Combinatorial and Algorithmic Issues of Reticulation-visible Networks

Louxin ZHANG
Department of Mathematics
National University of Singapore
matzlx@nus.edu.sg
A (binary) phylogenetic network is a rooted directed acyclic graph:

-- leaves: $\ell_1, \ell_2, \ell_3, \ell_4$
-- tree nodes: $t_1, t_2, t_3, t_4, \rho$
-- reticulation nodes: $r_1, r_2$

Phylogenetic trees are binary networks without reticulation.
Three classes of networks are investigated in

-- designing algorithms for inferring phylogenetic networks,
-- comparison of networks
-- validation of network models

Huson, Klopper, *RECOMB*, 2007
Huson, Rupp, Scornavacca, *Phylogenetic Networks*, 2010
Today’s Talk

- Introduction to the network classes
- Combinatorial characterization of network classes
- Algorithms for
  -- the tree containment problem
  -- the cluster containment problem
- Concluding remarks
A network is a **galled tree** if the smallest cycles (ignoring direction) on different reticulation nodes are **node-disjoint**.

Reticulations occur in disjoint regions in galled trees

A network is a **galled network** if the cycle involving \( r \) contains only \( r \) and tree nodes for each reticulation \( r \).

That is, reticulation nodes may appear in the same region, but must not have ancestor-descendent relationship.

Huson, Klopper, *RECOMB*, 2007
A node \( u \) is \textbf{stable} if there is a leaf \( \ell \) such that every path \( P(\rho, \ell) \) contains \( u \).

A network is
- \textbf{ret.-visible} if every ret. node is stable
- \textbf{nearly-stable} if for each node \( u \), either it or all parents are stable.
Binary phylogenetic networks

- Nearly stable
- Tree-sibling
- Reticulation-visible
- Genetically stable
- Nearly tree-child
- Tree-child
- Normal
- Level k
- Galled network
- Level 2
- Level 1
- Tree
III Characterization of Reticulation-visible Networks

\[ N: \] A reticulation-visible network
\[ \mathcal{R}_N: \] The set of reticulation nodes in \( N \)
\[ \mathcal{T}_N: \] The set of tree nodes in \( N \)

Consider

\[ N \setminus \mathcal{R}_N: \] The subnetwork induced by \( \mathcal{T}_N \)
$N$: A binary stable network
$\mathcal{R}_N$: The set of reticulation nodes
$\mathcal{T}_N$: The set of tree nodes

$N - \mathcal{R}_N$: The subnetwork induced by $\mathcal{T}_N$

- A reticulation is **inner** if its parents are in the same component of $N - \mathcal{R}_N$;
- A reticulation node is **inter** if its parents are in different components of $N - \mathcal{R}_N$;
**Theorem** Let $N$ be a network such that
\[ N - \mathcal{R}_N = C_1 \cup C_2 \cup \cdots \cup C_k. \]

(i) $N$ is ret.-visible iff either an inner-reticulation exits below $C_i$ or a leaf exists within $C_i$ for every $i$.

(ii) $N$ is a galled network iff every reticulation node is inner.

**Lemma** \( \#(\text{tree nodes of } N) = \#(\text{ret. nodes of } N) + (k - 1) \).

**Theorem 1**

(i) If \( N \) is ret.-visible, \( \#(\text{ret. nodes of } N) \leq 3(k - 1). \) (Bordewich & Semple 2016)

(ii) If \( N \) is nearly-stable, \( \#(\text{ret. nodes of } N) \leq 3(k-1). \)

(iii) If \( N \) is galled, \( \#(\text{ret. nodes of } N) \leq 2(k-1). \)

Our proof of Part (i)  By Lemma 1, 

\[ \text{(tree nodes)} = \text{(ret. Nodes)} + (k-1). \]

and so

\[ \text{(ret. nodes)} = \text{(components)} - 1 \]

\[ \leq \frac{\text{(tree nodes)}}{2} + k - 1 \]

\[ \leq \frac{\text{(ret. nodes)} + (k-1)}{2} + k - 1. \]
$N$ : A binary galled network with $k$ leaves.

