Comparing Phylogenetic Networks by Counting Triangles

Jesper Jansson
(The Hakubi Project & Akutsu Lab., Kyoto University, Japan)

Joint work with:
Andrzej Lingas (Lund University, Sweden)

2015-07-27
Structure of talk:

PART I: How to count monochromatic and “almost-monochromatic” triangles in an undirected, edge-colored graph
Structure of talk:

PART I: How to count monochromatic and “almost-monochromatic” triangles in an undirected, edge-colored graph

PART II: Phylogenetic trees and networks
Structure of talk:

PART I : How to count monochromatic and “almost-monochromatic” triangles in an undirected, edge-colored graph

PART II : Phylogenetic trees and networks

PART III : Computing the rooted triplet distance between galled trees by counting triangles
PART I: How to count monochromatic and “almost-monochromatic” triangles in an undirected, edge-colored graph
Counting triangles

<table>
<thead>
<tr>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td><em>A triangle</em> in an undirected graph is a cycle of length 3.</td>
</tr>
</tbody>
</table>
Counting triangles

Definition

A triangle in an undirected graph is a cycle of length 3.

Example:
Definition

A triangle in an undirected graph is a cycle of length 3.

Example:

This is a triangle.
Counting triangles

Definition

A triangle in an undirected graph is a cycle of length 3.

Example:

This is another triangle.
Definition
A triangle in an undirected graph is a cycle of length 3.

Example:

In total, the above graph has 4 triangles.
Counting triangles

Definition
A triangle in an undirected graph is a cycle of length 3.

Example:

In total, the above graph has 4 triangles.

Applications of triangle counting:
Graph mining, social network analysis, query plan optimization in databases, graph comparison, etc.
Definition

A triangle in an undirected graph is a cycle of length 3.

Let $G = (V, E)$ be an undirected, connected graph with $m$ edges and $n$ vertices. How fast can we count the number of triangles in $G$?
Definition

A triangle in an undirected graph is a cycle of length 3.

Let $G = (V, E)$ be an undirected, connected graph with $m$ edges and $n$ vertices. How fast can we count the number of triangles in $G$?

Naive algorithm: $O(n^3)$ time.
A triangle in an undirected graph is a cycle of length 3.

Let $G = (V, E)$ be an undirected, connected graph with $m$ edges and $n$ vertices. How fast can we count the number of triangles in $G$?

**Naive algorithm:** $O(n^3)$ time.

**Faster algorithm:** [Alon, Yuster, Zwick; 1997]

- Let $A$ be the adjacency matrix of $G$ (i.e., $A$ is a binary $(n \times n)$-matrix such that $A[u, v] = 1$ iff $\{u, v\} \in E$).
A triangle in an undirected graph is a cycle of length 3.

Let $G = (V, E)$ be an undirected, connected graph with $m$ edges and $n$ vertices. How fast can we count the number of triangles in $G$?

**Naive algorithm:** $O(n^3)$ time.

**Faster algorithm:** [Alon, Yuster, Zwick; 1997]

- Let $A$ be the adjacency matrix of $G$ (i.e., $A$ is a binary $(n \times n)$-matrix such that $A[u, v] = 1$ iff $\{u, v\} \in E$).
- Let $A^3 = A \times A \times A$. Then $(A^3)[u, v] =$ the number of walks of length 3 from $u$ to $v$. 

Therefore, $\frac{1}{6} \cdot \text{tr}(A^3)$ gives the number of triangles in $G$, where $\text{tr}(X)$ (“the trace of $X$”) is the sum of all elements on the main diagonal of matrix $X$. 

Definition

A triangle in an undirected graph is a cycle of length 3.

Let $G = (V, E)$ be an undirected, connected graph with $m$ edges and $n$ vertices. How fast can we count the number of triangles in $G$?

Naive algorithm: $O(n^3)$ time.

Faster algorithm: [Alon, Yuster, Zwick; 1997]
- Let $A$ be the adjacency matrix of $G$ (i.e., $A$ is a binary $(n \times n)$-matrix such that $A[u, v] = 1$ iff $\{u, v\} \in E$).
- Let $A^3 = A \times A \times A$. Then $(A^3)[u, v] = $ the number of walks of length 3 from $u$ to $v$.
- Therefore, $\frac{1}{6} \cdot \text{tr}(A^3)$ gives the number of triangles in $G$, where $\text{tr}(X)$ (“the trace of $X$”) is the sum of all elements on the main diagonal of matrix $X$. 
Example:

Adjacency matrix:

\[
A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\]
Example:

Adjacency matrix:

\[
A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 \\
\end{pmatrix}
\]

\[
\Rightarrow A^3 = \begin{pmatrix}
0 & 0 & 5 & 2 & 2 & 2 \\
0 & 0 & 5 & 2 & 2 & 2 \\
5 & 5 & 6 & 9 & 9 & 9 \\
2 & 2 & 9 & 6 & 7 & 7 \\
2 & 2 & 9 & 7 & 6 & 7 \\
2 & 2 & 9 & 7 & 7 & 6 \\
\end{pmatrix}
\]
Example:

Adjacency matrix:

\[
A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 \\
\end{pmatrix}
\]

\[
A^3 = \begin{pmatrix}
0 & 0 & 5 & 2 & 2 & 2 \\
0 & 0 & 5 & 2 & 2 & 2 \\
5 & 5 & 6 & 9 & 9 & 9 \\
2 & 2 & 9 & 6 & 7 & 7 \\
2 & 2 & 9 & 7 & 6 & 7 \\
2 & 2 & 9 & 7 & 7 & 6 \\
\end{pmatrix}
\]
Example:

Adjacency matrix:

\[ A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 \\
\end{pmatrix} \]

\[ A^3 = \]

\[ \begin{pmatrix}
0 & 0 & 5 & 2 & 2 & 2 \\
0 & 0 & 5 & 2 & 2 & 2 \\
5 & 5 & 6 & 9 & 9 & 9 \\
2 & 2 & 9 & 6 & 7 & 7 \\
2 & 2 & 9 & 7 & 6 & 7 \\
2 & 2 & 9 & 7 & 7 & 6 \\
\end{pmatrix} \]
Example:

Adjacency matrix:

\[
A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 \\
\end{pmatrix}
\]

\[A^3 = \begin{pmatrix}
0 & 0 & 5 & 2 & 2 & 2 \\
0 & 0 & 5 & 2 & 2 & 2 \\
5 & 5 & 6 & 9 & 9 & 9 \\
2 & 2 & 9 & 6 & 7 & 7 \\
2 & 2 & 9 & 7 & 6 & 7 \\
2 & 2 & 9 & 7 & 7 & 6 \\
\end{pmatrix}\]
Example:

Adjacency matrix:

\[
A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\]

\[
A^3 = \begin{pmatrix}
0 & 0 & 5 & 2 & 2 & 2 \\
0 & 0 & 5 & 2 & 2 & 2 \\
5 & 5 & 6 & 9 & 9 & 9 \\
2 & 2 & 9 & 6 & 7 & 7 \\
2 & 2 & 9 & 7 & 6 & 7 \\
2 & 2 & 9 & 7 & 7 & 6
\end{pmatrix}
\]
Example:

Adjacency matrix:

\[
A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\]

\[
A^3 = \begin{pmatrix}
0 & 0 & 5 & 2 & 2 & 2 \\
0 & 0 & 5 & 2 & 2 & 2 \\
5 & 5 & 6 & 9 & 9 & 9 \\
2 & 2 & 9 & 6 & 7 & 7 \\
2 & 2 & 9 & 7 & 6 & 7 \\
2 & 2 & 9 & 7 & 7 & 6
\end{pmatrix}
\]
Example:

Adjacency matrix:

\[
A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0 \\
\end{pmatrix}
\]

\[
\Rightarrow A^3 = \begin{pmatrix}
0 & 0 & 5 & 2 & 2 & 2 \\
0 & 0 & 5 & 2 & 2 & 2 \\
5 & 5 & 6 & 9 & 9 & 9 \\
2 & 2 & 9 & 6 & 7 & 7 \\
2 & 2 & 9 & 7 & 6 & 7 \\
2 & 2 & 9 & 7 & 7 & 6 \\
\end{pmatrix}
\]
Example:

Adjacency matrix:

$$A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}$$

$$A^3 = \begin{pmatrix}
0 & 0 & 5 & 2 & 2 & 2 \\
0 & 0 & 5 & 2 & 2 & 2 \\
5 & 5 & 6 & 9 & 9 & 9 \\
2 & 2 & 9 & 6 & 7 & 7 \\
2 & 2 & 9 & 7 & 6 & 7 \\
2 & 2 & 9 & 7 & 7 & 6
\end{pmatrix}$$
Example:

Adjacency matrix:

\[
A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\]

\[
A^3 = \begin{pmatrix}
0 & 0 & 5 & 2 & 2 & 2 \\
0 & 0 & 5 & 2 & 2 & 2 \\
5 & 5 & 6 & 9 & 9 & 9 \\
2 & 2 & 9 & 6 & 7 & 7 \\
2 & 2 & 9 & 7 & 6 & 7 \\
2 & 2 & 9 & 7 & 7 & 6
\end{pmatrix}
\]
Example:

\[
\begin{align*}
&\text{Adjacency matrix:} \\
&A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\end{align*}
\]

\[
\Rightarrow A^3 = \begin{pmatrix}
0 & 0 & 5 & 2 & 2 & 2 \\
0 & 0 & 5 & 2 & 2 & 2 \\
5 & 5 & 6 & 9 & 9 & 9 \\
2 & 2 & 9 & 6 & 7 & 7 \\
2 & 2 & 9 & 7 & 6 & 7 \\
2 & 2 & 9 & 7 & 7 & 6
\end{pmatrix}
\]
Example:

