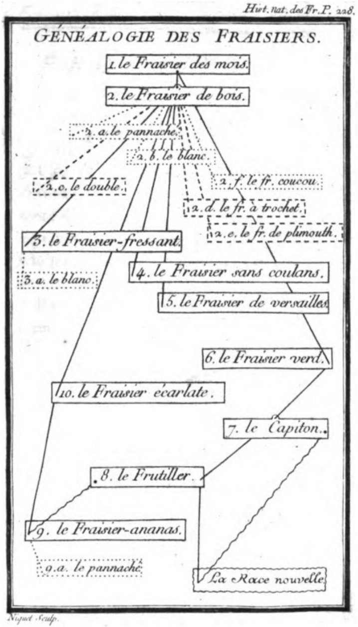
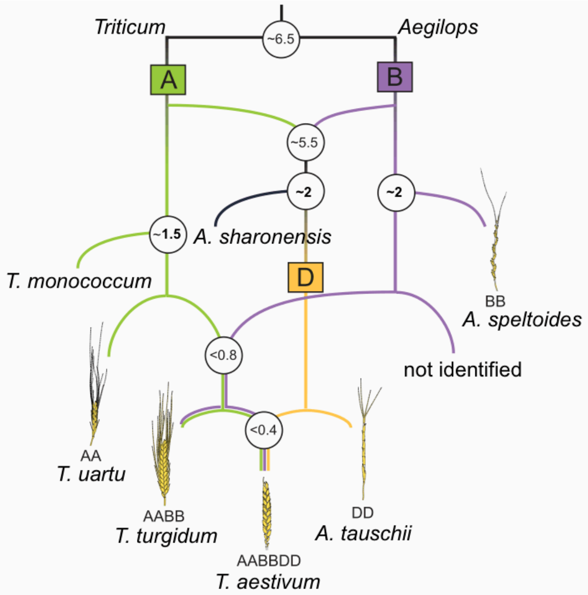


Table de L'Ordre des Chiens.
Chien de Berger pl. XXVIII.
Chien Courant pl. XXXII.
Mâtin pl. XXV.
Dogue pl. XLIII.
Doguin pl. XLIV.
Grand Epagneul
Petit Epagneul
Barbet
Burgos
Artois.
Roquet
Chien d'Alicante
Bouffe
Petit Barbet
Dogue de forte Race
Chien de Calabre
Grand Lévrier
Lévrier d'Italie.
Lévrier d'Angleterre.
Lévrier métis.
Chien Loup
Chien de Laponie
Chien de Sibérie
Chien d'Islande
Chien Lion
Chien Turc
Petit Danois
Grand Danois
Brac
Brac de Bengale
Basset
Nord
Midy

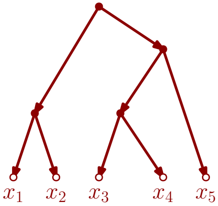
Strawberry phylogenetic network (Duchesne, 1766)



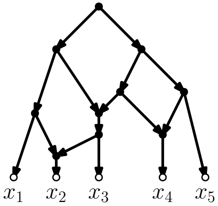
Wheat phylogenetic network (Marcussen et al., 2014)



Tree Containment problem



Tree T



Network N

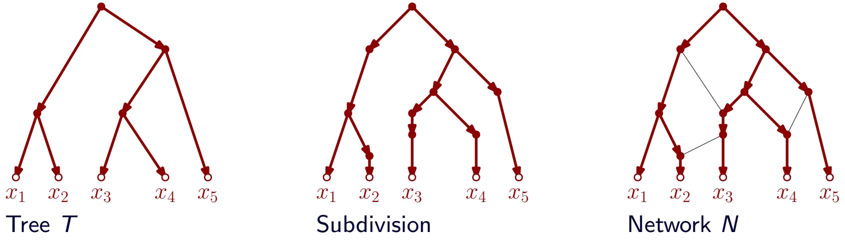
A phylogenetic network N on X **displays** a phylogenetic network T on X if a subdivision of T is a subgraph of N.

> TREE CONTAINMENT
> **Given:** rooted binary phylogenetic tree T on X, rooted binary phylogenetic network N on X
> **Question:** Does N display T?

- Stepping stone to other problems (eg network construction)
- Important verification step - check that a constructed network fits known data.
- NP-hard; FPT w.r.t. reticulation number of N, level of N

We study TREE CONTAINMENT parameterized by the **treewidth** of N

Tree Containment problem



A phylogenetic network N on X **displays** a phylogenetic network T on X if a subdivision of T is a subgraph of N.

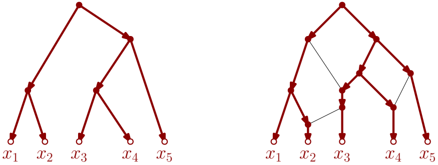
> TREE CONTAINMENT
> **Given:** rooted binary phylogenetic tree T on X, rooted binary phylogenetic network N on X
> **Question:** Does N display T?

- Stepping stone to other problems (eg network construction)
- Important verification step - check that a constructed network fits known data.
- NP-hard; FPT w.r.t. reticulation number of N, level of N

We study TREE CONTAINMENT parameterized by the **treewidth** of N

Our result, and similar problems

TREE CONTAINMENT
Given: rooted binary phylogenetic tree T on X, rooted binary phylogenetic network N on X
Question: Does N display T?



Theorem (van Iersel, Jones, Weller, 2022)

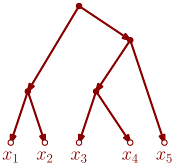
TREE CONTAINMENT *is fixed-parameter tractable (FPT) with respect to the treewidth k of the network. The algorithm has running time $2^{O(k^2)}|A|$.*

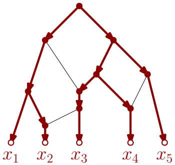
TREE CONTAINMENT has some similarities with:

- SUGRAPH ISOMORPHISM - but subdivisions allowed (homeomorphism)

Our result, and similar problems

TREE CONTAINMENT
Given: rooted binary phylogenetic tree T on X, rooted binary phylogenetic network N on X
Question: Does N display T?





Theorem (van Iersel, Jones, Weller, 2022)

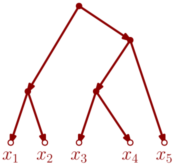
TREE CONTAINMENT *is fixed-parameter tractable (FPT) with respect to the treewidth k of the network. The algorithm has running time $2^{O(k^2)}|A|$.*

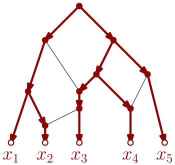
TREE CONTAINMENT has some similarities with:

- SUGRAPH ISOMORPHISM - but subdivisions allowed (homeomorphism)
- H-MINOR CONTAINMENT - but the "H" is large (and labelled)

Our result, and similar problems

TREE CONTAINMENT
Given: rooted binary phylogenetic tree T on X, rooted binary phylogenetic network N on X
Question: Does N display T?





Theorem (van Iersel, Jones, Weller, 2022)

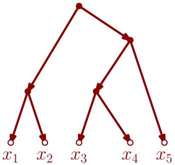
TREE CONTAINMENT *is fixed-parameter tractable (FPT) with respect to the treewidth k of the network. The algorithm has running time* $2^{O(k^2)}|A|$.