**Proof of (III)** (for galled networks) follows from the fact:

**FACTS**
(i) Every reticulation is inner
(ii) Every component contains either a tree leaf or at least 3 tree nodes.
Tree Containment Problem (TCP)

**Input:** A network $N$ and a binary tree $T$ with the same leaves.

**Question:** Does $N$ display $T$?
Cluster Containment Problem (CCP)

**Input:** A network $N$ and a subset $S$ of network leaves.

**Question:** Is $S$ a soft-wired cluster of $N$?
Both the TCP and CCP are NP-complete

Naïve \( O\left(\#(\text{edges}) \cdot 2^{\#(\text{ret. nodes})}\right) \) algorithms exist for both TCP and CCP on arbitrary networks
- Both the TCP and CCP are NP-complete
- Naïve $\mathcal{O}(\#(\text{edges}) \cdot 2^{\#(\text{ret. nodes})})$ algorithms exist for both TCP and CCP on arbitrary networks

**Theorem 2** (i) An $\mathcal{O}(\#(\text{edges})^2)$ TCP and $\mathcal{O}(\#(\text{edges}))$ CCP algorithm exist on ret.-visible networks.

(ii) An $\mathcal{O}(\#(\text{edges})^2 \cdot 2^{0.694 \cdot \#(\text{ret. nodes})})$ TCP and $\mathcal{O}(\#(\text{edges}) \cdot 2^{0.5 \cdot \#(\text{ret. nodes})})$ CCP algorithm exist in general.
- Both the TCP and CCP are NP-complete
- Naïve $O((\text{edges}) \cdot 2^{\text{ret. nodes}})$ algorithms exist for both TCP and CCP on arbitrary networks

**Theorem 2**

(i) An $O((\text{edges})^2)$ TCP and $O((\text{edges}))$ CCP algorithm exist on ret.-visible networks.

(ii) An $O((\text{edges})^2 \cdot 2^{0.694 \cdot \text{ret. nodes}})$ TCP and $O((\text{edges}) \cdot 2^{0.5 \cdot \text{ret. nodes}})$ CCP algorithm exist in general.

**Note 1.** The quadratic time alg. in (i) has been improved independently by Weller and Gunawan. They revised on the same step of our algorithm.

2. Bordewich and Semple (2016) published an $O((\text{edges})^3)$ for binary reticulation-visible networks independently.
Trick 1: Tree-Node Decomposition

$N$: A phylogenetic network

The tree nodes (including the root) are partitioned into
-- disjoint components
-- each component has a tree topology

Our algorithm works on these “large” components one by one in bottom-up manner.
Trick 2: The Visibility of Components

A component $X$ is **visible** if a leaf $\ell$ exists such that every path from $\rho$ to $\ell$ passes through $X$.

-- $C$ is invisible
-- $D$ is visible with respect to $\ell_1$
$N$ and $G$: An instance of the TCP

$X$: A component at the bottom
CASE 1

\( N \) and \( G \): An instance of the TCP

\( X \): A visible component at the bottom

Fact 1: In \( O(|E(X)| \cdot |V(G)|) \) time, we can
-- simplify \((N, G)\) by removing \( X \) and the corresponding part in \( G \),
-- or conclude that \( G \) is not displayed in \( N \).
CASE 2

**X**: An exposed, but invisible component

Select a reticulation node \( r \) below \( X \), say the parent of \( \ell_2 \).

-- Simplify \( N \) by removing an edge entering \( r \), obtaining two networks

---

**Fact 2:**

(a) \( X \) is visible w.r.t. \( \ell_2 \) in \( N' \). Pruning \( X \) reduces at least 2 ret. nodes.

(b) \( Y \) is visible w.r.t. \( \ell_2 \) in \( N'' \). Pruning \( Y \) reduces at least 1 ret. nodes.

(c) A tree is displayed in \( N \) if and only if it is displayed in \( N' \) or in \( N'' \).
Algorithmic and combinatorial work we have done

Available computer programs on Gambette’s Who is Who site

http://igm.univ-mlv.fr/~gambette/PhylnetServices/

-- Generate a random network,
-- Check network property
-- Check if a tree is displayed in an arbitrary network
-- Compute the soft version of the Robinson-Foulds distance between two networks

Is the work useful for reconstruction of phylogenetic networks?
The stability property also arises from program optimization in computer science.

In a flow graph with the root $\rho$, a node $u$ dominates another node $v$ if every path from $\rho$ to $v$ passes $u$.

Dominators in Directed Graphs: A Survey of Recent Results, Applications, and Open Problems

Using food web dominator trees to catch secondary extinctions in action

Antonio Bodini$^1$,*, Michele Bellingeri$^1$, Stefano Allesina$^2$ and Cristina Bondavalli$^1$
Thank You