Finding **bibliographic references:**
- Google Scholar: large coverage, including noise (preprints, fake papers...)
- Bibliographic resources with university access:
  ressources-en-ligne-de-a-a-z/
  → Science Direct for Elsevier journals, Springer, JSTOR, etc.
- University library

**Conferences** with computer science papers applied to phylogenetics:
- International: ISMB, RECOMB, WABI, ECCB, ISBRA, etc.
  + algorithmics conferences: SODA, CPM, ISAAC, COCOON, etc.
- In France: JOBIM, Alphy

**Scholarly organizations:**
Phylogenetic trees

Phylogenetic tree of a set of species:

- organizing them according to common characters
- describing their evolution

Lamarck: Histoire naturelle des animaux sans vertèbres (1815)
Phylogenetic tree of a set of species:

- organizing them according to common characters
- describing their evolution

“Newick” format:

(B:6.0,(A:5.0,C:3.0,E:4.0)Ancestor1:5.0,D:11.0);

Not unique:

((A:5.0,C:3.0,E:4.0)Ancestor1:5.0,B:6.0,D:11.0);

Possible to have nodes which are not binary:

• uncertainty
• or speciation known to be at the same time
Properties of phylogenetic trees

Characterizing a tree with:

- its “clusters”: one cluster of $T = \text{the set of leaves below one vertex of } T$
- its “triplets”: one triplet of $T = \text{a tree on 3 leaves contained in } T$
- distances between the leaves
Characterizing a **rooted tree** with:

- its “clusters”: one cluster of $T = \text{the set of leaves below one vertex of } T$
- its “triplets”: one triplet of $T = \text{a tree on 3 leaves contained in } T$
- **distances between the leaves**

Characterizing an **unrooted tree** with:

- its “splits”: one split of $T = \text{bipartition of the leaves induced by one edge of } T$
- its “quartets”: one quartet of $T = \text{an unrooted tree on 4 leaves contained in } T$
- **distances between the leaves**
Properties of rooted and unrooted trees

Clusters: “laminar family”, i.e. it contains no overlapping sets
→ reconstruction from clusters: Hasse Diagram of cluster inclusion

Triplets (binary trees): do not contain \{ab\mid c, b\mid cd, a\mid bd\} or \{ab\mid c, b\mid cd, ad\mid b\}

Guillemot & Mnich, Kernel and fast algorithm for Dense Triplet Inconsistency, 2013

Splits: “compatible split system”, i.e. for any pair of splits \(A1\mid B1, A2\mid B2\), at least one of the sets \(A1\cap A2, A1\cap B2, B1\cap A2, B1\cap B2\) is empty

Quartets (binary trees): for any leaf \(e\), \(ab\mid cd \in Q \Rightarrow ab\mid ce \in Q\) or \(ae\mid cd \in Q\)

Bandelt & Dress, Reconstructing the shape of a tree from observed dissimilarity data, 1986
Properties of rooted and unrooted trees

Tree distances:
Characterized by Buneman's **four-point condition:**

\[
\text{for all } a, b, c, d, \quad d(a,b) + d(c,d) \leq \max\{d(a,c) + d(b,d), \ d(a,d) + d(b,c)\}
\]

⇔ for any four points, we can relabel them a, b, c, d such that
\[d(a,b) + d(c,d) \leq d(a,c) + d(b,d) = d(a,d) + d(b,c).\]

Given a tree distance, only one possible tree.

Buneman, A Note on the Metric Properties of Trees, 1974

**Tree distances** when the tree contains a center at equal distance from all leaves:
Characterized by the **ultrametric inequality:**

\[
\text{For all } a, b, c, \quad d(a,b) \leq \max\{d(a,c), \ d(b,c)\}
\]

*molecular clock hypothesis!*
Reconstructing a tree from an ultrametric

**UPGMA algorithm** (Unweighted Pair Group Method with Arithmetic Mean):

- Initialize all clusters with leaf singletons
- While there are more than 2 clusters:
  - pick the nearest two clusters
  - combine them and update the distance matrix with average values (average weighted by the size of each of the two clusters)

→ Correctly reconstructs ultrametric distances, but not all tree distances

→ Neighbor-Joining...

Sokal & Michener, A statistical method for evaluating systematic relationships, 1958
Discovering the **Neighbor-Joining algorithm**: 

The Neighbor-Joining algorithm (NJ) is a tree reconstruction algorithm which identifies at each step the **two neighbor leaves** which **minimizes the total expected length of the tree**, and replaces them by their parent.