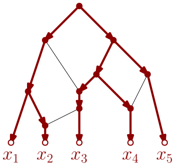
TREE CONTAINMENT has some similarities with:

- SUGRAPH ISOMORPHISM - but subdivisions allowed (homeomorphism)
- H-MINOR CONTAINMENT - but the "H" is large (and labelled)
- STEINER TREE - but we require specific topology

Our result, and similar problems

TREE CONTAINMENT
Given: rooted binary phylogenetic tree T on X, rooted binary phylogenetic network N on X
Question: Does N display T?





Theorem (van Iersel, Jones, Weller, 2022)

TREE CONTAINMENT *is fixed-parameter tractable (FPT) with respect to the treewidth k of the network. The algorithm has running time* $2^{O(k^2)}|A|$.

TREE CONTAINMENT has some similarities with:

- SUGRAPH ISOMORPHISM - but subdivisions allowed (homeomorphism)
- H-MINOR CONTAINMENT - but the "H" is large (and labelled)
- STEINER TREE - but we require specific topology

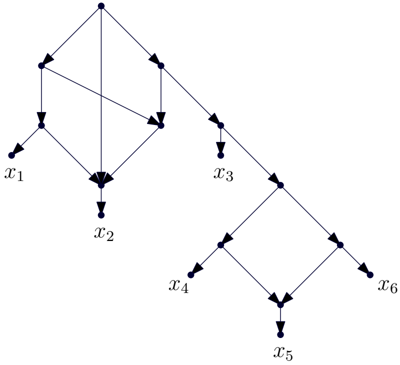
Big challenge: tracking interaction between two input graphs

Reticulation Number and Level

Definition

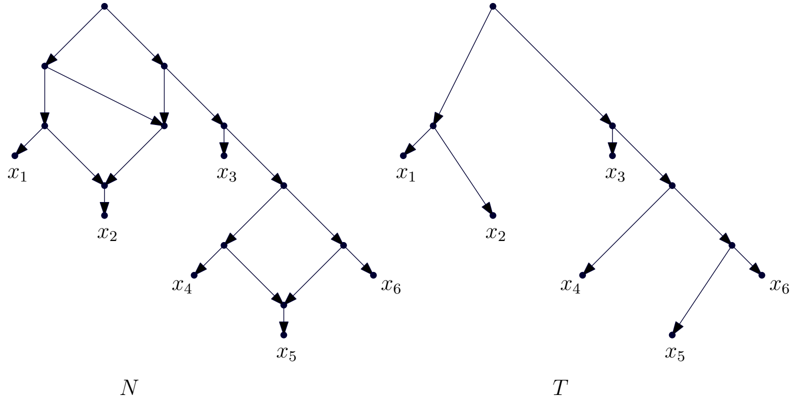
- the **reticulation number** r is defined as $r = |A| - |V| + 1$

  i.e. the number of arcs you need to delete to get a tree
- the **level** ℓ is the maximum reticulation number of a biconnected component

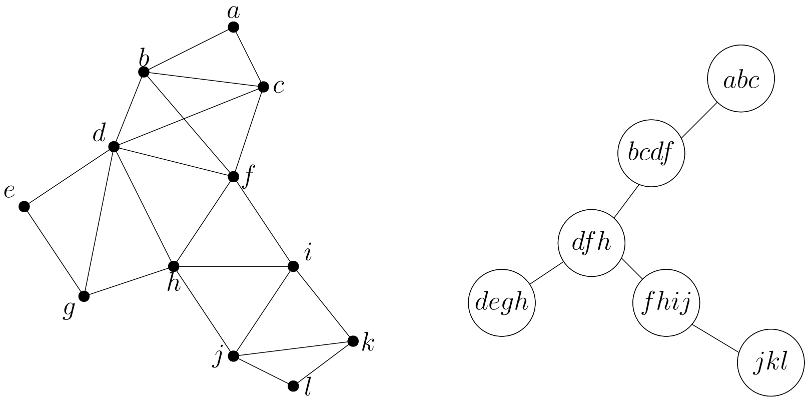


Reticulation Number and Level

- Tree Containment can easily be solved in $O(2^{\ell}|A|)$ time
- this can be improved to $O(2^{\ell/2}|V|^2)$ (Kanj, Nakhleh, Than, Xia, 2008)



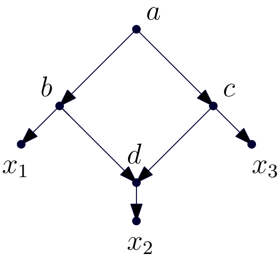
- The **treewidth** $tw(G)$ of G is the smallest **width** of a **tree decomposition** of G.



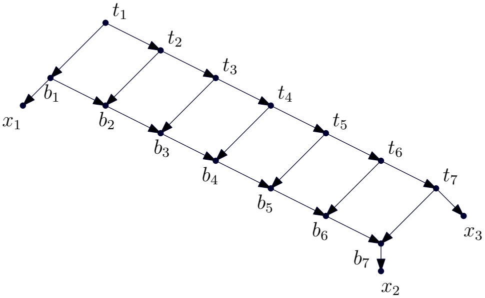
- Tree decomposition: a (non-phylogenetic) tree $\mathcal{T}$ whose vertices ('bags') are subsets of $V(G)$, and such that:
  1. Every vertex of G appears in at least one bag.
  2. For every edge uv in G, u, v appear in at least one bag together.
  3. For every vertex v in G, the bags containing v form a **connected subgraph** of $\mathcal{T}$.
- The **width** of a tree decomposition is the maximum size of a bag -1.

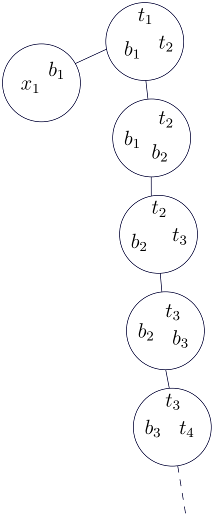
Treewidth vs Level

- a tree has treewidth 1 and level 0
- treewidth $\leq$ level $+1$



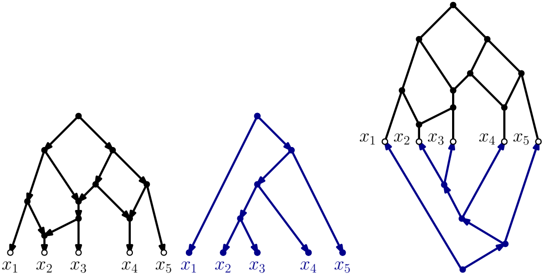
Treewidth vs Level





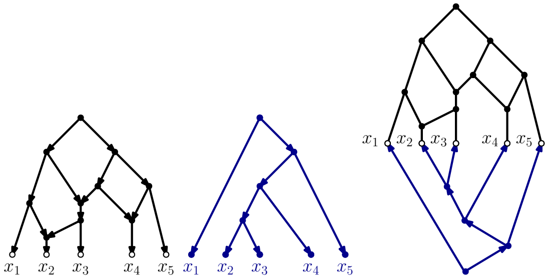
Display Graph

The **display graph** $D(N, T)$ is the graph derived from N, T by identifying leaves with the same taxon label.



Display Graph

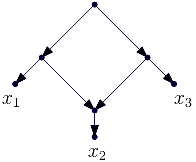
The **display graph** $D(N, T)$ is the graph derived from N, T by identifying leaves with the same taxon label.



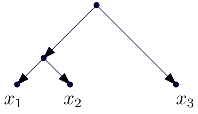
Theorem (Janssen, Jones, Kelk, Stamoulis & Wu, 2019)

If N displays T, then the display graph $D(N, T)$ has treewidth at most $2tw(N) + 1$.

