Solving the Tree Containment Problem for Genetically Stable Networks in Quadratic Time

Philippe Gambette  Andreas D. M. Gunawan  Anthony Labarre
Stéphane Vialette  Louxin Zhang

International Workshop on Combinatorial Algorithms

October 6th, 2015
Context and motivations

- **Phylogenetic trees** are routinely used to represent evolution, but they cannot display exchanges of genetic material between species;
- When these happen, we rely on **phylogenetic networks** instead;

Example (tree)

**Phylogenetic Tree of Life**

(Example tree from Wikimedia)

Example (network)

(Example network from The Genealogical World of Phylogenetic Networks)

- We still need to verify that the network "contains" a prescribed set of trees to ensure consistency with previous biological knowledge;
A **phylogenetic network** is a rooted DAG with a labelled leaf set \( \{\ell_1, \ell_2, \ldots, \ell_k\} \).

![Diagram of a phylogenetic network](image)

We only consider **binary** networks and trees, i.e. all internal nodes have degree three.
A **phylogenetic network** is a rooted DAG with a labelled leaf set \( \{\ell_1, \ell_2, \ldots, \ell_k\} \).

We only consider **binary** networks and trees, i.e. all internal nodes have degree three.
Phylogenetic networks and related concepts

A **phylogenetic network** is a rooted DAG with a labelled leaf set \( \{\ell_1, \ell_2, \ldots, \ell_k\} \).

We only consider **binary** networks and trees, i.e. all internal nodes have degree three.
Phylogenetic networks and related concepts

A **phylogenetic network** is a rooted DAG with a labelled leaf set \( \{\ell_1, \ell_2, \ldots, \ell_k\} \).

We only consider **binary** networks and trees, i.e. all internal nodes have degree three.
A **phylogenetic network** is a rooted DAG with a labelled leaf set \( \{\ell_1, \ell_2, \ldots, \ell_k\} \).

- **root**: indegree 0;
- **tree nodes**: indegree 1, outdegree 2;
- **reticulations**: indegree 2, outdegree 1;
- **leaves**: outdegree 0;

We only consider **binary** networks and trees, i.e. all internal nodes have degree three.
A **subdivision** of a tree $T$ is a tree $T'$ obtained by inserting any number of vertices into the edges of $T$.

**Example (a tree and a subdivision)**

![Diagram](image)
The **tree containment** problem

Network $N$ **displays** tree $T$ if we can obtain a subdivision of $T$ by removing incoming edges from reticulations and “dummy leaves”.

\[
\begin{array}{c}
\ell_1 \\
\ell_2 \\
\ell_3 \\
\ell_4 \\
\ell_5
\end{array}
\]

\[
\begin{array}{c}
\ell_1 \\
\ell_2 \\
\ell_3 \\
\ell_4 \\
\ell_5
\end{array}
\]
The **TREE CONTAINMENT** problem

Network $N$ **displays** tree $T$ if we can obtain a subdivision of $T$ by removing incoming edges from reticulations and “dummy leaves”.

Network $N$ displays tree $T$ if we can obtain a subdivision of $T$ by removing incoming edges from reticulations and “dummy leaves”.
Network $N$ \textbf{displays} tree $T$ if we can obtain a subdivision of $T$ by removing incoming edges from reticulations and “dummy leaves”.

\textbf{The TREE CONTAINMENT problem}
The TREE CONTAINMENT problem

Network $N$ displays tree $T$ if we can obtain a subdivision of $T$ by removing incoming edges from reticulations and “dummy leaves”.

Problem (TREE CONTAINMENT)

**Input:** a phylogenetic network $N$, a phylogenetic tree $T$.

**Question:** does $N$ display $T$?
TREE CONTAINMENT prior to this work

\[ A \rightarrow B \] class \( A \) contains class \( B \)

- solvable in polynomial time
- in \( P \) by class inclusion
- NP-complete

(Adapted from http://phylnet.univ-mlv.fr/isiphync by Philippe Gambette)
Our contributions

1. **genetically stable** (GS) networks;
2. inclusion relations w.r.t. other classes;
3. **TREE CONTAINMENT** in P for GS networks;

\[ A \rightarrow B \quad \text{class A contains class B} \]

- solvable in polynomial time
- in P by class inclusion
- NP-complete

(adapted from http://phylnet.univ-mlv.fr/isiphync by Philippe Gambette)
Genetically stable networks

A node \( v \) in a network \( N \) is \textbf{stable on a leaf} \( \ell \) if every path from the root to \( \ell \) contains \( v \).
Genetically stable networks

A node \( v \) in a network \( N \) is **stable on a leaf** \( \ell \) if every path from the root to \( \ell \) contains \( v \). A network \( N \) is **genetically stable** if every reticulation has a stable parent (on any leaf).
Genetically stable networks

A node \( v \) in a network \( N \) is **stable on a leaf** \( \ell \) if every path from the root to \( \ell \) contains \( v \). A network \( N \) is **genetically stable** if every reticulation has a stable parent (on any leaf).