**Q1.** Given a tree $T_{ij}$ made of a central vertex $u$ with $n-2$ leaf neighbors, as well as a neighbor $v$ of $u$ having two leaf neighbors $i$ and $j$, and an additive metric $d$ corresponding to $T_{ij}$, evaluate the total length $L_{ij}$ of $T_{ij}$ depending on:

- the sum $S_1$ of all distances between leaf neighbors of $u$ on the one side and $i$ and $j$ on the other side;
- the sum $S_2$ of all distances between leaf neighbors of $u$;
- the distance $d(i,j)$ between $i$ and $j$. 

![Diagram of Neighbor-Joining algorithm](image)
Discovering the **Neighbor-Joining algorithm**: 

Q2. Rewrite $L_{ij}$ to express this total length depending on the sum of distances between all pairs of leaves of $T_{ij}$, as well as $d(i,j)$, $r_i$ and $r_j$, where $r_x$ is the sum of distances between leaf $x$ and all other leaves of $T_{ij}$.

Q3. The NJ algorithm consists in repeating, starting from a star tree: choose two vertices $i$ and $j$ which minimize $L_{ij}$ and replace them by node $v$ in the distance matrix corresponding to $d$. Give an appropriate formula to compute $d(v,k)$ for each leaf $k$ of $T_{ij}$ depending on the distances between leaves of $T_{ij}$ (including $i$ and $j$: for them, use $d(i,j)$, $r_i$ and $r_j$).
Reconstructing a tree from its triplets

**BUILD algorithm:**

- Build the following graph: leaves as vertices; for each triplet \( a|bc \), add edge \( bc \).
- While there is more than one connected component:
  - each connected component corresponds to one subtree
  - recursively apply the algorithm on the leaf set of each connected component

\[ \{a|bc, a|de, a|df, b|dg, b|ef, c|df, d|ac, d|fg, e|ab, f|de, g|ab, g|ac\} \]

→ When missing triplets, efficient implementation in \( O(|T| + n^2 \log n) \)

---

Aho, Sagiv, Szymanski & Ullman, Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions, 1981.

Henzinger, King & Warnow, Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology, 1999.
Reconstructing a tree from its triplets

BUILD algorithm:

• Build the following graph: leaves as vertices; for each triplet \( a|bc \), add edge bc.

• While there is more than one connected component:
  - each connected component corresponds to one subtree
  - recursively apply the algorithm on the leaf set of each connected component

Example:

\{a|bc, a|de, a|df, b|dg, b|ef, c|df, d|ac, d|fg, e|ab, f|de, g|ab, g|ac\}

→ When missing triplets, efficient implementation in \( O(|T| + n^2 \log n) \)

Henzinger, King & Warnow, Constructing a tree from homeomorphic subtrees, with applications to computational evolutionary biology, 1999.
Visualizing phylogenetic trees

Visualize branch lengths or not

Several kinds of visualizations:

- rectangular phylogram
- rectangular cladogram
- slanted cladogram
- circular phyogram
- circular cladogram
- inner circular cladogram
- radial phylogram
- radial cladogram

“Equal angle” algorithm to draw a radial phylogram on $n$ leaves:

- Compute the angles “bottom-up” starting with angle $2i\pi/n$ for leaf $i$
- Locate the nodes “top-down” using:
  - the angles
  - the edge lengths
- Add the labels (avoiding overlap)

“Equal daylight” algorithm to optimize used space:
Comparing trees

- **Maximum Agreement Subtree (MAST):** Finden & Gordon, Obtaining common pruned trees, 1985
  - Given $T_1$ and $T_2$ on the set $X$ of leaves, an *agreement subtree* $T$ of $T_1$ and $T_2$, on a subset $X'$ of leaves, is such that $T_1$ and $T_2$ restricted to $X'$ are equal to $T$.
  - a *maximum agreement subtree* is an agreement subtree of maximum size

→ Polynomial-time algorithm for 2 trees (rooted or not), with fixed degree (or not)
  Steel & Warnow, Kaikoura tree theorems: Computing the maximum agreement subtree, 1993

→ $O(n \log n)$ algorithm for 2 rooted binary trees
  Cole & Hariharan, An $O(n \log n)$ algorithm for the maximum agreement subtree problem for binary trees, 1996