Adjacency matrix:

\[
A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\]

\Rightarrow A^3 = \begin{pmatrix}
0 & 0 & 5 & 2 & 2 & 2 \\
0 & 0 & 5 & 2 & 2 & 2 \\
5 & 5 & 6 & 9 & 9 & 9 \\
2 & 2 & 9 & 6 & 7 & 7 \\
2 & 2 & 9 & 7 & 6 & 7 \\
2 & 2 & 9 & 7 & 7 & 6
\end{pmatrix}
Example:

Adjacency matrix:

\[
A = \begin{pmatrix}
0 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 1 & 0 & 0 & 0 \\
1 & 1 & 0 & 1 & 1 & 1 \\
0 & 0 & 1 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 1 \\
0 & 0 & 1 & 1 & 1 & 0
\end{pmatrix}
\]

\[
A^3 = \begin{pmatrix}
0 & 0 & 5 & 2 & 2 & 2 \\
0 & 0 & 5 & 2 & 2 & 2 \\
5 & 5 & 6 & 9 & 9 & 9 \\
2 & 2 & 9 & 6 & 7 & 7 \\
2 & 2 & 9 & 7 & 6 & 7 \\
2 & 2 & 9 & 7 & 7 & 6
\end{pmatrix}
\]
Example:

Adjacency matrix:

\[ A = \begin{pmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 1 & 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 0 & 0 & 1 & 1 & 0 & 1 \\ 0 & 0 & 1 & 1 & 1 & 0 \end{pmatrix} \]

\[ A^3 = \begin{pmatrix} 0 & 0 & 5 & 2 & 2 & 2 \\ 0 & 0 & 5 & 2 & 2 & 2 \\ 5 & 5 & 6 & 9 & 9 & 9 \\ 2 & 2 & 9 & 6 & 7 & 7 \\ 2 & 2 & 9 & 7 & 6 & 7 \\ 2 & 2 & 9 & 7 & 7 & 6 \end{pmatrix} \]

\[ \Rightarrow \frac{1}{6} \cdot tr(A^3) = \frac{1}{6} \cdot (0 + 0 + 6 + 6 + 6 + 6) = 4. \]
Definition

A **triangle** in an undirected graph is a cycle of length 3.

Let $G = (V, E)$ be an undirected, connected graph with $m$ edges and $n$ vertices.

**Alon, Yuster, Zwick; 1997:**

- $O(n^{\omega})$-time method to **count** the number of triangles in $G$.
- $O(m^{\frac{2\omega}{\omega+1}})$-time matrix multiplication-based method to **determine** if $G$ contains a triangle, and if so, **output** a triangle.

Here, $\omega$ denotes the exponent in the running time of the fastest existing method for matrix multiplication.
Matrix multiplication

**Input:** Two matrices $A, B$ of size $(n \times n)$ over a field.

**Output:** The matrix product $AB$. 

Major open question: Does an $O(n^2)$-time algorithm exist?
Matrix multiplication

**Input:** Two matrices $A, B$ of size $(n \times n)$ over a field.

**Output:** The matrix product $AB$.

(Below, the time complexity = the number of arithmetic operations.)

- Naive algorithm: $O(n^3)$ time
Matrix multiplication

**Input:** Two matrices $A, B$ of size $(n \times n)$ over a field.

**Output:** The matrix product $AB$.

(Below, the time complexity = the number of arithmetic operations.)

- Naive algorithm: $O(n^3)$ time
- Strassen; 1969: $O(n^{\log_2 7}) = o(n^{2.808})$ time

Major open question: Does an $O(n^2)$-time algorithm exist?
Matrix multiplication

**Input:** Two matrices $A, B$ of size $(n \times n)$ over a field.

**Output:** The matrix product $AB$.

(Below, the time complexity = the number of arithmetic operations.)

- Naive algorithm: $O(n^3)$ time
- Strassen; 1969: $O(n^\log_2 7) = o(n^{2.808})$ time

$$A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}, \quad B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \quad \Rightarrow \quad AB = \begin{pmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{pmatrix}$$
Matrix multiplication

**Input:** Two matrices $A, B$ of size $(n \times n)$ over a field.

**Output:** The matrix product $AB$.

(Below, the time complexity = the number of arithmetic operations.)

- **Naive algorithm:** $O(n^3)$ time
- **Strassen; 1969:** $O(n^{\log_2 7}) = o(n^{2.808})$ time

Given:

\[
A = \begin{pmatrix} A_{1,1} & A_{1,2} \\ A_{2,1} & A_{2,2} \end{pmatrix}, \quad B = \begin{pmatrix} B_{1,1} & B_{1,2} \\ B_{2,1} & B_{2,2} \end{pmatrix} \Rightarrow AB = \begin{pmatrix} A_{1,1}B_{1,1} + A_{1,2}B_{2,1} & A_{1,1}B_{1,2} + A_{1,2}B_{2,2} \\ A_{2,1}B_{1,1} + A_{2,2}B_{2,1} & A_{2,1}B_{1,2} + A_{2,2}B_{2,2} \end{pmatrix}
\]

Define:

\[
\begin{align*}
P_1 &= A_{1,1}(B_{1,2} - B_{2,2}) \\
P_2 &= (A_{1,1} + A_{1,2})B_{2,2} \\
P_3 &= (A_{2,1} + A_{2,2})B_{1,1} \\
P_4 &= A_{2,2}(B_{2,1} - B_{1,1}) \\
P_5 &= (A_{1,1} + A_{2,2})(B_{1,1} + B_{2,2}) \\
P_6 &= (A_{1,2} - A_{2,2})(B_{2,1} + B_{2,2}) \\
P_7 &= (A_{1,1} - A_{2,1})(B_{1,1} + B_{1,2})
\end{align*}
\]

Then:

\[
AB = \begin{pmatrix} -P_2 + P_4 + P_5 + P_6 & P_1 + P_2 \\ P_3 + P_4 & P_1 - P_3 + P_5 - P_7 \end{pmatrix}
\]

**Note:** Only 7 submatrix multiplications are needed! (The naive algorithms uses 8.)

\[
T(n) = 7T(n/2) + \Theta(n^2) = O(n^{\log_2 7}) = o(n^{2.808})
\]
Matrix multiplication

**Input:** Two matrices $A, B$ of size $(n \times n)$ over a field.
**Output:** The matrix product $AB$.

(Below, the time complexity = the number of arithmetic operations.)

- Naive algorithm: $O(n^3)$ time
- Strassen; 1969: $O(n^{\log_2 7}) = o(n^{2.808})$ time
- Pan; 1978: $o(n^{2.796})$ time
- Bini \textit{et al.}; 1979: $o(n^{2.780})$ time
- Schönhage; 1981: $o(n^{2.522})$ time
- Romani; 1982: $o(n^{2.517})$ time
- Coppersmith \& Winograd; 1990: $o(n^{2.376})$ time
- Stothers; 2010: $o(n^{2.374})$ time
- Vassilevska Williams; 2012: $o(n^{2.37288})$ time
- Le Gall; 2014: $o(n^{2.37287})$ time

Major open question: Does an $O(n^2)$-time algorithm exist?
Matrix multiplication

**Input:** Two matrices $A, B$ of size $(n \times n)$ over a field.

**Output:** The matrix product $AB$.

(Below, the time complexity = the number of arithmetic operations.)

- Naive algorithm: $O(n^3)$ time
- Strassen; 1969: $O(n^{\log_2 7}) = o(n^{2.808})$ time
- Pan; 1978: $o(n^{2.796})$ time
- Bini et al.; 1979: $o(n^{2.780})$ time
- Schönhage; 1981: $o(n^{2.522})$ time
- Romani; 1982: $o(n^{2.517})$ time
- Coppersmith & Winograd; 1990: $o(n^{2.376})$ time
- Stothers; 2010: $o(n^{2.374})$ time
- Vassilevska Williams; 2012: $o(n^{2.37288})$ time
- Le Gall; 2014: $o(n^{2.37287})$ time

**Major open question:** Does an $O(n^2)$-time algorithm exist?
Matrix multiplication

**Input:** Two matrices $A, B$ of size $(n \times n)$ over a field.

**Output:** The matrix product $AB$.

(Below, the time complexity = the number of arithmetic operations.)

- Naive algorithm: $O(n^3)$ time
- Strassen; 1969: $O(n^{\log_2 7}) = o(n^{2.808})$ time
- Pan; 1978: $o(n^{2.796})$ time
- Bini et al.; 1979: $o(n^{2.780})$ time
- Schönhage; 1981: $o(n^{2.522})$ time
- Romani; 1982: $o(n^{2.517})$ time
- ... 
- Coppersmith & Winograd; 1990: $o(n^{2.376})$ time
- Stothers; 2010: $o(n^{2.374})$ time
- Vassilevska Williams; 2012: $o(n^{2.37288})$ time
- Le Gall; 2014: $o(n^{2.37287})$ time

**Major open question:** Does an $O(n^2)$-time algorithm exist?
Counting triangles, cont.

Let \( G = (V, E) \) be an undirected, connected graph with \( m \) edges and \( n \) vertices.

Alon, Yuster, Zwick; 1997:

- \( O(n^\omega) \)-time method to *count* the number of triangles in \( G \).

- \( O(m^{\frac{2\omega}{\omega+1}}) \)-time matrix multiplication-based method to *determine* if \( G \) contains a triangle, and if so, *output* a triangle.

\( \omega < 2.373 \) is a famous result from STOC 2012 by V. Vassilevska Williams.
Counting triangles, cont.

Let $G = (V, E)$ be an undirected, connected graph with $m$ edges and $n$ vertices.

Alon, Yuster, Zwick; 1997:

- $O(n^\omega)$-time method to count the number of triangles in $G$.
  