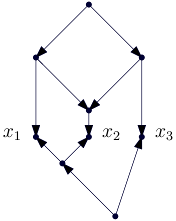
Treewidth of the display graph



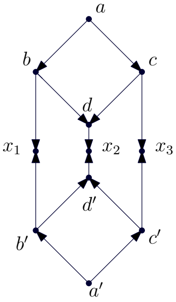
N



T



$D(N, T)$



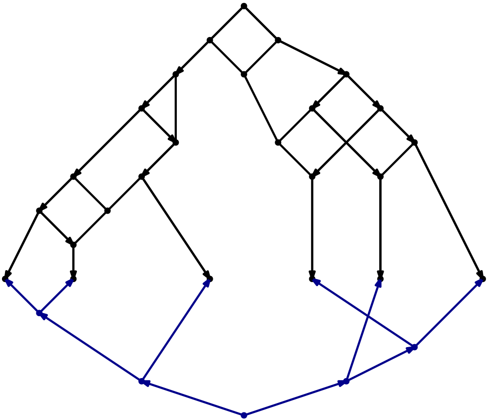
$D(N, N)$

Theorem (Janssen, Jones, Kelk, Stamoulis & Wu, 2019)

If N displays T, then the display graph $D(N, T)$ has treewidth at most $2tw(N) + 1$.

Embedding function

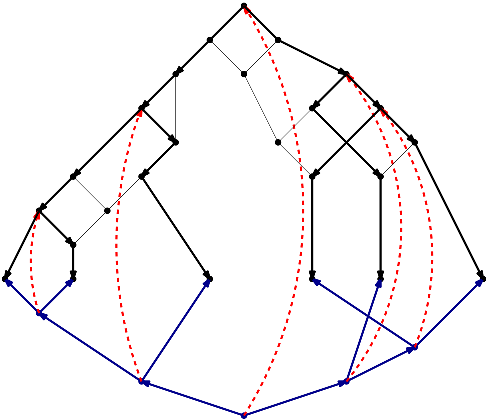
Represent a solution to Tree Containment by an **embedding function** on the display graph.



- Map every **vertex** u in T to a **vertex** $\phi(u)$ in N.
- (Each leaf is mapped to itself)
- Map each **arc** uv in T to a **path** $\phi(uv)$ in N from $\phi(u)$ to $\phi(v)$
- Paths are arc-disjoint, other constraints for technical reasons...

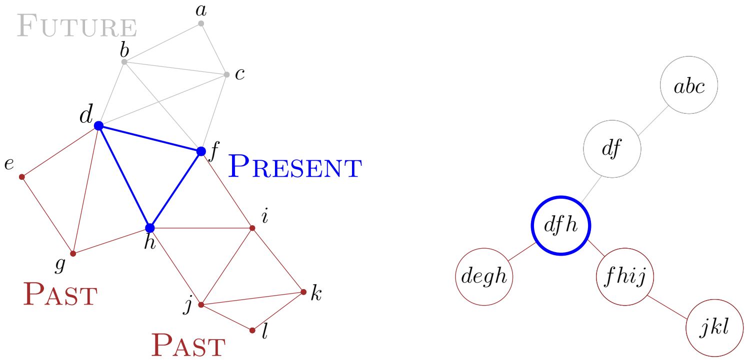
Embedding function

Represent a solution to TREE CONTAINMENT by an **embedding function** on the display graph.



- Map every **vertex** u in T to a **vertex** $\phi(u)$ in N.
- (Each leaf is mapped to itself)
- Map each **arc** uv in T to a **path** $\phi(uv)$ in N from $\phi(u)$ to $\phi(v)$
- Paths are arc-disjoint, other constraints for technical reasons...

For a dynamic programming algorithm, we can think of a bag in the tree decomposition in terms of Past/Present/Future:

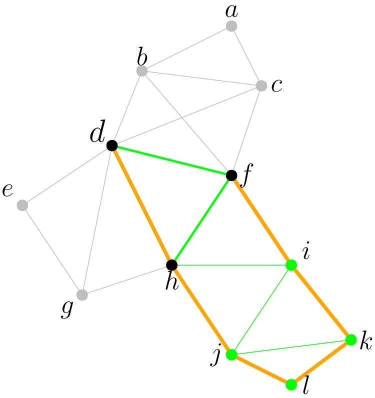


- **Past:** the part of the graph we've already explored
- **Present:** the current bag
- **Future:** the part of the graph we have yet to explore

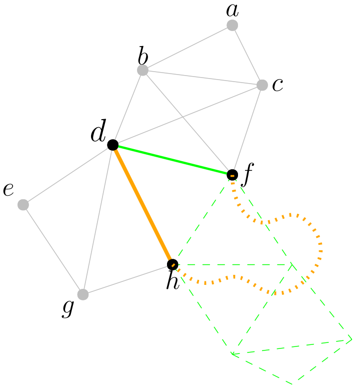
The Present separates the Past from the Future

General approach: Reduce information about a **partial solution** to a **small signature**
e.g. Hamiltonian Path:

Partial Solution

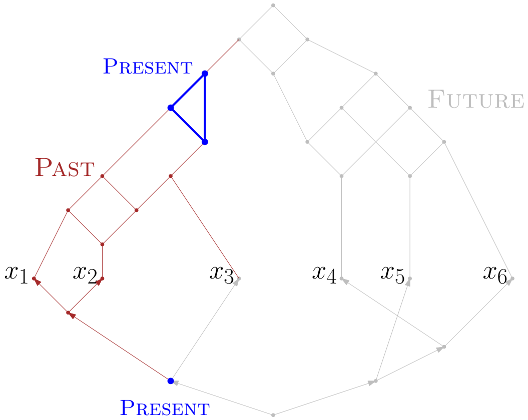


Signature



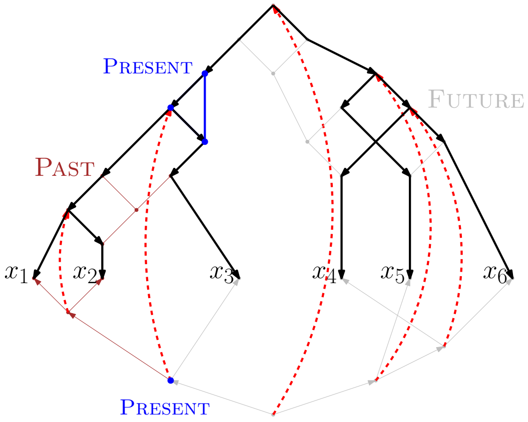
Tracking embedding within bag

How do we define signatures for TREE CONTAINMENT?



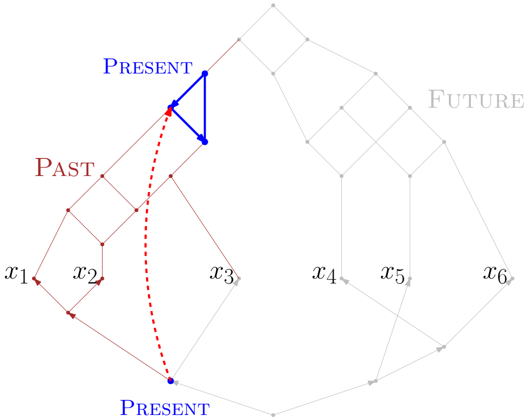
First idea: Track the embedding function restricted to Present.

How do we define signatures for TREE CONTAINMENT?



First idea: Track the embedding function restricted to Present.