→ NP-hard for 3 (or more) trees of unbounded degree
  Amir & Keselman, Maximum agreement subtree in a set of evolutionary trees, 1997

→ Polynomial-time algorithm if one of the trees has bounded maximum degree
  Farach, Przytycka & Thorup, On the agreement of many trees, 1995

Dynamic programming for 2 rooted binary trees:

$$n_{\text{MAST}}(T_1,T_2) = \max$$

![Diagram of trees](attachment:tree_diagram.png)
Comparing trees

- **Maximum Compatible Tree (MCT):**
  - if a tree is not binary, several binary trees *refine* it
  - a compatible tree on a subset $X'$ of leaves is a binary tree which refines the trees $T_1$ and $T_2$ restricted to the leaves of $X'$.

  → Polynomial-time algorithm for 2 trees of bounded degree
  → NP-hard if one tree can have arbitrarily large degree

  Hein, Jiang, Wang & Zhang, On the complexity of comparing evolutionary trees, 1996
  → $O(\min\{3^p kn, 2.27^p + kn^3\})$ for $k$ trees, where $p$ is the number of leaves to remove

  Berry & Nicolas, Improved parametrized complexity of the maximum agreement subtree and maximum compatible tree problems, 2006

- **Tanglegrams:**
  - display both trees for visual comparison, linking their leaves with edges, minimizing edge crossings.
  - general problem NP-complete
  - planar embedding in linear time
  - if one tree is fixed, $O(n \log n)$

Venkatachalam, Apple, St John, Gusfield, Untangling tanglegrams: comparing trees by their drawings, 2010
Comparing trees

Distances between trees:

• **Robinson Foulds distance** between T1 and T2:
  - Number of different splits ("symmetric difference metric")
  - Minimum number of edge contractions/decontractions to go from T1 to T2

• **quartet distance** between T1 and T2:
  - Number of different quartets
  → computed in $O(dn \log n)$ for trees of max degree $d$

→ diameter of the quartet distance?

  • Conjecture: at most $(2/3+o(1)) \text{ BINOM}(n,4)$
  • 2014: $> 2/3 \text{ BINOM}(n,4)$
  → at most $(0.9+o(1)) \text{ BINOM}(n,4)$
  • 2016: at most $(0.69+o(1)) \text{ BINOM}(n,4)$
  → at most $(2/3+o(1)) \text{ BINOM}(n,4)$ for caterpillars
  • 2019: strongly explicit example for $> 2/3 \text{ BINOM}(n,4)$

Robinson and Foulds, Comparison of phylogenetic trees, 1981
Brodal, Fagerberg, Pedersen, Mailund and Sand, SODA 2013
Alon, Snir & Yuster, SODA 2014
Alon, Naves & Sudakov, SODA 2016
Chor, Erdős & Komornik, Ann. Comb. 2019
Comparing trees

Distances between trees:

- **Robinson Foulds distance** between T1 and T2:
  - Number of different splits (“symmetric difference metric”)
  - Minimum number of edge contractions/decontractions to go from T1 to T2

- **quartet distance** between T1 and T2:
  - Number of different quartets
  → computed in $O(dn \log n)$ for trees of max degree $d$

- **SPR distance** between T1 and T2:
  - Minimum number of SPR moves to go from T1 to T2
  → NP-hard

→ computed in $O(2.42^k k + n^3)$ for 2 rooted binary trees with SPR distance $k$
Comparing trees

**Day's algorithm**

Linear-time computation of the Robinson Foulds distance between two rooted binary trees $T_1$ and $T_2$:

- relabel leaves of $T_1$ (from left to right) from 1 to $n$, relabel $T_2$'s leaves accordingly
- represent the leaves below each internal node $x$ of $T_1$ as an interval $[i_x, j_x]$
- store these intervals in an array $t$: $[i_x, j_x]$ stored in $t[i]$ if $x$ is the right child of its parent, in $t[j]$ otherwise
- visit the internal nodes of $T_2$, finding the minimum $m$, the maximum $M$, and the number $l$ of leaves below each of them: if $l = M - m + 1$ and the corresponding interval exists in $t$ (in the $m$'th or in the $M$'th cell), then the interval is present both in $T_1$ and $T_2$. Otherwise it corresponds to a cluster in $T_2$ but not in $T_1$. 