  $\Rightarrow O(n^\omega) \leq o(n^{2.373})$

- $O(m^{2\omega/(\omega+1)}$)-time matrix multiplication-based method to determine if $G$ contains a triangle, and if so, output a triangle.

$\omega < 2.373$ is a famous result from STOC 2012 by V. Vassilevska Williams.
Let $G = (V, E)$ be an undirected, connected graph with $m$ edges and $n$ vertices.

**Alon, Yuster, Zwick; 1997:**

- $O(n^\omega)$-time method to *count* the number of triangles in $G$.
  \[ \Rightarrow O(n^\omega) \leq o(n^{2.373}) \]

- $O(m^{\frac{2\omega}{\omega+1}})$-time matrix multiplication-based method to *determine* if $G$ contains a triangle, and if so, *output* a triangle.
  \[ \Rightarrow O(m^{\frac{2\omega}{\omega+1}}) \leq o(m^{1.408}) \]

$\omega < 2.373$ is a famous result from STOC 2012 by V. Vassilevska Williams.
Counting triangles, cont.

Let $G = (V, E)$ be an undirected, connected graph with $m$ edges and $n$ vertices.

Alon, Yuster, Zwick; 1997:
- $O(n^\omega)$-time method to *count* the number of triangles in $G$.
  \[ \Rightarrow O(n^\omega) \leq o(n^{2.373}) \]
- $O(m^{\frac{2\omega}{\omega + 1}})$-time matrix multiplication-based method to *determine* if $G$ contains a triangle, and if so, *output* a triangle.
  \[ \Rightarrow O(m^{\frac{2\omega}{\omega + 1}}) \leq o(m^{1.408}) \]

$\omega < 2.373$ is a famous result from STOC 2012 by V. Vassilevska Williams.

Our paper:

**Theorem 1**

We can *count* the number of triangles in $G$ in $O(m^{\frac{2\omega}{\omega + 1}})$ time.

\[ \Rightarrow O(m^{\frac{2\omega}{\omega + 1}}) \leq o(m^{1.408}) \]

(Better than the above when $m \ll n^2$.)
Theorem 1

We can count the number of triangles in $G$ in $O(m^{\frac{2\omega}{\omega+1}})$ time.

Proof:

1. Count the number of triangles whose three vertices have degree $\geq t$:
   - Apply the triangle counting method of Alon et al. (1997) to the subgraph of $G$ induced by all vertices of degree $\geq t$.
   - The number of vertices in $G$ with degree $\geq t$ is $O(m^\omega)$.
   - This takes $O(m^\omega t^\omega) = O(m^{\frac{2\omega}{\omega+1}})$ time.

2. Count the number of triangles with at least one vertex of degree $< t$:
   - Enumerate the set $F$ of edges with at least one endpoint of degree $< t$.
   - For $i = 1, \ldots, |F|$, do:
     - Pick an endpoint $v$ of edge $e_i$ in $F$ with degree $< t$;
     - for each edge $e$ incident to $e_i$ at $v$, check if $e_i$ and $e$ induce a triangle in $G$ that doesn't include any edge $e_j \in F$ with $j < i$.
   - Counting such triangles takes $O(mt) = O(m^{1+\omega-1}) = O(m^{\frac{2\omega}{\omega+1}})$ time.
Theorem 1

We can count the number of triangles in $G$ in $O(m^{\frac{2\omega}{\omega+1}})$ time.

Proof: Let $t = m^{\frac{\omega-1}{\omega+1}}$.

1. **Count the number of triangles whose three vertices have degree $\geq t$:**
   
   Apply the triangle counting method of Alon *et al.* (1997) to the subgraph of $G$ induced by all vertices of degree $\geq t$.
   
   The number of vertices in $G$ with degree $\geq t$ is $O(m^{t})$.
   
   ⇒ This takes $O(m^t) = O(m^{\frac{\omega}{\omega+1}}) = O(m^{\frac{2\omega}{\omega+1}})$ time.
Theorem 1

We can count the number of triangles in $G$ in $O(m^{\frac{2\omega}{\omega+1}})$ time.

Proof: Let $t = m^{\frac{\omega-1}{\omega+1}}$.

1. **Count the number of triangles whose three vertices have degree $\geq t$:**
   Apply the triangle counting method of Alon et al. (1997) to the subgraph of $G$ induced by all vertices of degree $\geq t$.
   The number of vertices in $G$ with degree $\geq t$ is $O\left(\frac{m}{t}\right)$.
   $\Rightarrow$ This takes $O\left(\frac{m^\omega}{t^\omega}\right) = O\left(m^\omega - \frac{(\omega-1)\omega}{\omega+1}\right) = O\left(m^{\frac{2\omega}{\omega+1}}\right)$ time.

2. **Count the number of triangles with at least one vertex of degree $< t$:**
   Enumerate the set $F$ of edges with at least one endpoint of degree $< t$.
   For $i = 1, \ldots, |F|$, do: Pick an endpoint $v$ of edge $e_i$ in $F$ with degree $< t$; for each edge $e$ incident to $e_i$ at $v$, check if $e_i$ and $e$ induce a triangle in $G$ that doesn’t include any edge $e_j \in F$ with $j < i$.
   $\Rightarrow$ Counting such triangles takes $O(mt) = O\left(m^{1+\frac{\omega-1}{\omega+1}}\right) = O\left(m^{\frac{2\omega}{\omega+1}}\right)$ time. □
Counting monochromatic triangles

Now, suppose the edges of the given graph are colored.

Definition

A monochromatic triangle in an undirected, edge-colored graph is a cycle of length 3 whose edges have the same color.
Counting monochromatic triangles

Now, suppose the edges of the given graph are colored.

**Definition**

A *monochromatic triangle* in an undirected, edge-colored graph is a cycle of length 3 whose edges have the same color.

**Example:**

This graph contains one monochromatic triangle (*blue – blue – blue*).
Counting monochromatic triangles

Now, suppose the edges of the given graph are colored.

**Definition**

A monochromatic triangle in an undirected, edge-colored graph is a cycle of length 3 whose edges have the same color.

Let $G$ be an undirected, connected, edge-colored graph with $m$ edges and $n$ vertices.

**Vassilevska, Williams, Yuster; 2010:**

- $O(n^{\frac{3+\omega}{2}})$-time method to determine if $G$ contains a monochromatic triangle, and if so, output such a triangle. $\Rightarrow O(n^{\frac{3+\omega}{2}}) \leq o(n^{2.687})$
Counting monochromatic triangles

Now, suppose the edges of the given graph are colored.

**Definition**

A monochromatic triangle in an undirected, edge-colored graph is a cycle of length 3 whose edges have the same color.

Let $G$ be an undirected, connected, edge-colored graph with $m$ edges and $n$ vertices.

**Vassilevska, Williams, Yuster; 2010:**

- $O(n^{\frac{3+\omega}{2}})$-time method to determine if $G$ contains a monochromatic triangle, and if so, output such a triangle. $\Rightarrow O(n^{\frac{3+\omega}{2}}) \leq o(n^{2.687})$

**Our paper:**

**Theorem 2**

We can count the number of monochromatic triangles in $G$ with colors belonging to any specified set $C$ in $O(n^{\frac{3+\omega}{2}})$ time.

$\Rightarrow O(n^{\frac{3+\omega}{2}}) \leq o(n^{2.687})$
Theorem 2

We can count the number of monochromatic triangles in \( G \) with colors belonging to any specified set \( C \) in \( O(n^{3+\omega}/2) \) time.

Proof:
Theorem 2

We can *count* the number of monochromatic triangles in $G$ with colors belonging to any specified set $C$ in $O(n^{3+\omega}/2)$ time.

**Proof:** For each color $i \in C$, let $E_i$ be the set of edges in $G$ colored by $i$. Color $i$ is said to be **heavily used** if $|E_i| \geq n^{(\omega+1)/2}$.
Theorem 2

We can *count* the number of monochromatic triangles in $G$ with colors belonging to any specified set $C$ in $O(n^{3+\omega})$ time.

**Proof:** For each color $i \in C$, let $E_i$ be the set of edges in $G$ colored by $i$. Color $i$ is said to be heavily used if $|E_i| \geq n^{(\omega+1)/2}$.

1. For each heavily used color in $C$, count the number of monochromatic triangles with this color by applying the $O(n^\omega)$-time triangle counting method of Alon *et al.* (1997) to the subgraph induced by edges colored by $i$. 

$$\Rightarrow O(n^2/n^{(\omega+1)/2}) \cdot O(n^\omega) = O(n^{2-\omega+1/2} + \omega) = O(n^{3+\omega}/2) \text{ time in total.}$$
Theorem 2

We can count the number of monochromatic triangles in $G$ with colors belonging to any specified set $C$ in $O(n^{3+\omega}/2)$ time.

Proof: For each color $i \in C$, let $E_i$ be the set of edges in $G$ colored by $i$. Color $i$ is said to be heavily used if $|E_i| \geq n^{(\omega+1)/2}$.

1. For each heavily used color in $C$, count the number of monochromatic triangles with this color by applying the $O(n^\omega)$-time triangle counting method of Alon et al. (1997) to the subgraph induced by edges colored by $i$. 
   \[ O(n^2/n^{(\omega+1)/2}) \cdot O(n^\omega) = O(n^{2-\omega+1/2+\omega}) = O(n^{3+\omega}/2) \] time in total.