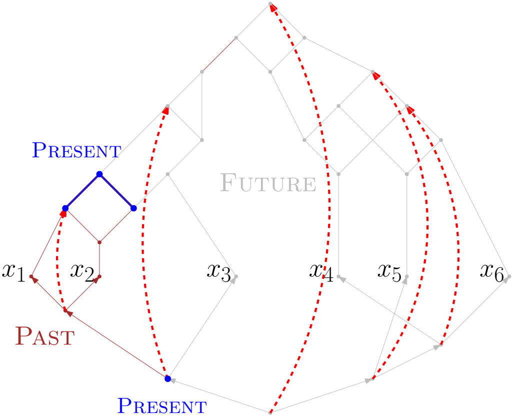
How do we define signatures for TREE CONTAINMENT?



First idea: Track the embedding function restricted to Present.

Tracking embedding within bag

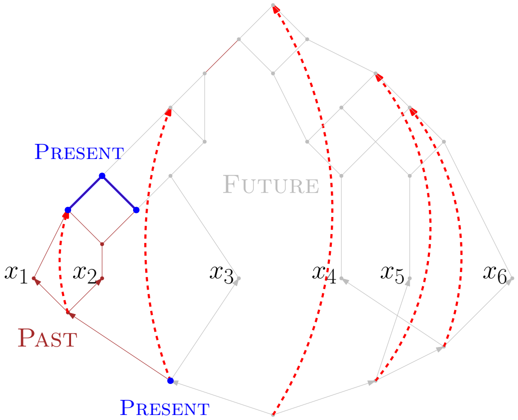
How do we define signatures for TREE CONTAINMENT?



First idea: Track the embedding function restricted to Present.

Tracking embedding within bag

How do we define signatures for TREE CONTAINMENT?

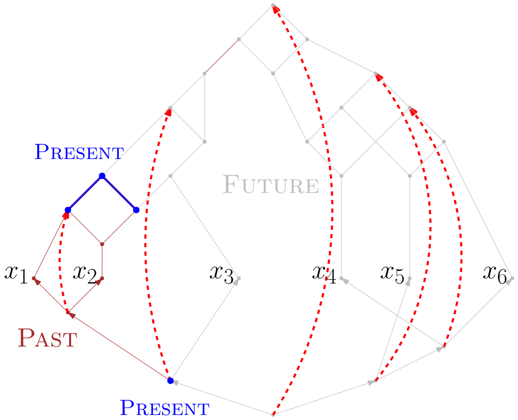


First idea: Track the embedding function restricted to Present.

Problem(s):

- The correct embedding may map **Present** vertices in T to **Past/Future** vertices in N

How do we define signatures for TREE CONTAINMENT?



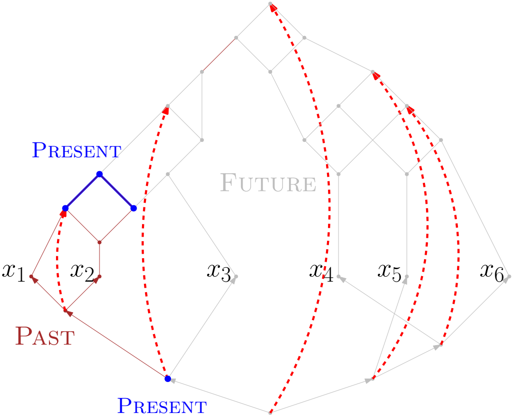
First idea: Track the embedding function restricted to Present.

Problem(s):

- The correct embedding may map **Present** vertices in T to **Past/Future** vertices in N
- **Past** tree vertices may have been mapped to **Present/Future** network vertices

Tracking embedding within bag

How do we define signatures for Tree Containment?



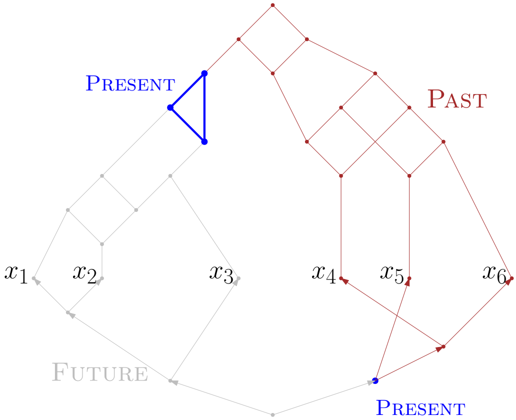
First idea: Track the embedding function restricted to Present.

Problem(s):

- The correct embedding may map **Present** vertices in T to **Past/Future** vertices in N
- **Past** tree vertices may have been mapped to **Present/Future** network vertices
- We may want to map **Future** vertices in T to **Past/Present** vertices in N!

Tracking embedding within bag

How do we define signatures for TREE CONTAINMENT?



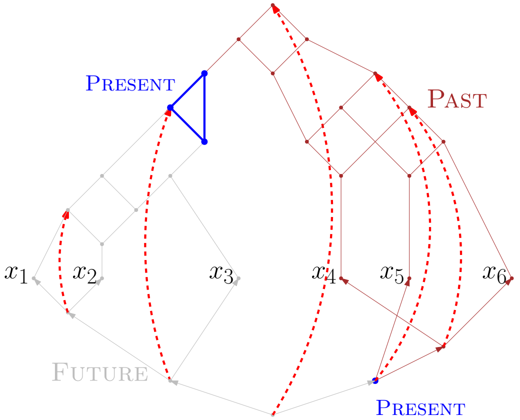
First idea: Track the embedding function restricted to Present.

Problem(s):

- The correct embedding may map **Present** vertices in T to **Past/Future** vertices in N
- **Past** tree vertices may have been mapped to **Present/Future** network vertices
- We may want to map **Future** vertices in T to **Past/Present** vertices in N!

Tracking embedding within bag

How do we define signatures for TREE CONTAINMENT?

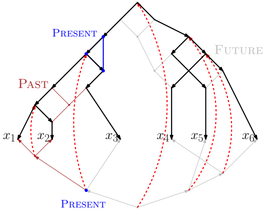


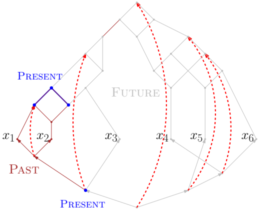
First idea: Track the embedding function restricted to Present.

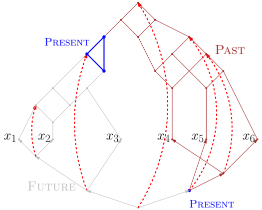
Problem(s):

- The correct embedding may map **Present** vertices in T to **Past/Future** vertices in N
- **Past** tree vertices may have been mapped to **Present/Future** network vertices
- We may want to map **Future** vertices in T to **Past/Present** vertices in N!

Which information do we keep?