A GS network

\[
\begin{align*}
\ell_1 & \quad \ell_2 & \quad \ell_3 & \quad \ell_4 \\
\quad & \quad & a & \quad d \\
\quad & b & & \quad \\
\quad & & c & \quad \quad \\
& & & \quad \quad \\
& & & \quad \quad \\
\end{align*}
\]

- \( a, b, c \) stable on \( \ell_2 \)
- \( d \) stable on \( \ell_4 \)
Genetically stable networks

A node \( v \) in a network \( N \) is **stable on a leaf** \( \ell \) if every path from the root to \( \ell \) contains \( v \). A network \( N \) is **genetically stable** if every reticulation has a stable parent (on any leaf).

A GS network

A non-GS network

\( \ell_2 \) can be reached through either \( a \) or \( b \)

no other leaf “needs” \( a \) or \( b \)
Overview of the algorithm

The subtree induced by two sibling leaves \( \ell, \ell' \) and their parent \( \alpha \) in a tree is called a **cherry**, and is denoted by \( \{\alpha, \ell, \ell'\} \).
Overview of the algorithm

The subtree induced by two sibling leaves $\ell, \ell'$ and their parent $\alpha$ in a tree is called a **cherry**, and is denoted by $\{\alpha, \ell, \ell'\}$.

Algorithm for **TREE CONTAINMENT** in GS networks

1. Select a cherry $C = \{\alpha, \ell, \ell'\}$ in $T$;
2. If there is no **match** for $C$ in $N$, report **NO**;
3. Otherwise, **remove** the match from $N$ and $C$ from $T$;
4. If $T$ is now a single node, report **YES**, otherwise go back to 1;

Matches and removals are such that $N$ displays $T$ if and only if $N'$ displays $T'$.
Matching cherries: stability helps

Stability narrows down choices for matching $\alpha$, $(\alpha, \ell_1)$ and $(\alpha, \ell_2)$ in $N$:

Lemma (1)

If $N$ displays $T$ through some subdivision $T'$, then $\alpha$ must be matched to a node $p$ such that:

1. $\ell_1$ and $\ell_2$ are the only leaves on which $p$ can be stable;
2. $\ell_1$ is the only leaf on which vertices in $P_1 \setminus \{p\}$ can be stable;
3. $\ell_2$ is the only leaf on which vertices in $P_2 \setminus \{p\}$ can be stable.
Matching cherries: **genetic** stability helps

Lemma (1) allows us to focus on **specific** paths, i.e. paths $P$ from $x$ to $\ell$ such that each vertex in $P \setminus \{x\}$ is either stable only on $\ell$ or not stable at all. What if several choices exist?

![Diagram](image)

**Lemma (2)**

*If $N$ is genetically stable and contains vertices $x$ and $y$ connected to leaves $\ell_1$ and $\ell_2$ through specific paths that only intersect at $x$ (resp. $y$), then either $y \in P_1 \cup P_2$ or $x \in Q_1 \cup Q_2$.***
Modifying \( N \) and \( T \) when \( N \) is genetically stable

Lemma (2) allows us to restrict our search to the lowest common ancestor \( p \) of \( \ell_1 \) and \( \ell_2 \) such that paths \( p \leadsto \ell_1 \) and \( p \leadsto \ell_2 \) in \( N \) are specific.

\[
T : \begin{array}{c}
\alpha \\
\ell_1 & \ell_2
\end{array}
\]

\[
N : \begin{array}{c}
p \\
\ell_1 & \ell_2
\end{array}
\]

Lemma (3)

If \( p \), \( P_1 \) and \( P_2 \) match \( \alpha \), \((\alpha, \ell_1)\) and \((\alpha, \ell_2)\) in a GS network \( N \), then \( N \) displays \( T \) if and only if \( N \setminus P_1 \setminus P_2 \) displays \( T \setminus \{\ell_1, \ell_2\} \).
Finding a match for $\alpha$, $(\alpha, \ell_1)$ and $(\alpha, \ell_2)$ in $N$

1. Move up from $\ell_1$ until we find a lowest common ancestor of $\ell_1$ and $\ell_2$ connected to $\ell_2$ by a path free of nodes stable on other leaves;

2. Move up from $\ell_2$ to $w_1$ while remaining in a specific path to $\ell_2$;

3. If we succeed, we obtain two specific paths to $\ell_1$ and $\ell_2$ in $N$;
Correctness and running time

The previous lemmas prove the correctness of the algorithm.

Algorithm for **TREE CONTAINMENT** in GS networks

1. Select a cherry $C = \{\alpha, \ell, \ell'\}$ in $T$;
2. If there is no **match** for $C$ in $N$, report **NO**;
3. Otherwise, **remove** the match from $N$ and $C$ from $T$;
4. If $T$ is now a single node, report **YES**, otherwise go back to 1;

The running time is dominated by checking stability, which implies a running time of $O(|V| \cdot (|E| + |V|)) = O(|L|^2)$ where $|L|$ is the number of leaves of $N$. 
Relevance of GS networks

A fair amount of real-world networks could be genetically stable:

![Graph showing percentage of phylogenetic networks on n leaves generated with the coalescent with recombination model (recombination rate r) in each class.](image-url)
Future work

- Major open problem: complexity for reticulation-visible networks;
- Refine hardness results;
- Improve the complexity for tractable cases;