2. For each non-heavily used color $i \in C$, apply the counting method of Theorem 1 to the subgraph induced by $E_i$. 
   \[ O(|E_i|^{2\omega}/(\omega+1)) \] time for color $i$. 
   The total time taken by non-heavily used colors $i \in C$ is maximized if $|E_i| = \Theta(n^{\omega+1/2})$ holds for all of them, and there are $\Theta(n^{2-\omega+1/2})$ of them. 
   \[ \Rightarrow \] The total time to count monochromatic triangles for the non-heavily used colors in $C$ is $O(n^{2-\omega+1/2}) \cdot O((n^{\omega+1/2})^{2\omega}/(\omega+1)) = O(n^{3+\omega}/2)$. \[\square\]
Next, consider a relaxation of the concept of a monochromatic triangle:

**Definition**

Let $R$ be a binary relation on the edge colors. A triangle in $G$ with two edges of the same color $i$ and the third one of color $k$ such that $iRk$ holds is called an $R$-chromatic triangle.
Next, consider a relaxation of the concept of a monochromatic triangle:

**Definition**

Let $R$ be a binary relation on the edge colors. A triangle in $G$ with two edges of the same color $i$ and the third one of color $k$ such that $i R k$ holds is called an $R$-chromatic triangle.

**Example:** If $R$ stands for $<$ then $k$ is required to be larger than $i$. 
Next, consider a relaxation of the concept of a monochromatic triangle:

**Definition**

Let $R$ be a binary relation on the edge colors. A triangle in $G$ with two edges of the same color $i$ and the third one of color $k$ such that $iRk$ holds is called an $R$-chromatic triangle.

**Example:** If $R$ stands for $<$ then $k$ is required to be larger than $i$. Suppose blue $<$ red $<$ green.

This graph contains two $R$-chromatic triangles (blue – blue – red).
Next, consider a relaxation of the concept of a monochromatic triangle:

**Definition**

Let $R$ be a binary relation on the edge colors. A triangle in $G$ with two edges of the same color $i$ and the third one of color $k$ such that $iRk$ holds is called an $R$-chromatic triangle.

**Example:** If $R$ stands for $<$ then $k$ is required to be larger than $i$.

Modify the previous techniques to obtain:

**Theorem 3**

Let $R$ be a binary relation on the edge colors of $G$ computable in constant time. The number of $R$-chromatic triangles in $G$ can be computed in $O(n^{\frac{3+\omega}{2}})$ time.

$$O(n^{\frac{3+\omega}{2}}) \leq o(n^{2.687})$$
Let $G$ be an undirected, connected, edge-colored graph with $m$ edges and $n$ vertices.

**Theorem 2**

We can *count* the number of monochromatic triangles in $G$ with colors belonging to any specified set $C$ in $O(n^{3+\omega}/2)$ total time.

**Definition**

Let $R$ be a binary relation on the edge colors. A triangle in $G$ with two edges of the same color $i$ and the third one of color $k$ such that $iRk$ holds is called an $R$-chromatic triangle.

**Theorem 3**

Let $R$ be a binary relation on the edge colors of $G$ computable in constant time. The number of $R$-chromatic triangles in $G$ can be computed in $O(n^{3+\omega}/2)$ time.
PART II : Phylogenetic trees and networks
We will apply the triangle counting technique to the problem of measuring the structural dissimilarity between two *phylogenetic networks*. 
We will apply the triangle counting technique to the problem of measuring the structural dissimilarity between two phylogenetic networks.

First, some background and definitions.
**Definition**

A **phylogenetic tree** is an unordered tree whose leaves are uniquely labeled.

<table>
<thead>
<tr>
<th>Arachnida</th>
<th>Mammalia</th>
<th>Reptilia</th>
<th>Aves</th>
<th>Amphibia</th>
</tr>
</thead>
</table>
A phylogenetic tree is an unordered tree whose leaves are uniquely labeled. Can describe divergent evolutionary history for a set of objects, where: "objects" = Biological species, categories of species, populations, proteins, nucleic acids, natural languages, chain letters, medieval manuscripts, or ...
Phylogenetic trees

**Definition**

A **phylogenetic tree** is an unordered tree whose leaves are uniquely labeled.

Can describe divergent evolutionary history for a set of objects, where:

“objects” = Biological species, categories of species, populations, proteins, nucleic acids, natural languages, chain letters, medieval manuscripts, or ...

Main idea:

- Represent objects by *leaves* in the tree.
- Select branching structure so that *internal nodes* correspond to common ancestors.
Phylogenetic tree, example

(Figure from http://biology.unm.edu/ccouncil/Biology_203)
Phylogenetic supertree, example

Phylogenetic supertree, example

The phylogenetic tree model has been used by biologists for a long time to describe evolutionary history.

But sometimes, the objects fail to fit into the phylogenetic tree model.

- Horizontal gene transfer
- Hybrid speciation

“Reticulation events”
Phylogenetic networks

The phylogenetic tree model has been used by biologists for a long time to describe evolutionary history.

But sometimes, the objects fail to fit into the phylogenetic tree model.

- Horizontal gene transfer
- Hybrid speciation

“Reticulation events”

**Phylogenetic network:** Generalization of phylogenetic tree in which internal nodes may have more than one parent.
Phylogenetic networks

The phylogenetic tree model has been used by biologists for a long time to describe evolutionary history.

But sometimes, the objects fail to fit into the phylogenetic tree model.

- Horizontal gene transfer
- Hybrid speciation

“Reticulation events”

**Phylogenetic network:** Generalization of phylogenetic tree in which internal nodes may have more than one parent.

(Thus, the phylogenetic network model extends the traditional phylogenetic tree model.)
Phylogenetic network, example

Phylogenetic network, definition

Definition

**Phylogenetic network** = A directed acyclic graph with a single root vertex and distinctly labeled leaves, and no vertices having both indegree 1 and outdegree 1.
**Definition**

**Phylogenetic network** = A directed acyclic graph with a single root vertex and distinctly labeled leaves, and no vertices having both indegree 1 and outdegree 1.

Bioinformaticians may need to measure the *dissimilarity* between phylogenetic networks:

Intuitively, this measure considers two phylogenetic networks that share many small embedded subtrees to be similar.
**Definition**

**Phylogenetic network** = A directed acyclic graph with a single root vertex and distinctly labeled leaves, and no vertices having both indegree 1 and outdegree 1.

Bioinformaticians may need to measure the **dissimilarity** between phylogenetic networks:
- To identify parts of them that look alike.
Phylogenetic network, definition

**Definition**

**Phylogenetic network** = A directed acyclic graph with a single root vertex and distinctly labeled leaves, and no vertices having both indegree 1 and outdegree 1.

Bioinformaticians may need to measure the **dissimilarity** between phylogenetic networks:

- To identify parts of them that look alike.
- To evaluate the performance of a new tree/network inference method.
**Phylogenetic network, definition**

**Definition**

**Phylogenetic network** = A directed acyclic graph with a single root vertex and distinctly labeled leaves, and no vertices having both indegree 1 and outdegree 1.

Bioinformaticians may need to measure the **dissimilarity** between phylogenetic networks:

- To identify parts of them that look alike.
- To evaluate the performance of a new tree/network inference method.
- To support queries in a phylogenetic database.
Phylogenetic network, definition

**Definition**

**Phylogenetic network** = A directed acyclic graph with a single root vertex and distinctly labeled leaves, and no vertices having both indegree 1 and outdegree 1.

Bioinformaticians may need to measure the **dissimilarity** between phylogenetic networks:

- To identify parts of them that look alike.
- To evaluate the performance of a new tree/network inference method.
- To support queries in a phylogenetic database.

Several measures of dissimilarity have been proposed in the literature:
Definition

Phylogenetic network = A directed acyclic graph with a single root vertex and distinctly labeled leaves, and no vertices having both indegree 1 and outdegree 1.

Bioinformaticians may need to measure the dissimilarity between phylogenetic networks:

- To identify parts of them that look alike.
- To evaluate the performance of a new tree/network inference method.
- To support queries in a phylogenetic database.

Several measures of dissimilarity have been proposed in the literature:

- the Robinson-Foulds distance, the tripartitions distance, the $\mu$-distance, the nodal distance, the split nodal distance, the Nakhleh distance, etc.
Phylogenetic network, definition

**Definition**

**Phylogenetic network** = A directed acyclic graph with a single root vertex and distinctly labeled leaves, and no vertices having both indegree 1 and outdegree 1.

Bioinformaticians may need to measure the **dissimilarity** between phylogenetic networks:

- To identify parts of them that look alike.
- To evaluate the performance of a new tree/network inference method.
- To support queries in a phylogenetic database.

Several measures of dissimilarity have been proposed in the literature:

- the Robinson-Foulds distance, the tripartitions distance, the $\mu$-distance, the nodal distance, the split nodal distance, the Nakhleh distance, etc.

Here, we focus on how to compute the **rooted triplet distance**. Intuitively, this measure considers two phylogenetic networks that share many small embedded subtrees to be similar.
A rooted phylogenetic tree with exactly three leaves is called a **rooted triplet**.
A rooted phylogenetic tree with exactly three leaves is called a rooted triplet.

Let \( \{a, b, c\} \) be a leaf label set of cardinality 3.
There are four different possible trees leaf-labeled by \( \{a, b, c\} \), denoted as follows:

\[
\begin{align*}
\text{Rooted fan triplet} & = \text{One internal node} & (a \mid b \mid c) \\
\text{Rooted binary triplet} & = \text{Two internal nodes} & (a b \mid c, a c \mid b, b c \mid a)
\end{align*}
\]
A rooted phylogenetic tree with exactly three leaves is called a **rooted triplet**.

Let \( \{a, b, c\} \) be a leaf label set of cardinality 3. There are four different possible trees leaf-labeled by \( \{a, b, c\} \), denoted as follows:

Two types of rooted triplets:

- **Rooted fan triplet** = One internal node \((a|b|c)\)
- **Rooted binary triplet** = Two internal nodes \((ab|c), (ac|b), \text{ and } bc|a)\)
Let $U$ be a phylogenetic network and let $a, b, c$ be three leaf labels in $U$.