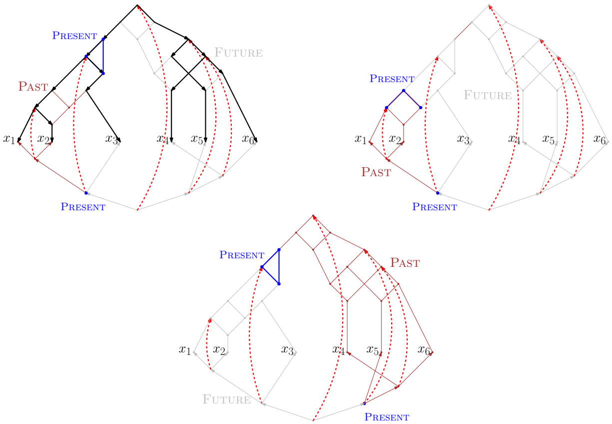






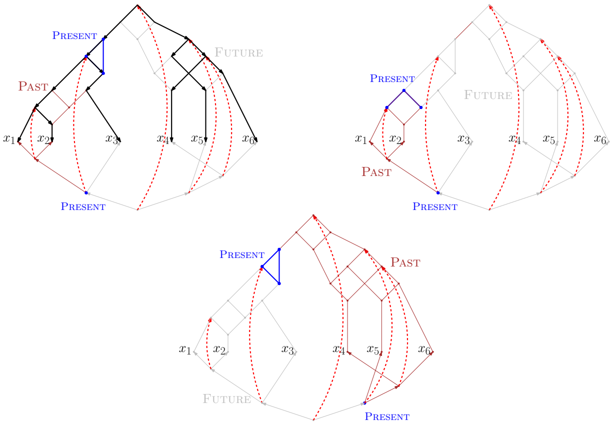
	$\phi(u) \in$ Past	$\phi(u) \in$ Present	$\phi(u) \in$ Future
$u \in$ Past			
$u \in$ Present			
$u \in$ Future			

Which information do we keep?



	$\phi(u) \in$ Past	$\phi(u) \in$ Present	$\phi(u) \in$ Future
$u \in$ Past			
$u \in$ Present		Keep	
$u \in$ Future			

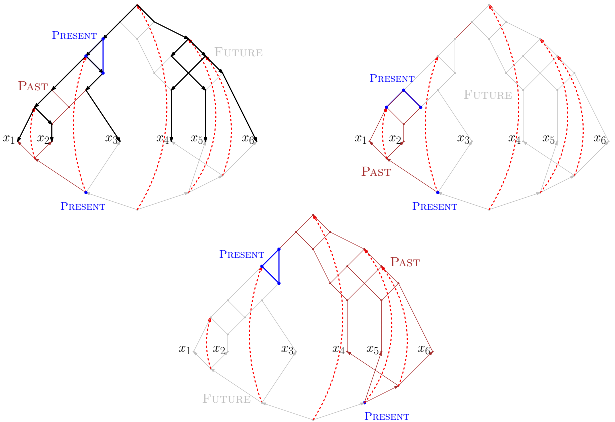
Which information do we keep?



	$\phi(u) \in$ Past	$\phi(u) \in$ Present	$\phi(u) \in$ Future
$u \in$ Past			
$u \in$ Present	Keep*	Keep	Keep*
$u \in$ Future			

- * do not store identities of vertices in Past/Future
  (e.g. if $\phi(u) = v$ and $v \in$ Past, we only record that $\phi(u) \in$Past)

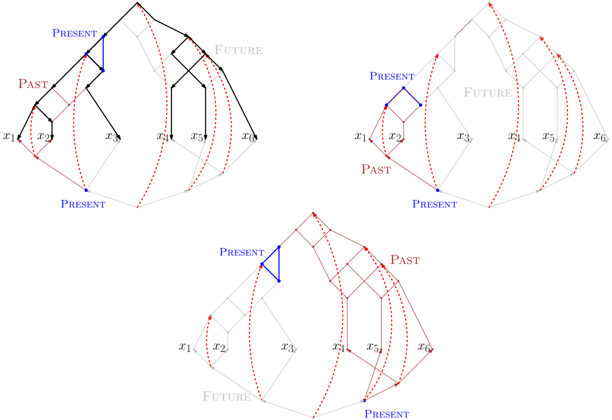
Which information do we keep?



	$\phi(u) \in$ Past	$\phi(u) \in$ Present	$\phi(u) \in$ Future
$u \in$ Past		Keep*	Keep*
$u \in$ Present	Keep*	Keep	Keep*
$u \in$ Future			

- * do not store identities of vertices in Past/Future
  (e.g. if $\phi(u) = v$ and $v \in$ Past, we only record that $\phi(u) \in$Past)

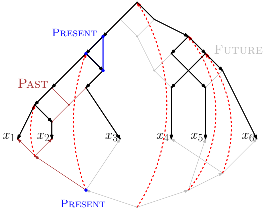
Which information do we keep?

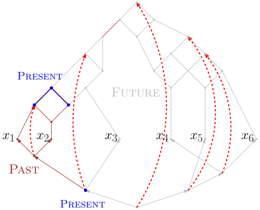


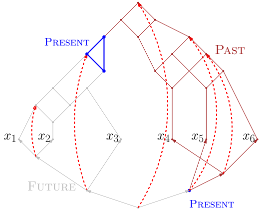
	$\phi(u) \in$ Past	$\phi(u) \in$ Present	$\phi(u) \in$ Future
$u \in$ Past		Keep*	Keep*
$u \in$ Present	Keep*	Keep	Keep*
$u \in$ Future	Keep*	Keep*	

- * do not store identities of vertices in Past/Future (e.g. if $\phi(u) = v$ and $v \in$ Past, we only record that $\phi(u) \in$Past)

Which information do we keep?



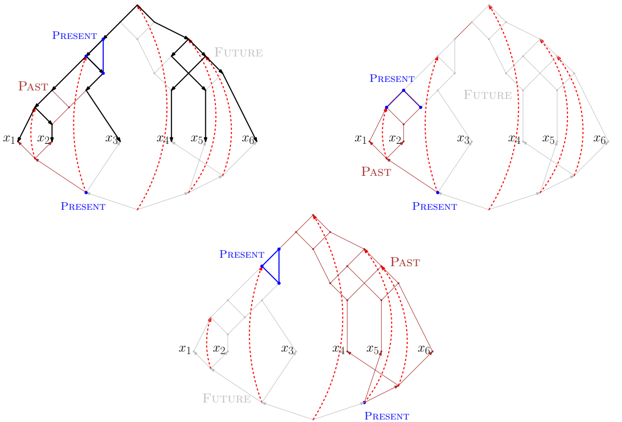




	$\phi(u) \in$ Past	$\phi(u) \in$ Present	$\phi(u) \in$ Future
$u \in$ Past	Forget	Keep*	Keep*
$u \in$ Present	Keep*	Keep	Keep*
$u \in$ Future	Keep*	Keep*	

- * do not store identities of vertices in Past/Future
(e.g. if $\phi(u) = v$ and $v \in$ Past, we only record that $\phi(u) \in$Past)

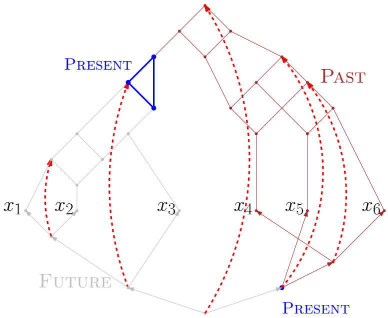
Which information do we keep?