![Diagram of trees](image)
Exploring the tree space

**NNI: nearest neighbor interchange**
Consider an edge $e$ and exchange the adequate subtrees connected to $e$.

**SPR: subtree pruning and regrafting**
Disconnect a subtree and reattach it somewhere else.

**TBR: tree bisection and reconnection**
Delete an edge in the tree, reconnect the two parts with a new edge anywhere.

An NNI is a special kind of SPR, which is a special kind of TBR.

NNIs allow to explore the whole tree space. *Proof: induction on...*
Exploring the tree space... to find the optimal tree

Exploring the tree space is useful to find the optimal topology for:

• **Parsimony**  
  Given the tree topology, find the scenario which explains current genetic sequences with the minimum number of operations along the tree edges

• **Likelihood**  
  Given the tree topology and a statistical model of evolution, find the scenario which produces current genetic sequences with the highest probability

  Models of evolution: Jukes Cantor’69, Kimura’80, Felsenstein’81

• **Distance optimization**  
  Given the tree topology, find edge lengths which best explain distance data between current genetic sequences

  Bootstrap: apply the same algorithm on “resampled” data

  Felsenstein, Inferring phylogenies, 2002, 39-44

Exploring the tree space... by randomly generating trees

Which model do you choose to randomly generate (rooted binary) trees?

→ *Random tree generation also used to simulate data to test algorithms!*

- **Labeled tree **equi-probability**

- **Yule-Harding model**:  
  - start from a root with two labeled children  
  - choose one leaf at random to split it, creating a new labeled leaf

- **Kingman’s coalescent model** (*population genetics*):  
  - start from a population of \( n \) leaves (each leaf representing a gene copy)  
  - the probability that two gene copies come from the same copy in the previous generation is \( 1/2n \)

  equivalent to repeating, for \( k \) gene copies:  
  - go back \( \approx 4n/(k(k-1)) \) generations in time  
  - combine 2 random lineages  
  - decrease \( k \) by 1


Exercise – Tree shapes and random generation models

Q1. Evaluate the probability of each rooted binary tree shape on 4 leaves, for each of the three random generation models.

Q2. Evaluate the probability of the rooted binary caterpillar tree (i.e. a tree where no node has two children having two children) on $n$ leaves with the tree equiprobability model, the Yule-Harding model as well as with Kingman’s coalescent model.
Dealing with real data to build the tree of life

The model of evolution seen so far is too simple, not only mutations but also:
• deletions
• insertions
• duplications (paralogs), tandem duplications
• inversions
• translocations
• gene transfer across species / hybridization

Differences (number of leaves, tree topology, etc.):
• between gene trees
• between gene tree and species tree

→ “Tree of 1 percent” (31-protein tree of life)
→ Consensus tree (same leaf set) / supertree (partial leaf sets)
→ Reconciliation between trees
→ Duplication/Loss/Transfer models

Dagan & Martin, The tree of one percent, 2006
Consensus trees

How to combine information from several gene trees on the same set of taxa?

• **Strict consensus tree**: keep only splits which appear in all unrooted trees

• **Majority consensus tree**: keep splits which appear in more than half the trees

→ The resulting splits are compatible
Consensus trees

How to combine information from several gene trees on the same set of taxa?

• **Strict consensus** tree: keep only splits which appear in all unrooted trees

• **Majority consensus** tree: keep splits which appear in more than half the trees
  → The resulting splits are compatible
  (suppose they are not and consider two incompatible splits: as those two splits both appear in a majority of trees, they are the splits of one common tree at least, so they are compatible: contradiction)

• **Adams consensus** tree (for rooted trees):
  - for all trees, consider its maximal clusters for inclusion
  - considering all non-empty intersections between maximal clusters, one gets a partition of $X$
  - apply the same procedure recursively on each set of the partition

$O(kn^2)$ algorithm for $k$ trees improved to $O(kn \log n)$ in 2017

Adams, Consensus Techniques and the Comparison of Taxonomic Trees, 1972
Adams, N-Trees as Nestings: Complexity, Similarity and Consensus, 1986
Jansson, Li & Sung, On Finding the Adams Consensus Tree, 2017
Gusfield's algorithm for perfect phylogeny

The **perfect phylogeny** problem

Matrix $M$ of binary sequences with $n$ lines (species) and $m$ columns (characters). Given $M$, decide if there exists a tree and binary sequences labeling internal nodes such that:
- the root is labeled by a sequence of only zeros
- each character may change only once from a zero to a one, never from a one to a zero, from a parent to a child in the tree.