**Definition**

- The rooted fan triplet $a|b|c$ is **consistent with** $U$ iff $U$ contains a vertex $w$ such that there exist three directed paths $w \rightarrow a$, $w \rightarrow b$, and $w \rightarrow c$ that are vertex-disjoint except for in the common start vertex $w$. 

- The rooted binary triplet $ab|c$ is consistent with $U$ iff $U$ contains vertices $w$ and $z$ such that there exist four directed paths $w \rightarrow a$, $w \rightarrow b$, $z \rightarrow w$, and $z \rightarrow c$ of non-zero length that are vertex-disjoint except for in the vertices $w$ and $z$.
Let $U$ be a phylogenetic network and let $a, b, c$ be three leaf labels in $U$.

**Definition**

- The rooted fan triplet $a|b|c$ is **consistent with** $U$ iff $U$ contains a vertex $w$ such that there exist three directed paths $w \rightarrow a$, $w \rightarrow b$, and $w \rightarrow c$ that are vertex-disjoint except for in the common start vertex $w$.

- The rooted binary triplet $ab|c$ is **consistent with** $U$ iff $U$ contains vertices $w$ and $z$ such that there exist four directed paths $w \rightarrow a$, $w \rightarrow b$, $z \rightarrow w$, and $z \rightarrow c$ of non-zero length that are vertex-disjoint except for in the vertices $w$ and $z$. 

Remark: The top network is consistent with $a|b|c$, $bc|a$, and the bottom network is consistent with $ab|c$, $bc|a$. 

2015-07-27
Rooted triplet consistency

Let $U$ be a phylogenetic network and let $a, b, c$ be three leaf labels in $U$.

**Definition**

- The rooted fan triplet $a|b|c$ is **consistent with $U$** iff $U$ contains a vertex $w$ such that there exist three directed paths $w \rightarrow a$, $w \rightarrow b$, and $w \rightarrow c$ that are vertex-disjoint except for in the common start vertex $w$.

- The rooted binary triplet $ab|c$ is **consistent with $U$** iff $U$ contains vertices $w$ and $z$ such that there exist four directed paths $w \rightarrow a$, $w \rightarrow b$, $z \rightarrow w$, and $z \rightarrow c$ of non-zero length that are vertex-disjoint except for in the vertices $w$ and $z$.

**Remark:** The top network is consistent with $a|b|c$, $bc|a$, and the bottom network is consistent with $ab|c$, $bc|a$. 
The rooted triplet distance

**Definition [Gambette & Huber; 2012]**

Let $U_1$, $U_2$ be two phylogenetic networks on the same leaf label set $L$. The *rooted triplet distance* between $U_1$ and $U_2$, denoted by $d_{rt}(U_1, U_2)$, is the number of rooted fan triplets and rooted binary triplets that are consistent with exactly one of $U_1$ and $U_2$. 
The rooted triplet distance

Definition [Gambette & Huber; 2012]

Let $U_1$, $U_2$ be two phylogenetic networks on the same leaf label set $L$. The rooted triplet distance between $U_1$ and $U_2$, denoted by $d_{rt}(U_1, U_2)$, is the number of rooted fan triplets and rooted binary triplets that are consistent with exactly one of $U_1$ and $U_2$.

Example:

$U_1$ is consistent with: $bc|a$, $ab|d$, $bd|a$, $ac|d$, $cd|a$, $bc|d$

$U_2$ is consistent with: $bc|a$, $bd|a$, $cd|a$, $b|c|d$
The rooted triplet distance

**Definition** [Gambette & Huber; 2012]

Let $U_1$, $U_2$ be two phylogenetic networks on the same leaf label set $L$. The **rooted triplet distance** between $U_1$ and $U_2$, denoted by $d_{rt}(U_1, U_2)$, is the number of rooted fan triplets and rooted binary triplets that are consistent with exactly one of $U_1$ and $U_2$.

**Example:**

$U_1$ is consistent with: $bc|a$, $ab|d$, $bd|a$, $ac|d$, $cd|a$, $bc|d$

$U_2$ is consistent with: $bc|a$, $bd|a$, $cd|a$, $bc|d$

$\Rightarrow d_{rt}(U_1, U_2) = 4$
The rooted triplet distance

**Definition [Gambette & Huber; 2012]**

Let $U_1$, $U_2$ be two phylogenetic networks on the same leaf label set $L$. The **rooted triplet distance between $U_1$ and $U_2$**, denoted by $d_{rt}(U_1, U_2)$, is the number of rooted fan triplets and rooted binary triplets that are consistent with exactly one of $U_1$ and $U_2$.

The rooted triplet distance between two phylogenetic trees was introduced by Dobson in 1975. Denote $n = |L|$. It is known that then $d_{rt}$ can be computed in:

- $O(n^3)$ time by a naive algorithm.
- $O(n^2)$ time [Bansal et al.; 2011].
- $O(n \log n)$ time [Brodal et al.; 2013].
- $O(n \log^3 n)$ time with a more “practical” algorithm [Jansson & Rajaby; 2015].
The rooted triplet distance

Definition [Gambette & Huber; 2012]

Let $U_1$, $U_2$ be two phylogenetic networks on the same leaf label set $L$. The rooted triplet distance between $U_1$ and $U_2$, denoted by $d_{rt}(U_1, U_2)$, is the number of rooted fan triplets and rooted binary triplets that are consistent with exactly one of $U_1$ and $U_2$.

The rooted triplet distance between two phylogenetic trees was introduced by Dobson in 1975. Denote $n = |L|$. It is known that then $d_{rt}$ can be computed in:

- $O(n^3)$ time by a naive algorithm.
- $O(n^2)$ time [Bansal et al.; 2011].
- $O(n \log n)$ time [Brodal et al.; 2013].
- $O(n \log^3 n)$ time with a more “practical” algorithm [Jansson & Rajaby; 2015].

For phylogenetic networks, it seems harder to get an $o(n^3)$-time algorithm.

Our paper: Subcubic time for a special type of phylogenetic networks called galled trees.
Phylogenetic network = A directed acyclic graph with a single root vertex and distinctly labeled leaves, and no vertices having both indegree 1 and outdegree 1.

An important special class of phylogenetic networks is the galled tree:

Definition [Ma et al.; 1998 & Gusfield et al.; 2003]
Let $U$ be a phylogenetic network. $U$ is a galled tree if all cycles in its underlying undirected graph are vertex-disjoint.
Galled tree, definition

**Definition**

*Phylogenetic network* = A directed acyclic graph with a single root vertex and distinctly labeled leaves, and no vertices having both indegree 1 and outdegree 1.

An important special class of phylogenetic networks is the *galled tree*:

**Definition [Ma et al.; 1998 & Gusfield et al.; 2003]**

Let $U$ be a phylogenetic network. $U$ is a *galled tree* if all cycles in its underlying undirected graph are vertex-disjoint.

- Cycles are allowed in a galled tree, but all such cycles have to be disjoint.  
  ⇒ Simple structure, almost like a tree.
Phylogenetic network = A directed acyclic graph with a single root vertex and distinctly labeled leaves, and no vertices having both indegree 1 and outdegree 1.

An important special class of phylogenetic networks is the *galled tree*:

Definition [Ma et al.; 1998 & Gusfield et al.; 2003]

Let $U$ be a phylogenetic network. $U$ is a galled tree if all cycles in its underlying undirected graph are vertex-disjoint.

- Cycles are allowed in a galled tree, but all such cycles have to be disjoint.  
  $\Rightarrow$ Simple structure, almost like a tree.

- Some problems that are $\mathcal{NP}$-hard for unrestricted phylogenetic networks have been shown to be polynomial-time solvable for galled trees.
**Galled tree, definition**

**Definition**

**Phylogenetic network** = A directed acyclic graph with a single root vertex and distinctly labeled leaves, and no vertices having both indegree 1 and outdegree 1.

---

An important special class of phylogenetic networks is the *galled tree*:

**Definition [Ma et al.; 1998 & Gusfield et al.; 2003]**

Let $U$ be a phylogenetic network. $U$ is a **galled tree** if all cycles in its underlying undirected graph are vertex-disjoint.

---

- Cycles are allowed in a galled tree, but all such cycles have to be disjoint.  
  ⇒ Simple structure, almost like a tree.

- Some problems that are $\mathcal{NP}$-hard for unrestricted phylogenetic networks have been shown to be **polynomial-time solvable** for galled trees.

- When the number of reticulation events is small and most of them have occurred recently, a galled tree may suffice to accurately describe the underlying evolutionary branching pattern [Gusfield et al.; 2003].
PART II, summary

Definition [Gambette & Huber; 2012]

Let $U_1$, $U_2$ be two phylogenetic networks on the same leaf label set $L$. The rooted triplet distance between $U_1$ and $U_2$, denoted by $d_{rt}(U_1, U_2)$, is the number of rooted fan triplets and rooted binary triplets that are consistent with exactly one of $U_1$ and $U_2$.

Definition

Let $U$ be a phylogenetic network. $U$ is a galled tree if all cycles in its underlying undirected graph are vertex-disjoint.
PART III : Computing the rooted triplet distance between galled trees by counting triangles
Common ancestors

Let $U$ be a phylogenetic network and let $a, c, w$ be vertices in $U$.

- $w$ is an **ancestor** of $a$ iff there is a directed path $P$ from $w$ to $a$. Equivalently, $a$ is a **descendant** of $w$. (If $|P| > 0$ then “proper ancestor/descendant”.)

**Example:**

![Diagram showing common ancestors](image-url)
Let $U$ be a phylogenetic network and let $a, c, w$ be vertices in $U$.

- $w$ is an ancestor of $a$ iff there is a directed path $P$ from $w$ to $a$. Equivalently, $a$ is a descendant of $w$. (If $|P| > 0$ then “proper ancestor/descendant”.)