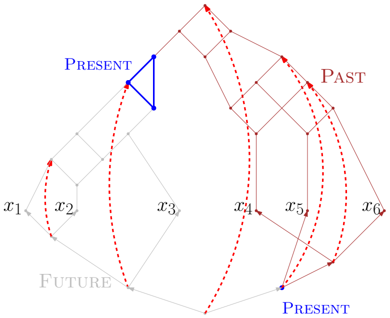
	$\phi(u) \in$ Past	$\phi(u) \in$ Present	$\phi(u) \in$ Future
$u \in$ Past	Forget	Keep*	Keep*
$u \in$ Present	Keep*	Keep	Keep*
$u \in$ Future	Keep*	Keep*	'Forget'

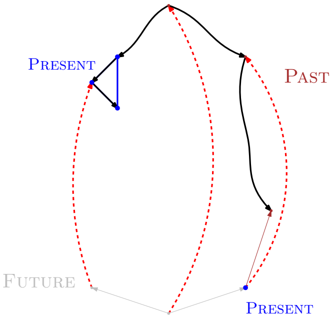
- * do not store identities of vertices in Past/Future (e.g. if $\phi(u) = v$ and $v \in$ Past, we only record that $\phi(u) \in$Past)

Example Signature

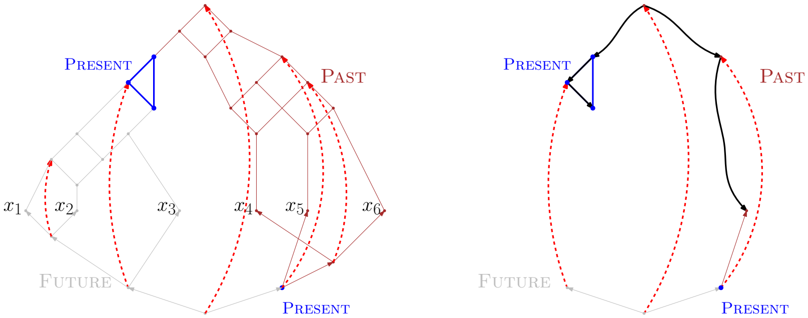


Example Signature





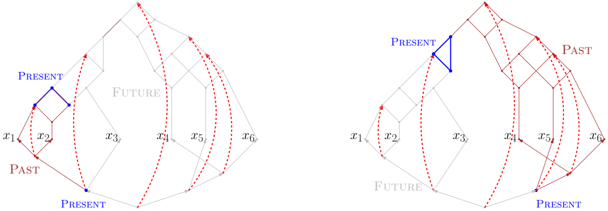
Example Signature



Ommited details:

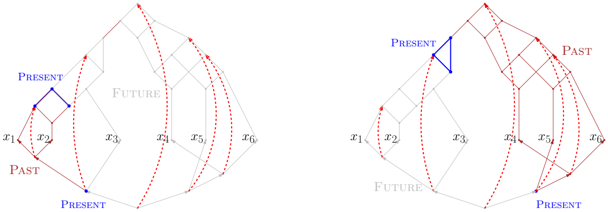
- Tree arcs / corresponding paths only removed if all their vertices are in Past (or Future)
- Vertices only removed if all their incident arcs are removed
- Long network paths within Past/Future are contracted

Bounding information in a signature



Recall Present = one bag of the tree decomposition, thus $|\text{Present}| \leq tw(D(N,T)) + 1$.

	$\phi(u) \in$ Past	$\phi(u) \in$ Present	$\phi(u) \in$ Future
$u \in$ Past	Forget	**Keep***	**Keep***
$u \in$ Present	**Keep***	**Keep**	**Keep***
$u \in$ Future	**Keep***	**Keep***	'Forget'

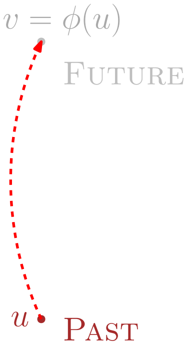


Recall Present $=$ one bag of the tree decomposition, thus $|\text{Present}| \leq tw(D(N,T)) + 1$.

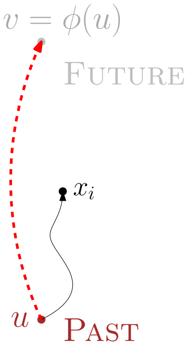
	$\phi(u) \in$ Past	$\phi(u) \in$ Present	$\phi(u) \in$ Future
$u \in$ Past		Keep*	**Keep***
$u \in$ Present	Keep*	Keep	Keep*
$u \in$ Future	**Keep***	Keep*	

- Relatively easy to bound information involving the Present
- Vertices that move between Past/Future are more tricky...

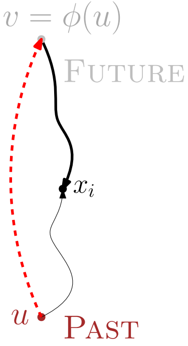
Bounding time-travellers



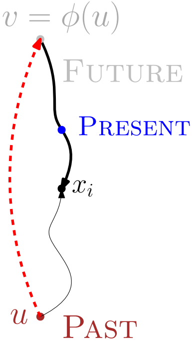
Bounding time-travellers

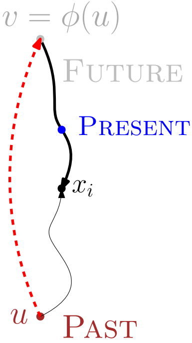


Bounding time-travellers



Bounding time-travellers





Number of *lowest* tree vertices u for which $u \in$ Past, $\phi(u) \in$ Future (or vice versa) can be bounded by the number of vertices in Present.

Summary

- Size of signatures (and number of signatures per bag) is bounded by a function of treewidth
- Deciding whether a given signature for a bag has a corresponding (partial) solution can be decided using only signatures on child bags.

Theorem (van Iersel, Jones, Weller, 2022)

Tree Containment *is fixed-parameter tractable (FPT) with respect to the treewidth k of the network. The algorithm has running time $2^{O(k^2)}|A|$.*

Hybridization Number

HYBRIDIZATION NUMBER
Given: (Un)rooted phylogenetic trees $T_1, \ldots, T_r$ on X, integers w, k.
Parameter: w
Question: Does there exist a phylogenetic network N with treewidth $\leq w$ and reticulation number $\leq k$ such that N displays each of $T_1, \ldots, T_r$?

- Open question: Is HYBRIDIZATION NUMBER FPT w.r.t treewidth (for constant r)?

Hybridization Number

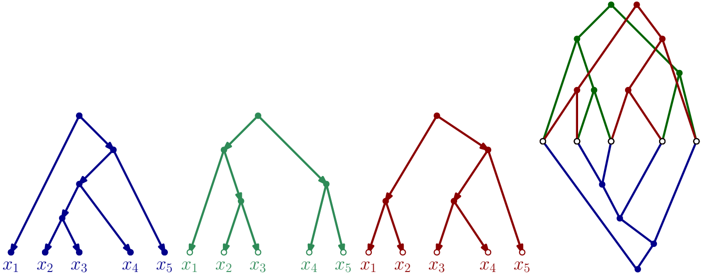
HYBRIDIZATION NUMBER
Given: (Un)rooted phylogenetic trees $T_1, \ldots, T_r$ on X, integers w, k.
Parameter: w
Question: Does there exist a phylogenetic network N with treewidth $\leq w$ and reticulation number $\leq k$ such that N displays each of $T_1, \ldots, T_r$?

- Open question: Is HYBRIDIZATION NUMBER FPT w.r.t treewidth (for constant r)?
- Key challenge: We don't know N to do dynamic programming!

Hybridization Number

Hybridization Number
Given: (Un)rooted phylogenetic trees $T_1, \ldots, T_r$ on X, integers w,k.
Parameter: w
Question: Does there exist a phylogenetic network N with treewidth $\leq w$ and reticulation number $\leq k$ such that N displays each of $T_1, \ldots, T_r$?