Gusfield's test for perfect phylogeny in $O(nm)$:
- sort the columns of $M$ in decreasing order (radix sort), considering them as binary numbers
- remove duplicate columns to get matrix $M'$
  - for each cell $M'_{ij}=1$, define $S_{ij} = \{ k < j \mid M'_{ik} = 1 \}$
  - $L_{ij} = \max S_{ij}$ if $S_{ij}$ not empty, 0 otherwise
- for each column $j$, $L_j = \max \{ L_{ij} \mid M'_{ij} = 1 \}$
- check if $L_{ij} = L_j$ for each $M'_{ij} = 1$
Linear time preprocessing to answer **lowest common ancestor queries** in constant time.

**Linear-time reduction** to Minimum Range Queries with a -/+ 1 difference (find the minimum in table $T$ between indices $i$ and $j$):
- Eulerian tour of the tree storing node depth
- Minimum Range Queries with a -/+ 1 difference

Solve Minimum Range Queries in **constant time** with $O(n \log n)$ preprocessing
- Compute & store minimums for all intervals of range a power of 2: $O(n \log n)$
- Answer queries in constant time combining 2 intervals (starts in $i$ + ends in $j$)

Strategy to obtain **linear** space and preprocessing time:
- Group table cells into groups of size $\frac{1}{2} \log n$
- Compute the min for each group: table of size $O(n / \log n) \rightarrow O(n)$ time
- For each group, subtract the first element:
  - each group starts with 0: limited number of group types
  - store the location of the min for each group type in a lookup table

Harel & Tarjan, Fast Algorithms for Finding Nearest Common Ancestors, 1984
Bender & Farach-Colton, The LCA Problem Revisited, 2000
Erik Demaine: Advanced Data Structures course at MIT, 2012
https://courses.csail.mit.edu/6.851/spring12/lectures/
Maximizing triplet consistency

We have seen the BUILD algorithm to reconstruct a tree from its triplets.

Can we reconstruct the tree if there are errors in the triplets?

**Triplet edition problem:**
*Input:* set $X$ of leaves, set $R$ of triplets, positive integer $k \leq n^3$.
*Output:* yes if there exists a tree containing $k$ triplets of $R$, no otherwise.

Triplet edition is NP-complete:
- In NP: check in polynomial time that a solution is correct $\rightarrow$ BUILD algorithm!
- NP-hard: reduction from Cyclic ordering
Maximizing triplet consistency

Triplet edition problem:
**Input:** set $X$ of leaves, set $R$ of triplets, positive integer $k \leq n^3$.
**Output:** yes if there exists a tree containing $k$ triplets of $R$, no otherwise.

Cyclic ordering problem:
**Input:** set $A$ of elements, set $C$ of ordered triples $(a,b,c)$ of distinct elements of $A$.
**Output:** yes if there exists a bijection $f: A \rightarrow [1..|A|]$ such that for each $(a,b,c)$ in $C$, either $f(a)<f(b)<f(c)$ or $f(b)<f(c)<f(a)$ or $f(c)<f(a)<f(b)$.

Reduction:
Given an instance of the Cyclic ordering problem, build an instance of the Triplet edition problem:
- $X = A \cup \{x_0, x_1, x_2, \ldots, x_{|C|}\}$, $k = |A|((|A|-1)/2+2)|C|$;
- for all $a \neq b$ in $X$, add $b|ax_0$ and $a|bx_0$ to $R$;
- for each $i$ in $[1..|C|]$, add $b|ax_i$, $c|bx_i$ and $a|cx_i$ to $R$.

Jansson, On the Complexity of inferring rooted evolutionary trees, 2001
Maximizing triplet consistency

Removing $k$ triplets to obtain a triplet set consistent with a tree?

NP-complete

Bryant, Building trees, hunting for trees, and comparing trees: theory and methods in phylogenetic analysis, 1997
Jansson, On the Complexity of inferring rooted evolutionary trees, 2001
Wu, Constructing the maximum consensus tree from rooted triples, 2004

But fixed-parameter tractable using the “obstructions” (or “conflicts”) on dense instances (one triplet for each set of 3 leaves):
“do not contain \{ab\mid c, b\mid cd, a\mid bd\} or \{ab\mid c, b\mid cd, ad\mid b\}”
Maximizing triplet consistency

Removing $k$ triplets to obtain a triplet set consistent with a tree?