- $w$ is a junction common ancestor (jca) of $a$ and $c$ iff there exist two directed paths from $w$ to $a$ and from $w$ to $c$, respectively, which are vertex-disjoint except for in the start vertex $w$.

**Example:**

$w$ is a jca of $a$ and $c$.

$z$ is a jca of $a$ and $c$.
Common ancestors

Let $U$ be a phylogenetic network and let $a, c, w$ be vertices in $U$.

- $w$ is an ancestor of $a$ iff there is a directed path $P$ from $w$ to $a$. Equivalently, $a$ is a descendant of $w$. (If $|P| > 0$ then “proper ancestor/descendant”.)

- $w$ is a junction common ancestor (jca) of $a$ and $c$ iff there exist two directed paths from $w$ to $a$ and from $w$ to $c$, respectively, which are vertex-disjoint except for in the start vertex $w$.

- $w$ is a lowest common ancestor (lca) of $a$ and $c$ iff: (i) $w$ is an ancestor of both $a$ and $c$; and (ii) $w$ has no proper descendant that is an ancestor of both $a$ and $c$.

Example:

- $w$ is a jca of $a$ and $c$.
- $z$ is a jca of $a$ and $c$.
- $w$ is an lca of $a$ and $c$.
- $z$ is not an lca of $a$ and $c$. 
Connection: triangle counting ↔ rooted triplet distance

Observation for rooted fan triplets:

\[ T_1 : \begin{array}{c} p \\ q \end{array} \begin{array}{c} c \\ a \end{array} \begin{array}{c} d \\ b \end{array}, \quad T_2 : \begin{array}{c} r \\ a \end{array} \begin{array}{c} c \\ b \end{array} \begin{array}{c} s \\ d \end{array} \]

\[(a|c|d, b|c|d) \text{ in } T_1 \text{ and } (a|b|c, a|c|d) \text{ in } T_2\]
Observation for rooted fan triplets:

\[
T_1 : \begin{array}{c}
| \quad p \\
q & c & d \\
a & b \\
\end{array}
\]

\[
T_2 : \begin{array}{c}
| \quad r \\
| a & c \\
| b & d \\
| s \\
\end{array}
\]

\((a|c|d, b|c|d \text{ in } T_1 \text{ and } a|b|c, a|c|d \text{ in } T_2)\)

\[x|y|z\text{ is consistent with the tree } T_1 \text{ iff } \text{lca}(x,y) = \text{lca}(x,z) = \text{lca}(y,z) \text{ in } T_1.\]

The same holds for \( T_2.\)
Connection: triangle counting $\leftrightarrow$ rooted triplet distance

Observation for rooted fan triplets:

$T_1: \quad \begin{array}{c} \quad p \\ \quad q \\ a \\ b \end{array}$

$T_2: \quad \begin{array}{c} \quad r \\ \quad s \\ a \\ b \end{array}$

$(a|c|d, b|c|d$ in $T_1$ and $a|b|c, a|c|d$ in $T_2$)

$x|y|z$ is consistent with the tree $T_1$ iff $\text{lca}(x,y) = \text{lca}(x,z) = \text{lca}(y,z)$ in $T_1$. The same holds for $T_2$.

If we create an undirected complete graph $G = (L, E)$ in which every edge is assigned a “color” of the form $(\nu_1, \nu_2)$, where $\nu_1 \in T_1$ and $\nu_2 \in T_2$, as follows:

- For each edge $\{u, \nu\} \in E$, color it with the color $(\text{lca}^{T_1}(u, \nu), \text{lca}^{T_2}(u, \nu))$.

then:
Observation for rooted fan triplets:

\[ T_1 : p \quad q \quad c \quad d \quad b \quad a \]

\[ T_2 : r \quad a \quad c \quad s \quad b \quad d \]

\((a|c|d, b|c|d) \text{ in } T_1 \text{ and } (a|b|c, a|c|d) \text{ in } T_2\)

\(x|y|z\) is consistent with the tree \(T_1\) iff \(\lca(x,y) = \lca(x,z) = \lca(y,z)\) in \(T_1\).

The same holds for \(T_2\).

If we create an undirected complete graph \(G = (L, E)\) in which every edge is assigned a “color” of the form \((v_1, v_2)\), where \(v_1 \in T_1\) and \(v_2 \in T_2\), as follows:

- For each edge \(\{u, v\} \in E\), color it with the color \((\lca^{T_1}(u, v), \lca^{T_2}(u, v))\).

then:

\(x|y|z\) is consistent with both \(T_1\) and \(T_2\) iff all three edges \(\{x, y\}, \{x, z\}, \{y, z\}\) have the same color in \(G\), so the number of rooted fan triplets common to both trees equals the number of monochromatic triangles in \(G\).
Connection: triangle counting $\leftrightarrow$ rooted triplet distance

Observation for rooted fan triplets:

(a|c|d, b|c|d in $T_1$ and $a|b|c$, $a|c|d$ in $T_2$)

$x|y|z$ is consistent with the tree $T_1$ iff $\text{lca}(x,y) = \text{lca}(x,z) = \text{lca}(y,z)$ in $T_1$. The same holds for $T_2$.

If we create an undirected complete graph $\mathcal{G} = (L, E)$ in which every edge is assigned a “color” of the form $(v_1, v_2)$, where $v_1 \in T_1$ and $v_2 \in T_2$, as follows:

- For each edge $\{u, v\} \in E$, color it with the color $(\text{lca}^{T_1}(u, v), \text{lca}^{T_2}(u, v))$.

then:

$x|y|z$ is consistent with both $T_1$ and $T_2$ iff all three edges $\{x, y\}$, $\{x, z\}$, $\{y, z\}$ have the same color in $\mathcal{G}$, so the number of rooted fan triplets common to both trees equals the number of monochromatic triangles in $\mathcal{G}$. 

\[2015-07-27\]
Apply Theorem 2 for counting monochromatic triangles.
⇒ $O(n^{3+\omega}/2)$ time
Apply Theorem 2 for counting monochromatic triangles.
⇒ $O(n^{(3+\omega)/2})$ time

**Lemma 6**

Let $T_1$, $T_2$ be two phylogenetic trees on the same set $L$ of $n$ leaves. The number of rooted fan triplets consistent with both $T_1$ and $T_2$ can be computed in $O(n^{3+\omega}/2) \leq o(n^{2.687})$ time.
Apply Theorem 2 for counting monochromatic triangles.
⇒ $O(n^{(3+\omega)/2})$ time

Lemma 6
Let $T_1$, $T_2$ be two phylogenetic trees on the same set $L$ of $n$ leaves. The number of rooted fan triplets consistent with both $T_1$ and $T_2$ can be computed in $O(n^{3+\omega/2}) \leq o(n^{2.687})$ time.

Next, we refine this technique to the more complicated case of galled trees.

(We have to modify the definition of the edge colors slightly and count $R$-chromatic triangles for a suitably defined binary relation $R$, and apply Theorem 3 instead.)
The case of galled trees

There are essentially three different ways that a rooted fan triplet \( a|b|c \) can be consistent with a galled tree \( U \):

\[ (1) \quad (2) \quad (3) \]

Let \( U \downarrow \) be the tree obtained by removing one reticulation edge from each cycle. Define \( U \uparrow \) in the same way, but removing all the other reticulation edges instead. In (1) & (2), \( a|b|c \) is consistent with one of the two trees \( U \downarrow \) and \( U \uparrow \) ("type 1"). In (3), \( a|b|c \) is consistent with both trees \( U \downarrow \) and \( U \uparrow \) ("type 2"). In (1) & (3), \( \text{lca}_U(a, b) = \text{lca}_U(a, c) = \text{lca}_U(b, c) \) but this is not true in (2).
The case of galled trees

There are essentially three different ways that a rooted fan triplet $a|b|c$ can be consistent with a galled tree $U$:

Let $U \downarrow$ be the tree obtained by removing one reticulation edge from each cycle. Define $U \uparrow$ in the same way, but removing all the other reticulation edges instead.
The case of galled trees

There are essentially three different ways that a rooted fan triplet \( a | b | c \) can be consistent with a galled tree \( U \):

1. Let \( U \downarrow \) be the tree obtained by removing one reticulation edge from each cycle. Define \( U \uparrow \) in the same way, but removing all the other reticulation edges instead.

2. In (1) & (2), \( a | b | c \) is consistent with one of the two trees \( U \downarrow \) and \( U \uparrow \) ("type 1").

3. In (3), \( a | b | c \) is consistent with both trees \( U \downarrow \) and \( U \uparrow \) ("type 2").
The case of galled trees

There are essentially three different ways that a rooted fan triplet $a|b|c$ can be consistent with a galled tree $U$:

Let $U \searrow$ be the tree obtained by removing one reticulation edge from each cycle. Define $U \swarrow$ in the same way, but removing all the other reticulation edges instead.

In (1) & (2), $a|b|c$ is consistent with one of the two trees $U \searrow$ and $U \swarrow$ ("type 1").

In (3), $a|b|c$ is consistent with both trees $U \searrow$ and $U \swarrow$ ("type 2").

In (1) & (3), $\text{lca}^U(a, b) = \text{lca}^U(a, c) = \text{lca}^U(b, c)$ but this is not true in (2).
Characterization of rooted fan triplet consistency

Technical lemma:

Lemma 5

Let $a$, $b$, and $c$ be three leaf labels in a galled tree $U$. It holds that:

1. The rooted fan triplet $a|b|c$ is consistent with $U$ and $a|b|c$ is of type 1 in $U$ if and only if either:

   (i) $lca^U(a, b) = lca^U(a, c) = lca^U(b, c) = w$ for some vertex $w$, $w$ is equal to all of $lca(a, b)$, $lca(a, c)$, and $lca(b, c)$ in one of $U\searrow$ and $U\nearrow$, and $w$ is equal to exactly one of $lca(a, b)$, $lca(a, c)$, and $lca(b, c)$ in the other; or

   (ii) For two $(x, y)$ among $\{(a, b), (a, c), (b, c)\}$, it holds that $lca^U(x, y) = lca^U\searrow(x, y) = lca^U\nearrow(x, y) = w$ for some vertex $w$, and for the third pair $(x, y)$, there exists a proper descendant $v$ of $w$ in $U$ such that $v = lca^U(x, y)$ and $v = lca(x, y)$ in exactly one of $U\searrow$ and $U\nearrow$ while $w = lca(x, y)$ in the other.