- Open question: Is Hybridization Number FPT w.r.t treewidth (for constant r)?
- Key challenge: We don't know N to do dynamic programming!
- But: If N displays $T_1, \ldots, T_r$ then display graph $D(T_1, \ldots, T_r)$ has treewidth $< r \cdot (tw(N) + 1) \leq r(w + 1)$
- Hope for a DP algorithm?

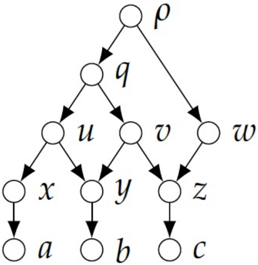


Scanwidth

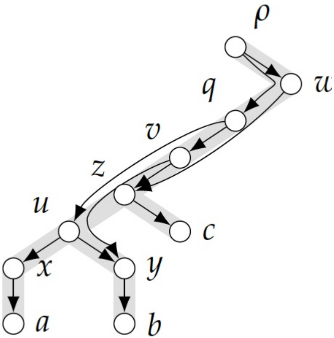
Definition

A **tree extension** of a network N is a tree T with the same vertex set as N such that

- $\exists$ u-v path in N $\implies$ $\exists$ u-v path in T



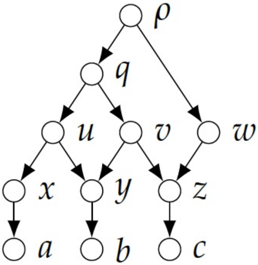
network N



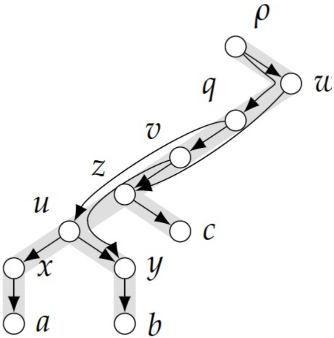
a tree-extension T of N

Definition

- The **width** of a tree extension is the maximum number of network edges travelling through an edge of the tree extension.
- The **scanwidth** of a network is the minimum width of a tree extension.



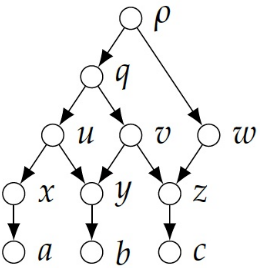
network N



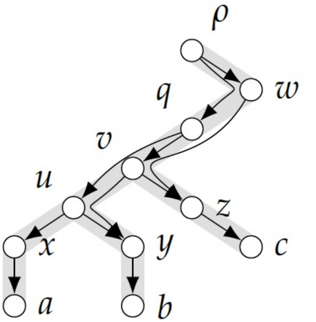
a tree-extension T of N

Definition

- The **width** of a tree extension is the maximum number of network edges travelling through an edge of the tree extension.
- The **scanwidth** of a network is the minimum width of a tree extension.

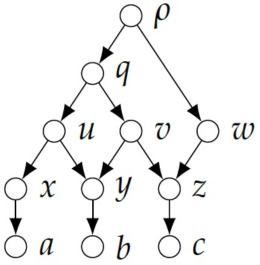


network N

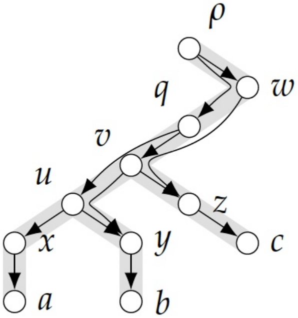


an optimal tree-extension of N

The idea of scanwidth is that you "scan" a network with multiple scanners.

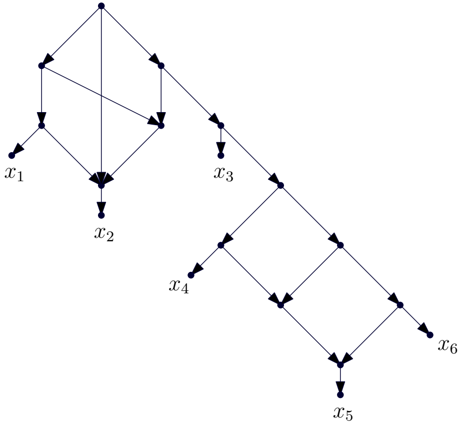


network N



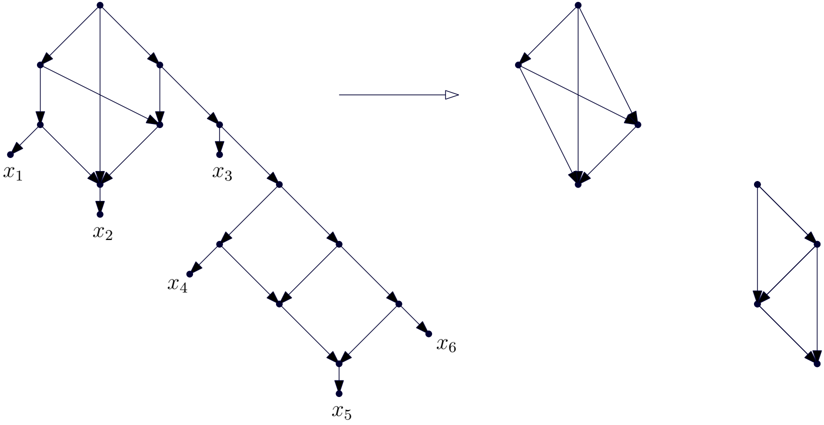
an optimal tree-extension of N

The scanwidth of a network is the maximum scanwidth of a biconnected component.

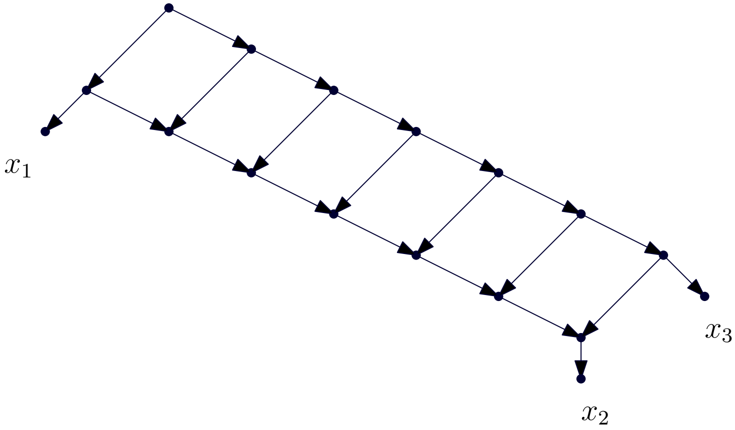


Decomposition and reduction

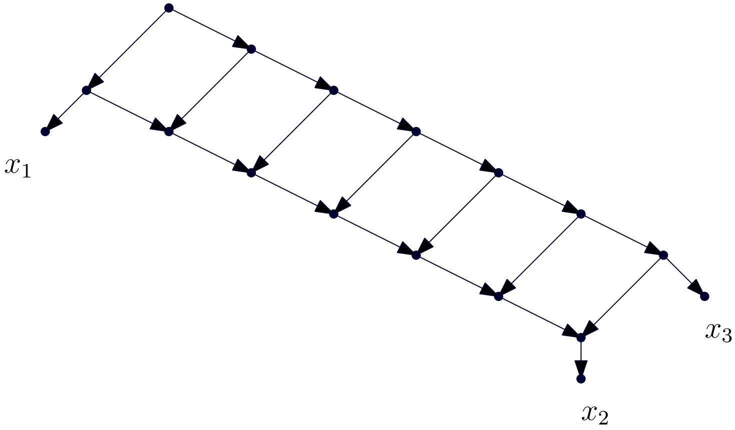
- split into biconnected components (delete trivial ones)
- suppress indegree-1 outdegree-1 vertices