NP-complete

Bryant, Building trees, hunting for trees, and comparing trees: theory and methods in phylogenetic analysis, 1997
Jansson, On the Complexity of inferring rooted evolutionary trees, 2001
Wu, Constructing the maximum consensus tree from rooted triples, 2004

But fixed-parameter tractable using the “obstructions” (or “conflicts”) on dense instances (one triplet for each set of 3 leaves):
“do not contain $\{ab|c, b|cd, a|bd\}$ or $\{ab|c, b|cd, ad|b\}$”

“Bounded search tree” algorithm:
while there is a conflict, solve it in all possible ways
Maximizing triplet consistency

Removing $k$ triplets to obtain a triplet set consistent with a tree?

NP-complete

But fixed-parameter tractable using the “obstructions” (or “conflicts”) on dense instances (one triplet for each set of 3 leaves):
“do not contain \{ab|c, b|cd, a|bd\} or \{ab|c, b|cd, ad|b\}”

“Bounded search tree” algorithm:
while there is a conflict, solve it in all possible ways

$\Rightarrow O(6^k \text{poly}(n))$ time algorithm

Optimized algorithm in $O(n^4)+2^{O(k^{1/3} \log k)}$

Bryant, Building trees, hunting for trees, and comparing trees: theory and methods in phylogenetic analysis, 1997
Jansson, On the Complexity of inferring rooted evolutionary trees, 2001
Wu, Constructing the maximum consensus tree from rooted triples, 2004
Guillemot & Mnich, Kernel and fast algorithm for dense triplet inconsistency, 2010
Taking into account horizontal transfer or hybridization

Fit gene trees into a *phylogenetic network* of the species:
- rooted direct acyclic graph with labeled leaves
- contains some vertices with indegree >1: *hybrid vertices*

**Hybridization number**

A tree $T$ is *contained* in $N$ if $T$ can be obtain from $N$ by arc contractions & deletions.

Given an integer $k$, and 2 trees $T_1$ and $T_2$, does there exist a *hybridization network* $N$, i.e. a phylogenetic network which contains the two trees, with at most $k$ hybrid vertices?

Hybridization number: minimum $k$ for a hybridization network of $T_1$ and $T_2$.

Computing the hybridization number is NP-complete but fixed-parameter tractable.

Property: $\text{HybridizationNumber}(T_1,T_2) \geq d_{\text{SPR}}(T_1,T_2)$

---

Bordewich & Semple, Computing the minimum number of hybridization events for a consistent evolutionary history, 2007

Bordewich & Semple, Computing the hybridization number of two phylogenetic trees is fixed-parameter tractable, 2007
**Property:** \( \text{HybridizationNumber}(T1,T2) \geq \text{SPR}(T1,T2) \)

\( \rightarrow \) deduce the SPR scenario from the hybridization network

The converse is false: problems with cycles!

**Example:**

![Diagram](image-url)
**Property:** HybridizationNumber(T1, T2) ≥ SPR(T1, T2)

→ deduce the SPR scenario from the hybridization network

The converse is false: problems with cycles!

Example:
**Property:** HybridizationNumber(T1,T2) \(\geq\) SPR(T1,T2)

\[ \rightarrow \text{deduce the SPR scenario from the hybridization network} \]

The converse is false: problems with cycles!

**Example:**

Hybridization network of T1 and T2?
Property: \(\text{HybridizationNumber}(T_1,T_2) \geq \text{SPR}(T_1,T_2)\)

\[\rightarrow \text{deduce the SPR scenario from the hybridization network}\]

The converse is false: problems with cycles!

Example:
Property: $\text{HybridizationNumber}(T_1, T_2) \geq \text{SPR}(T_1, T_2)$

$\rightarrow$ deduce the SPR scenario from the hybridization network

The converse is false: problems with cycles!

Example:
Property: HybridizationNumber(T1,T2) ≥ SPR(T1,T2)

→ deduce the SPR scenario from the hybridization network

The converse is false: problems with cycles!

Example:
Properties of trees not valid on phylogenetic networks

• Possible to have several lowest common ancestors

• Possible to have a triplet present «twice» in the network

• Several paths between two leaves
  → how to define the distance between them?

→ a lot of problems are **NP-complete** on **phylogenetic networks**

→ a lot of challenges to get **efficient algorithms in practice**
  (FPT algorithms, approximation algorithms, linear programming, heuristics, etc.)

A fundamental problem on phylogenetic networks with recent progress: **TreeContainment**

→ Presentation on *Finding a gene tree in a phylogenetic network*