2. The rooted fan triplet $a|b|c$ is consistent with $U$ and $a|b|c$ is of type 2 in $U$ if and only if $lca^U\searrow(a, b) = lca^U\nearrow(a, b) = lca^U\searrow(a, c) = lca^U\nearrow(a, c) = lca^U\searrow(b, c) = lca^U\nearrow(b, c)$. 
Counting the shared rooted fan triplets

For $p, q \in \{1, 2\}$, let $T_{p,q} =$ the number of rooted fan triplets consistent with both $U_1$ and $U_2$ that are of type $p$ in $U_1$ and of type $q$ in $U_2$.

**Goal:** Compute the sum $T_{1,1} + T_{1,2} + T_{2,1} + T_{2,2}$ (i.e., the number of rooted fan triplets consistent with both $U_1$ and $U_2$).
Counting the shared rooted fan triplets

For $p, q \in \{1, 2\}$, let $T_{p,q}$ = the number of rooted fan triplets consistent with both $U_1$ and $U_2$ that are of type $p$ in $U_1$ and of type $q$ in $U_2$.

**Goal:** Compute the sum $T_{1,1} + T_{1,2} + T_{2,1} + T_{2,2}$ (i.e., the number of rooted fan triplets consistent with both $U_1$ and $U_2$).

- $T_{1,1} + 2T_{1,2} + 2T_{2,1} + 4T_{2,2}$ can be computed in $O(n^{(3+\omega)/2})$ time.
  (Use Lemma 6 to compute the number of rooted fan triplets shared between $U_1^\downarrow$ and $U_2^\downarrow$, between $U_1^\downarrow$ and $U_2^\uparrow$, between $U_1^\uparrow$ and $U_2^\downarrow$, and between $U_1^\uparrow$ and $U_2^\uparrow$, and sum them up.)
Counting the shared rooted fan triplets

For $p, q \in \{1, 2\}$, let $T_{p,q}$ = the number of rooted fan triplets consistent with both $U_1$ and $U_2$ that are of type $p$ in $U_1$ and of type $q$ in $U_2$.

**Goal:** Compute the sum $T_{1,1} + T_{1,2} + T_{2,1} + T_{2,2}$ (i.e., the number of rooted fan triplets consistent with both $U_1$ and $U_2$).

- $T_{1,1} + 2T_{1,2} + 2T_{2,1} + 4T_{2,2}$ can be computed in $O(n^{(3+\omega)/2})$ time.
  (Use Lemma 6 to compute the number of rooted fan triplets shared between $U_1\downarrow$ and $U_2\downarrow$, between $U_1\downarrow$ and $U_2\uparrow$, between $U_1\uparrow$ and $U_2\downarrow$, and between $U_1\uparrow$ and $U_2\uparrow$, and sum them up.)

- $T_{2,2}$ can be computed in $O(n^{(3+\omega)/2})$ time.
  (Similar to Lemma 6. For each $\{u, v\} \in E$: Let $y_i = lca_{U_i\uparrow}(u, v)$ and $z_i = lca_{U_i\downarrow}(u, v)$ for $i \in \{1, 2\}$. If $y_1 = z_1$ and $y_2 = z_2$ then color $\{u, v\}$ with $(y_1, y_2)$; otherwise, color $\{u, v\}$ with a null color that never occurs again.)
Counting the shared rooted fan triplets

For $p, q \in \{1, 2\}$, let $T_{p,q}$ = the number of rooted fan triplets consistent with both $U_1$ and $U_2$ that are of type $p$ in $U_1$ and of type $q$ in $U_2$.

**Goal:** Compute the sum $T_{1,1} + T_{1,2} + T_{2,1} + T_{2,2}$ (i.e., the number of rooted fan triplets consistent with both $U_1$ and $U_2$).

- $T_{1,1} + 2T_{1,2} + 2T_{2,1} + 4T_{2,2}$ can be computed in $O(n^{(3+\omega)/2})$ time.
  (Use Lemma 6 to compute the number of rooted fan triplets shared between $U_1 \downarrow$ and $U_2 \downarrow$, between $U_1 \downarrow$ and $U_2 \uparrow$, between $U_1 \uparrow$ and $U_2 \downarrow$, and between $U_1 \uparrow$ and $U_2 \uparrow$, and sum them up.)

- $T_{2,2}$ can be computed in $O(n^{(3+\omega)/2})$ time.
  (Similar to Lemma 6. For each $\{u, v\} \in E$: Let $y_i = lca_{U_i \downarrow}^i (u, v)$ and $z_i = lca_{U_i \uparrow}^i (u, v)$ for $i \in \{1, 2\}$. If $y_1 = z_1$ and $y_2 = z_2$ then color $\{u, v\}$ with $(y_1, y_2)$; otherwise, color $\{u, v\}$ with a null color that never occurs again.)

- $T_{1,2}$ and $T_{2,1}$ can be computed in $O(n^{(3+\omega)/2})$ time.
  (Use the technical lemma!)
- $T_{1,2}$ and $T_{2,1}$ can be computed in $O(n^{(3+\omega)/2})$ time.
  (Use the technical lemma!)
- $T_{1,2}$ and $T_{2,1}$ can be computed in $O(n^{(3+\omega)/2})$ time.
  (Use the technical lemma!)

**Details:** As before, create an undirected complete graph $G = (L, E)$ and assign a color to each edge $\{u, v\} \in E$ as follows:

- Let $x_1 = lca^{U_1}(u, v)$, $y_2 = lca^{U_2\downarrow}(u, v)$, and $z_2 = lca^{U_2\uparrow}(u, v)$.
  If $x_1$ is the lca of $u$ and $v$ in both $U_1\downarrow$ and $U_1\uparrow$ while $y_2 = z_2$ then $\{u, v\}$ is colored with $(x_1, y_2)$. On the other hand, if $x_1$ is the lca of $u$ and $v$ in exactly one of $U_1\downarrow$ and $U_1\uparrow$ while $y_2 = z_2$ then $\{u, v\}$ is colored with $(x_1^*, y_2)$.
  Otherwise, $\{u, v\}$ is colored with a null color that never occurs again in $G$. 

- $T_{1,2}$ and $T_{2,1}$ can be computed in $O(n^{3+\omega}/2)$ time.
  (Use the technical lemma!)

**Details:** As before, create an undirected complete graph $G = (L, E)$ and assign a color to each edge $\{u, v\} \in E$ as follows:

- Let $x_1 = \text{lca}^{U_1}(u, v)$, $y_2 = \text{lca}^{U_2\\downarrow}(u, v)$, and $z_2 = \text{lca}^{U_2\\uparrow}(u, v)$.
  If $x_1$ is the lca of $u$ and $v$ in both $U_1\\downarrow$ and $U_1\\uparrow$ while $y_2 = z_2$ then $\{u, v\}$ is colored with $(x_1, y_2)$. On the other hand, if $x_1$ is the lca of $u$ and $v$ in exactly one of $U_1\\downarrow$ and $U_1\\uparrow$ while $y_2 = z_2$ then $\{u, v\}$ is colored with $(x_1^*, y_2)$. Otherwise, $\{u, v\}$ is colored with a null color that never occurs again in $G$.

Here, colors of the form $(w^*, \ldots)$ are used to indicate that the lca in $U_1$ of two leaves in $L$ is also the lca in one of, but not both of, $U_1\\downarrow$ and $U_1\\uparrow$. 
$T_{1,2}$ and $T_{2,1}$ can be computed in $O(n^{(3+\omega)/2})$ time.
(Use the technical lemma!)

**Details:** As before, create an undirected complete graph $G = (L,E)$ and assign a color to each edge $\{u, v\} \in E$ as follows:

- Let $x_1 = \text{lca}^{U_1}(u, v)$, $y_2 = \text{lca}^{U_2'}(u, v)$, and $z_2 = \text{lca}^{U_2'}(u, v)$.
  If $x_1$ is the lca of $u$ and $v$ in both $U_1'$ and $U_1'$ while $y_2 = z_2$ then $\{u, v\}$ is colored with $(x_1, y_2)$. On the other hand, if $x_1$ is the lca of $u$ and $v$ in exactly one of $U_1'$ and $U_1'$ while $y_2 = z_2$ then $\{u, v\}$ is colored with $(x_1^*, y_2)$. Otherwise, $\{u, v\}$ is colored with a null color that never occurs again in $G$.

Here, colors of the form $(w^*, \ldots)$ are used to indicate that the lca in $U_1'$ of two leaves in $L$ is also the lca in one of, but not both of, $U_1'$ and $U_1'$.

Define a binary relation $R$ on the edge colors of $G$ by:

- $(i_1, i_2)R(k_1, k_2)$ holds if and only if: (i) either $i_1 = k_1^*$ or $k_1 = j^*$, where $j$ is a proper descendant of $i_1$; and (ii) $i_2 = k_2$.