Scanwidth vs Level



Scanwidth vs Level



Lemma

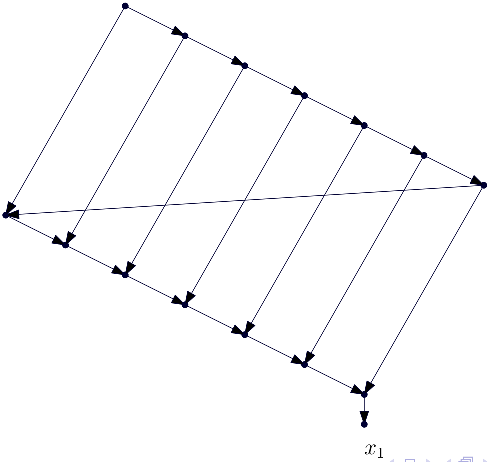
If W is a weakly connected sink set then $\delta^-(W) \leq r + 1$.

Hence, scanwidth $\leq r + 1$ with r the reticulation number.
Hence, scanwidth $\leq \ell + 1$ with ℓ the level.

Scanwidth vs Treewidth

Lemma

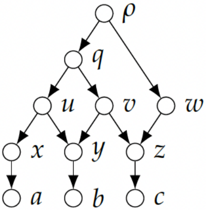
treewidth $\leq$ *scanwidth*



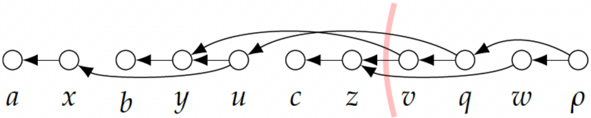
Scanwidth vs Cutwidth

Definition

- an **extension** of a network is a linear ordering of its vertices such that all arcs point to the left
- the **width** of an extension is the maximum number of arcs cut by any 'vertical cut'
- the **cutwidth** of a network is the minimum width of an extension



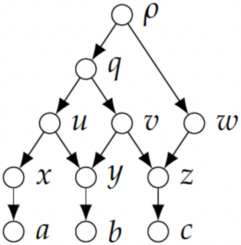
N



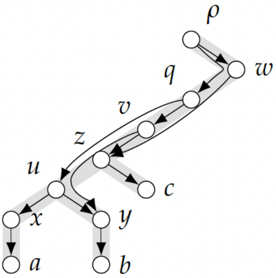
an extension of N

Scanwidth using extensions

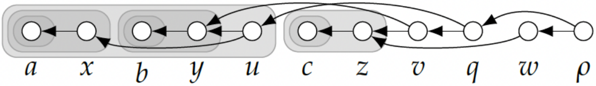
Scanwidth can also be defined using extension, but then you split each cut corresponding to weakly connected components



N



tree extension of N



extension of N

Lemma

scanwidth $\leq$ *cutwidth*

Scanwidth

- NP-hard to compute (Berry, Scornavacca and Weller, 2020)

Scanwidth

- NP-hard to compute (Berry, Scornavacca and Weller, 2020)
- Exact dynamic programming algorithm $O(k \cdot |V|^{k+2})$ (Holtgrefe, 2023)

Scanwidth

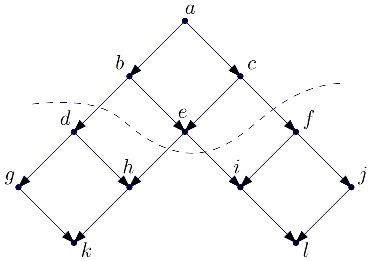
- NP-hard to compute (Berry, Scornavacca and Weller, 2020)
- Exact dynamic programming algorithm $O(k \cdot |V|^{k+2})$ (Holtgrefe, 2023)
- solves instances with up to 100 leaves and 30 indegree-2 vertices exactly

Scanwidth

- NP-hard to compute (Berry, Scornavacca and Weller, 2020)
- Exact dynamic programming algorithm $O(k \cdot |V|^{k+2})$ (Holtgrefe, 2023)
- solves instances with up to 100 leaves and 30 indegree-2 vertices exactly
- **Open question:** can scanwidth be computed in FPT time?
  (i.e. $f(k) \cdot |V|^c$ time with c a constant)

Dynamic programming algorithm idea

- for bipartition (L, R) of the vertices, compute the scanwidth assuming vertices from R are to the right from vertices in L in the extension
- split into weakly connected components when possible



Lemma

There are at most $|V|^k$ sets L for which $N[L]$

- *is weakly connected*
- *has no outgoing arcs*
- *has at most k incoming arcs*

Running time $O(k \cdot |V|^{k+2})$

Recent results (van Iersel, Jones, Weller)

Tree Containment can be solved using scanwidth:

Recent results (van Iersel, Jones, Weller)

Tree Containment can be solved using scanwidth:

- much simpler algorithm than using treewidth

Recent results (van Iersel, Jones, Weller)

TREE CONTAINMENT can be solved using scanwidth:

- much simpler algorithm than using treewidth
- dependency on parameter is much better: $2^{O(k \log k)}|A|$

Recent results (van Iersel, Jones, Weller)

TREE CONTAINMENT can be solved using scanwidth:

- much simpler algorithm than using treewidth
- dependency on parameter is much better: $2^{O(k \log k)}|A|$

  assuming a tree extension is given

Recent results (van Iersel, Jones, Weller)

TREE CONTAINMENT can be solved using scanwidth:

- much simpler algorithm than using treewidth
- dependency on parameter is much better: $2^{O(k \log k)}|A|$

  assuming a tree extension is given
- can be generalized to nonbinary

Recent results (van Iersel, Jones, Weller)

Tree Containment can be solved using scanwidth:

- much simpler algorithm than using treewidth
- dependency on parameter is much better: $2^{O(k \log k)}|A|$

  assuming a tree extension is given
- can be generalized to nonbinary
- faster is not possible under the ETH

Recent results (van Iersel, Jones, Weller)

TREE CONTAINMENT can be solved using scanwidth:

- much simpler algorithm than using treewidth
- dependency on parameter is much better: $2^{O(k \log k)}|A|$

  assuming a tree extension is given
- can be generalized to nonbinary
- faster is not possible under the ETH
- the generalization NETWORK CONTAINMENT is W[1]-hard

Recent results (van Iersel, Jones, Weller)

Tree Containment can be solved using scanwidth:

- much simpler algorithm than using treewidth
- dependency on parameter is much better: $2^{O(k \log k)}|A|$

  assuming a tree extension is given
- can be generalized to nonbinary
- faster is not possible under the ETH
- the generalization Network Containment is W[1]-hard

- **Open question:** can scanwidth be computed in FPT time?
  (i.e. $f(k) \cdot |V|^c$ time with c a constant)

Recent results (van Iersel, Jones, Weller)

Tree Containment can be solved using scanwidth:

- much simpler algorithm than using treewidth
- dependency on parameter is much better: $2^{O(k \log k)}|A|$

  assuming a tree extension is given
- can be generalized to nonbinary
- faster is not possible under the ETH
- the generalization Network Containment is W[1]-hard

- **Open question:** can scanwidth be computed in FPT time?
  (i.e. $f(k) \cdot |V|^c$ time with c a constant)
- **Open question:** is Hybridization Number FPT parameterized by scanwidth?