Then by Lemma 5, the number of $R$-chromatic triangles in $G$ equals $T_{1,2}$. 
\((i_1, i_2) R(k_1, k_2)\) holds if and only if: (i) either \(i_1 = k_1^*\) or \(k_1 = j^*\), where \(j\) is a proper descendant of \(i_1\); and (ii) \(i_2 = k_2\).
Counting the shared rooted triplets

This yields:

**Lemma 7**

Let $U_1$, $U_2$ be two galled trees with the same set of $n$ leaf labels. The number of *rooted fan triplets* consistent with both $U_1$ and $U_2$ can be computed in $O(n^{3+\omega}/2) \leq o(n^{2.687})$ time.
This yields:

**Lemma 7**

Let $U_1$, $U_2$ be two galled trees with the same set of $n$ leaf labels. The number of *rooted fan triplets* consistent with both $U_1$ and $U_2$ can be computed in $O(n^{3+\omega}/2) \leq o(n^{2.687})$ time.

What about the shared rooted binary triplets?
Counting the shared rooted triplets

This yields:

**Lemma 7**

Let $U_1$, $U_2$ be two galled trees with the same set of $n$ leaf labels. The number of *rooted fan triplets* consistent with both $U_1$ and $U_2$ can be computed in $O(n^{3+\omega}) \leq o(n^{2.687})$ time.

What about the shared rooted binary triplets?

Use matrix multiplication directly to count, for every pair of vertices $v_1, v_2$, where $v_1 \in U_1$ and $v_2 \in U_2$, how many of the leaves in $L$ that have a jca with $v_1$ in $U_1$ and a jca with $v_2$ in $U_2$ which do not coincide with certain vertices:

**Lemma 8**

Let $U_1$, $U_2$ be two galled trees with the same set of $n$ leaf labels. The number of *rooted binary triplets* consistent with both $U_1$ and $U_2$ can be computed in $O(n^\omega) \leq o(n^{2.373})$ time.
Combining Lemma 7 and Lemma 8:

**Theorem 4**

The rooted triplet distance between two galled trees $U_1, U_2$ with the same set of $n$ leaf labels can be computed in $O(n^{3+\omega}/2)$ time, where $n = |L|$.

\[ \Rightarrow O(n^{3+\omega}/2) \leq o(n^{2.687}) \]
Putting it all together

Combine Lemma 7 and Lemma 8:

**Theorem 4**

The rooted triplet distance between two galled trees $U_1, U_2$ with the same set of $n$ leaf labels can be computed in $O(n^{3+\omega}/2)$ time, where $n = |L|$.

\[ \Rightarrow O(n^{3+\omega}/2) \leq o(n^{2.687}) \]

**Proof:**

For $i \in \{1, 2\}$, let $F_i =$ the set of rooted fan triplets consistent with $U_i$, and $B_i =$ the set of rooted binary triplets consistent with $U_i$.

Then:

\[ d_{rt}(U_1, U_2) = \sum_{i=1}^{2}(|F_i| + |B_i|) - 2|F_1 \cap F_2| - 2|B_1 \cap B_2| \]
Putting it all together

Combine Lemma 7 and Lemma 8:

**Theorem 4**

The rooted triplet distance between two galled trees $U_1, U_2$ with the same set of $n$ leaf labels can be computed in $O(n^{3+\omega/2})$ time, where $n = |L|$.

$\Rightarrow O(n^{3+\omega/2}) \leq o(n^{2.687})$

**Proof:**

For $i \in \{1, 2\}$, let $F_i$ = the set of rooted fan triplets consistent with $U_i$, and $B_i$ = the set of rooted binary triplets consistent with $U_i$.

Then: $d_{rt}(U_1, U_2) = \sum_{i=1}^{2}(|F_i| + |B_i|) - 2|F_1 \cap F_2| - 2|B_1 \cap B_2|$

Compute $|F_i \cap F_i| = |F_i|$ and $|F_1 \cap F_2|$ in $O(n^{(3+\omega)/2})$ time using Lemma 7. Compute $|B_i \cap B_i| = |B_i|$ and $|B_1 \cap B_2|$ in $O(n^{\omega})$ time using Lemma 8.
We have shown that $d_{rt}$ for two galled trees can be computed in subcubic time. More precisely: $O(n^{\frac{3+\omega}{2}}) \leq o(n^{2.687})$ time.
Open problems

- Can $d_{rt}$ for two galled trees be computed in $O(n^2)$ time?
Open problems

- Can $d_{rt}$ for two galled trees be computed in $O(n^2)$ time?

- Can our method be extended to larger classes of phylogenetic networks than galled trees? E.g., *level-k phylogenetic networks* [Choy, Jansson, Sadakane, Sung; 2004] for any positive integer $k$?
Open problems

- Can $d_{rt}$ for two galled trees be computed in $O(n^2)$ time?

- Can our method be extended to larger classes of phylogenetic networks than galled trees? E.g., *level-k phylogenetic networks* [Choy, Jansson, Sadakane, Sung; 2004] for any positive integer $k$?
Open problems

- Can $d_{rt}$ for two galled trees be computed in $O(n^2)$ time?

- Can our method be extended to larger classes of phylogenetic networks than galled trees? E.g., level-$k$ phylogenetic networks [Choy, Jansson, Sadakane, Sung; 2004] for any positive integer $k$?
Open problems

- Can $d_{rt}$ for two galled trees be computed in $O(n^2)$ time?

- Can our method be extended to larger classes of phylogenetic networks than galled trees? E.g., *level-k phylogenetic networks* [Choy, Jansson, Sadakane, Sung; 2004] for any positive integer $k$?
Open problems

- Can $d_{rt}$ for two galled trees be computed in $O(n^2)$ time?

- Can our method be extended to larger classes of phylogenetic networks than galled trees? E.g., *level-k phylogenetic networks* [Choy, Jansson, Sadakane, Sung; 2004] for any positive integer $k$?

⇒ Level-2 network
Open problems

- Can $d_{rt}$ for two galled trees be computed in $O(n^2)$ time?

- Can our method be extended to larger classes of phylogenetic networks than galled trees? E.g., *level-k phylogenetic networks* [Choy, Jansson, Sadakane, Sung; 2004] for any positive integer $k$?

- What about fast algorithms for computing the unrooted quartet distance for *unrooted* phylogenetic networks?

Unrooted quartet:
Open problems

- Can $d_{rt}$ for two galled trees be computed in $O(n^2)$ time?

- Can our method be extended to larger classes of phylogenetic networks than galled trees? E.g., *level-k phylogenetic networks* [Choy, Jansson, Sadakane, Sung; 2004] for any positive integer $k$?

- What about fast algorithms for computing the unrooted quartet distance for *unrooted* phylogenetic networks?

\[
\begin{array}{c}
\text{Unrooted quartet:} \\
\begin{array}{c}
\bullet \quad w \\
\bullet \quad x \\
\bullet \quad y \\
\bullet \quad z
\end{array}
\end{array}
\]

- Do an experimental analysis to validate the rooted triplet distance as a means to compare phylogenetic networks. How well does it capture the notion of dissimilarity between galled trees in practice?
Open problems

- Can $d_{rt}$ for two galled trees be computed in $O(n^2)$ time?

- Can our method be extended to larger classes of phylogenetic networks than galled trees? E.g., *level-k phylogenetic networks* [Choy, Jansson, Sadakane, Sung; 2004] for any positive integer $k$?

- What about fast algorithms for computing the *unrooted quartet distance* for unrooted phylogenetic networks?

  Unrooted quartet:

  ![Unrooted quartet diagram](image)

- Do an experimental analysis to validate the rooted triplet distance as a means to compare phylogenetic networks. How well does it capture the notion of dissimilarity between galled trees in practice?

  One potential practical issue is that $d_{rt}$ IS NOT A METRIC!
Open problems

- Do an experimental analysis to validate the rooted triplet distance as a means to compare phylogenetic networks. How well does it capture the notion of dissimilarity between galled trees in practice?

One potential practical issue is that $d_{rt}$ IS NOT A METRIC!

**Example:** Both $ab|c$ and $bc|a$ are consistent with each of:

\[
\begin{align*}
& a \quad b \quad c \\
& a \\
& b \\
\end{align*}
\]

\[
\begin{align*}
& a \quad c \\
& a \\
& b \\
\end{align*}
\]

\[
\begin{align*}
& a \quad b \\
& a \\
& c \\
\end{align*}
\]

⇒ “$d_{rt}(U_1, U_2) = 0$ if and only if $U_1$ and $U_2$ are isomorphic” is FALSE.
Open problems

- Do an experimental analysis to validate the rooted triplet distance as a means to compare phylogenetic networks. How well does it capture the notion of dissimilarity between galled trees in practice?

One potential practical issue is that $d_{rt}$ IS NOT A METRIC!

**Example:** Both $ab|c$ and $bc|a$ are consistent with each of:

```
    a
   / \  
  b   c
    /   
   a   c
```

⇒ “$d_{rt}(U_1, U_2) = 0$ if and only if $U_1$ and $U_2$ are isomorphic” is FALSE.

However, $d_{rt}$ is a metric for the class $C_1^-$ (binary galled trees with no underlying cycles of length 4) [Gambette & Huber; 2012].
Open problems

- Can \( d_{rt} \) for two galled trees be computed in \( O(n^2) \) time?

- Can our method be extended to larger classes of phylogenetic networks than galled trees? E.g., *level-k phylogenetic networks* [Choy, Jansson, Sadakane, Sung; 2004] for any positive integer \( k \)?

- What about fast algorithms for computing the unrooted quartet distance for *unrooted* phylogenetic networks?

  Unrooted quartet:

  ![Unrooted quartet diagram](image)

- Do an experimental analysis to validate the rooted triplet distance as a means to compare phylogenetic networks. How well does it capture the notion of dissimilarity between galled trees in practice?

  One potential practical issue is that \( d_{rt} \) *IS NOT A METRIC!*

- Better ways to generalize the “rooted triplet distance” from trees to networks? E.g., use *trinets* [Huber & Moulton; 2013] for galled trees instead?
Further reading

This talk was based on: