Lecture: Bioinformatics

ENS Sacley, 2018

Some slides graciously provided by Daniel Huson & Celine Scornavacca
Phylogenetic Trees - Motivation

I think
Motivation

- study relation between species
- evolution of characteristics
- co-evolution (host-parasite)
- geological migration
- genetic development of viruses/diseases
Phylogenetic Trees - Motivation

**Motivation**

- Study relation between species
- Evolution of characteristics
- Co-evolution (host-parasite)
- Geological migration
- Genetic development of viruses/diseases

**Evolution**

genetic material changes over time

⇒ new species "Branch off"

⇒ "Tree of life"
Phylogenetic Trees - Motivation

[Phylogenetic tree diagram showing genetic distances and evolutionary relationships among different human populations.]

- New Guinean and Australian
- Pacific Islander
- Southeast Asian
- Northeast Asian
- Arctic Northeast Asian
- Amerind
- European
- North African and West Asian
- African

Genetic distance:
- 0.20
- 0.15
- 0.10
- 0.05
- 0.00
Phylogenetic Trees - Motivation

Ou et al. (1992), Page et Holmes (1998)
Rooted Phylogenetic Trees

evolution of species over time,
leaves extant,
hypothetical ancestors
possibly branch lengths (time)

Notation

taxon, cluster, triplet
Rooted Phylogenetic Trees

Evolution of species over time, leaves extant, hypothetical ancestors possibly branch lengths (time)

Notation

taxon, cluster, triplet

Exercise:

\[ xy | z \leftrightarrow LCA(xz) = LCA(yz) > LCA(xy) \]

Exercise

Time
Rooted Phylogenetic Trees

evolution of species over time,
leaves extant,
hypothetical ancestors
possibly branch lengths (time)

Notation

taxon, cluster, triplet
Rooted Phylogenetic Trees

evolution of species over time, leaves extant, hypothetical ancestors possibly branch lengths (time)

Notation
taxon, cluster, triplet
Rooted Phylogenetic Trees

evolution of species over time, leaves extant, hypothetical ancestors possibly branch lengths (time)

Exercise:
use $xy\|z \leftrightarrow \text{LCA}(xz)=\text{LCA}(yz)>\text{LCA}(xy)$ to prove $ab\|c + bc\|d \rightarrow ac\|d$

Notation

taxon, cluster, triplet

"Polytomies"

history not clear $\Rightarrow$ "soft"
known "fan out" $\Rightarrow$ "hard"
Unrooted Phylogenetic Trees

similarity between genomes,
leaves extant,
internal vertices have no meaning
possibly branch lengths (amount of change)

Notation

taxon, split, quartet
Unrooted Phylogenetic Trees

similarity between genomes, leaves extant, internal vertices have no meaning possibly branch lengths (amount of change)

Notation

- taxon
- split
- quartet
Unrooted Phylogenetic Trees

similarity between genomes,
leaves extant,
internal vertices have no meaning
possibly branch lengths (amount of change)

Notation

- taxon, split, quartet
Unrooted Phylogenetic Trees

similarity between genomes,
leaves extant,
internal vertices have no meaning
possibly branch lengths (amount of change)

Notation
taxon, split, quartet
Unrooted Phylogenetic Trees

similarity between genomes, leaves extant, internal vertices have no meaning possibly branch lengths (amount of change)

Notation
taxon, split, quartet
Reconstructing Phylogenetic Trees

Group Species By...

- morphology
- behavior
- geography

Diptera = two wings
Reconstructing Phylogenetic Trees

Group Species By...

- morphology
- behavior
- geography
- distance of sequences
- "genetic distance"
- etc.

Diptera = two wings
Reconstructing Phylogenetic Trees

Vertebrata
has backbone
Reconstructing Phylogenetic Trees

Vertebrata
has backbone

Tetrapoda
has 4 legs
Reconstructing Phylogenetic Trees

Vertebrata
- has backbone

Tetrapoda
- has 4 legs

Mammalia
- breast feeding

Keep in Mind
- automatic reconstruction should be:
  - fast (deal with tons of species & genes)
  - consistent (optimal data $\Rightarrow$ evolution correctly reflected)
  - non-arbitrary
Reconstructing Phylogenetic Trees

Vertebrata
- has backbone

Tetrapoda
- has 4 legs

- Salientia
  - can leap

- Mammalia
  - breast feeding
Reconstructing Phylogenetic Trees

Vertebrata
has backbone

Tetrapoda
has 4 legs

Salientia
can leap

Mammalia
breast feeding

Elephantidae
elephants

Asian Elephant
Loxodonta africana

African Elephant
Elephas maximus
Reconstructing Phylogenetic Trees

Vertebrata
- has backbone

Tetrapoda
- has 4 legs

Salientia
- can leap

Mammalia
- breast feeding

Elephantidae
- elephants

Loxodonta africana
- African Elephant

Elephas maximus
- Asian Elephant
Reconstructing Phylogenetic Trees

Keep in Mind
Automatic reconstruction should be
- fast (deal with tons of species & genes)
- consistent (optimal data → evolution correctly reflected)
- non-arbitrary

Vertebrata
has backbone

Tetrapoda
has 4 legs

Salientia
can leap

Mammalia
Breast feeding

Elephantidae
elephants

Loxodonta africana
African Elephant

Elephas maximus
Asian Elephant
Distance-Based Reconstruction

Idea: cluster hierarchically

\[ d_{X \cup Y, Z} = |X|d_{X, Z} + |Y|d_{Y, Z} \]

branch lengths?

assume molecular clock \( \Rightarrow \) ultrametric

Exercise Time

\[ \frac{7}{31} \]
Distance-Based Reconstruction

**Idea:** cluster hierarchically

- Distance update:
  \[ d(X \cup Y, Z) = \min(d(X, Z), d(Y, Z)) \]

- Branch lengths
  \[ \begin{array}{c} 2 \ 2 \ 3 \ \ 3 \\ 5/3 \ \ 5/3 \ \ 5/3 \end{array} \]

- Unweighted Pair Group Method with Arithmetic Mean (UPGMA):
  - find "closest pair"
  - "join" them
  - update distances & recurse
Distance-Based Reconstruction

Idea: cluster hierarchically

```
11
  8  7
  6  11  8
9  4  5  9
```

Idea: merge closest clusters
Distance-Based Reconstruction

Idea: cluster hierarchically

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>11</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>7</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>9</td>
<td>4</td>
<td>59</td>
</tr>
</tbody>
</table>

Idea: merge closest clusters
Distance-Based Reconstruction

Idea: cluster hierarchically

Idea: merge closest clusters

Update matrix

Branch lengths

Assume molecular clock

⇝ ultrametric

Unweighted Pair Group Method w/ Average

- find "closest pair"
- "join" them
- update distances & recurse

Exercise Time 7 / 31
Distance-Based Reconstruction

Idea: cluster hierarchically

Idea: merge closest clusters

Branch lengths ??

assume molecular clock \( \Rightarrow \) ultrametric
Distance-Based Reconstruction

**Idea:** cluster hierarchically

1. 11
2. 8 7
3. 6 11 8
4. 9 4 5 9

**Idea:** merge closest clusters

**Update matrix**

\[ d_{X \cup Y, Z} = \frac{|X|d_{X, Z} + |Y|d_{Y, Z}}{|X| + |Y|} \]

**Branch lengths ??**

assume molecular clock

\[ \sim \text{ultrametric} \]
Distance-Based Reconstruction

Idea: cluster hierarchically

Idea: merge closest clusters

update matrix

\[ d_{X \cup Y, Z} = \frac{|X|d_{X, Z} + |Y|d_{Y, Z}}{|X| + |Y|} \]

Branch lengths ??

assume molecular clock \( \leadsto \) ultrametric
Distance-Based Reconstruction

Idea: cluster hierarchically

update matrix

$$d_{X\cup Y,Z} = \frac{|X|d_{x,z} + |Y|d_{y,z}}{|X| + |Y|}$$

Idea: merge closest clusters

Branch lengths ??

assume molecular clock

$\sim$ ultrametric
Distance-Based Reconstruction

Idea: cluster hierarchically

update matrix

\[ d_{X \cup Y, Z} = \frac{|X|d_{x, z} + |Y|d_{y, z}}{|X| + |Y|} \]

Branch lengths ??

assume molecular clock

\[ \sim \text{ultrametric} \]
Distance-Based Reconstruction

**Idea:** cluster hierarchically

\[ d_{X \cup Y, Z} = \frac{|X| d_{X, Z} + |Y| d_{Y, Z}}{|X| + |Y|} \]

**Idea:** merge closest clusters

**Update matrix**

Branch lengths ??

assume molecular clock

\[ \leadsto \text{ultrametric} \]
Distance-Based Reconstruction

**Idea:** cluster hierarchically

**Idea:** merge closest clusters

**Update matrix**

\[ d_{X\cup Y,Z} = \frac{|X|d_{X,Z} + |Y|d_{Y,Z}}{|X| + |Y|} \]

**Branch lengths ??**

assume molecular clock

\[ \sim \text{ultrametric} \]
Distance-Based Reconstruction

Idea: cluster hierarchically

Idea: merge closest clusters

update matrix

\[ d_{X \cup Y, Z} = \frac{|X| d_{X, Z} + |Y| d_{Y, Z}}{|X| + |Y|} \]

Branch lengths ??

assume molecular clock

\[ \sim \text{ultrametric} \]
Distance-Based Reconstruction

Idea: cluster hierarchically

Unweighted Pair Group Method w/ Avg.
- find "closest pair"
- "join" them
- update distances & recurse

update matrix
\[ d_{X \cup Y, Z} = \frac{|X|d_{X, Z} + |Y|d_{Y, Z}}{|X| + |Y|} \]

Idea: merge closest clusters

Branch lengths ??
assume molecular clock \( \leadsto \) ultrametric

28/3
Distance-Based Reconstruction

Idea: cluster hierarchically

11
8 7
6 11 8
9 4 5 9

Idea: merge closest clusters

update matrix

$$d_{X\cup Y, Z} = \frac{|X|d_{X, Z} + |Y|d_{Y, Z}}{|X| + |Y|}$$

Unweighted Pair Group Method w/ Avg.

- find "closest pair"
- "join" them
- update distances & recurse

Branch lengths ??
assume molecular clock
~ ultrametric
Distance-Based Reconstruction

**Idea:** cluster hierarchically

```
12
12 10
12 6 10
12 2 10 6
```

**Idea:** merge closest clusters

\[ d_{X \cup Y, Z} = \frac{|X|d_{x, z} + |Y|d_{y, z}}{|X| + |Y|} \]

**Unweighted Pair Group Method w/ Avg.**

- find "closest pair"
- "join" them
- update distances \& recurse

**Exercise Time**

Branch lengths ??
assume molecular clock
\( \rightsquigarrow \) ultrametric
Distance-Based Reconstruction

**Idea:** cluster hierarchically

- 12
- 12 10
- 12 6 10
- 12 2 10 6

**Idea:** merge closest clusters

**Unweighted Pair Group Method w/ Avg.**

- find "closest pair"
- "join" them
- update distances & recurse

**Update matrix**

\[ d_{XU,Y,Z} = \frac{|X|d_{x,z} + |Y|d_{y,z}}{|X| + |Y|} \]

**Branch lengths ??**

assumes molecular clock

\[ \sim \text{ultrametric} \]
Distance-Based Reconstruction

What about unrooted trees?
Distance-Based Reconstruction

What about unrooted trees?
Distance-Based Reconstruction

**Problem:** correct pairs may not be closest

<table>
<thead>
<tr>
<th>B</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>11 8</td>
</tr>
<tr>
<td>D</td>
<td>9 12 10</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>
Distance-Based Reconstruction

**Problem:** correct pairs may not be closest

<table>
<thead>
<tr>
<th></th>
<th>B</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>C</td>
<td>11</td>
<td>8</td>
</tr>
<tr>
<td>D</td>
<td>9</td>
<td>12</td>
</tr>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>
Distance-Based Reconstruction

**Problem:** correct pairs may not be closest

**Neighbor Joining (unrooted)**
- Build new matrix:
  \[ Q_{x,y} = \sum_z (d_{x,z} + d_{y,z} - d_{x,y}) + 2d_{x,y} \]
- Find max in \( Q \)
- Join them
- Update distances & recurse
Distance-Based Reconstruction

**Problem:** correct pairs may not be closest

Neighbor Joining (unrooted)

- Build new matrix:
  \[ Q_{x,y} = \sum_z (d_{x,z} + d_{y,z} - d_{x,y}) + 2d_{x,y} \]
- Find max in \( Q \)
- Join them
- Update distances & recurse
Distance-Based Reconstruction

**Problem:** correct pairs may not be closest

![Illustration of different animals with distances]

**Neighbor Joining (unrooted)**

- Build new matrix:
  \[ Q_{X,Y} = \sum Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y} \]
- Find max in \( Q \)
- Join them
- Update distances & recurse

**Theorem**

\[ Q_{X,Y} \text{ max} \iff \text{any tree } T \text{ yielding } Q \text{ has "cherry" } (X, Y) \]
Distance-Based Reconstruction

**Problem:** correct pairs may not be closest

Neighbor Joining (unrooted)
- Build new matrix:
  \[ Q_{x,y} = \sum_z (d_{x,z} + d_{y,z} - d_{x,y}) + 2d_{x,y} \]
- Find max in \( Q \)
- Join them
- Update distances & recurse

Update distances
\[ d_{x\cup y,z} = \frac{1}{2}(d_{x,z} + d_{y,z} - d_{x,y}) \]

Branch lengths
\[ 2b(X) = \sum_{z \notin \{x,y\}} \frac{(d_{x,z} - d_{y,z} + d_{x,y})}{n-2} \]
Distance-Based Reconstruction

Problem: correct pairs may not be closest

Neighbor Joining (unrooted)
- Build new matrix:
  \[ Q_{x,y} = \sum_z (d_{x,z} + d_{y,z} - d_{x,y}) + 2d_{x,y} \]
- Find max in \( Q \)
- Join them
- Update distances & recurse

Update distances
\[ d_{x \cup y, z} = \frac{1}{2} (d_{x, z} + d_{y, z} - d_{x, y}) \]

Branch lengths
\[ 2b(X) = \frac{\sum_{z \not\in \{x, y\}} (d_{x, z} - d_{y, z} + d_{x, y})}{n-2} \]
Distance-Based Reconstruction

**Problem:** correct pairs may not be closest

Neighbor Joining (unrooted)

- Build new matrix: 
  \[ Q_{x,y} = \sum_z (d_{x,z} + d_{y,z} - d_{x,y}) + 2d_{x,y} \]
- Find max in \( Q \)
- Join them
- Update distances \& recurse

Update distances

\[ d_{x\cup y,z} = \frac{1}{2} (d_{x,z} + d_{y,z} - d_{x,y}) \]

Branch lengths

\[ 2b(X) = \frac{\sum_{z \notin \{x,y\}} (d_{x,z} - d_{y,z}) + d_{x,y}}{n-2} \]
Distance-Based Reconstruction

**Problem:** correct pairs may not be closest

**Neighbor Joining (unrooted)**
- Build new matrix: 
  \[ Q_{X,Y} = \sum_Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y} \]
- Find max in \( Q \)
- Join them
- Update distances and recurse

**Update distances**
\[ d_{X \cup Y,Z} = \frac{1}{2}(d_{X,Z} + d_{Y,Z} - d_{X,Y}) \]

**Branch lengths**
\[ 2b(X) = \frac{\sum_{Z \notin \{X,Y\}} (d_{X,Z} - d_{Y,Z} + d_{X,Y})}{n-2} \]
Distance-Based Reconstruction

**Problem:** correct pairs may not be closest

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>6</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>4</td>
<td>3</td>
</tr>
</tbody>
</table>

**Neighbor Joining (unrooted)**

- build new matrix:
  \[ Q_{x,y} = \sum_z (d_{x,z} + d_{y,z} - d_{x,y}) + 2d_{x,y} \]
- find max in \( Q \)
- join them
- update distances & recurse

**update distances**

\[ d_{X \cup Y, z} = \frac{1}{2}(d_{X,z} + d_{Y,z} - d_{X,Y}) \]

**Branch lengths**

\[ 2b(X) = \frac{\sum_{z \notin \{x,y\}} (d_{x,z} - d_{y,z} + d_{x,y})}{n-2} \]
Distance-Based Reconstruction

**Problem:** correct pairs may not be closest

**Neighbor Joining (unrooted)**
- Build new matrix:
  \[ Q_{x,y} = \sum_z (d_{x,z} + d_{y,z} - d_{x,y}) + 2d_{x,y} \]
- Find max in \( Q \)
- Join them
- Update distances \& recurse

**Update distances**
\[ d_{x \cup y, z} = \frac{1}{2}(d_{x,z} + d_{y,z} - d_{x,y}) \]

**Branch lengths**
\[ 2b(X) = \sum_{z \notin \{x, y\}} \frac{(d_{x,z} - d_{y,z} + d_{x,y})}{n-2} \]
Distance-Based Reconstruction

**Problem:** correct pairs may not be closest

**Neighbor Joining (unrooted)**
- Build new matrix:
  \[ Q_{X,Y} = \sum_Z (d_{X,Z} + d_{Y,Z} - d_{X,Y}) + 2d_{X,Y} \]
- Find max in \( Q \)
- Join them
- Update distances & recurse

**Update distances**
\[ d_{X \cup Y,Z} = \frac{1}{2}(d_{X,Z} + d_{Y,Z} - d_{X,Y}) \]

**Branch lengths**
\[ 2b(X) = \frac{\sum_{Z \notin \{X,Y\}} (d_{X,Z} - d_{Y,Z} + d_{X,Y})}{n-2} \]
Parsimony Reconstructing

Input: character state matrix $M$, rooted tree $T$

Task: assign characters to internal nodes minimizing total cost

$\Rightarrow O(nm)$ time [Fitch'71]

Large Parsimony

Input: character state matrix $M$

Task: find tree $T$ & assign characters to internal nodes minimizing total cost

$\Rightarrow$ NP-hard

Note: alignment is crucial!

Sum "distance" of endpoints of each edge

$\Rightarrow$ cost 6 (Hamming)

Exercise

Time

10 / 31
### Parsimony Reconstructing

<table>
<thead>
<tr>
<th></th>
<th>fur</th>
<th>AUS</th>
<th>pouch</th>
<th>land</th>
</tr>
</thead>
<tbody>
<tr>
<td>Monkey</td>
<td>✓</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Hunter</td>
<td>x</td>
<td>✓</td>
<td>x</td>
<td>✓</td>
</tr>
<tr>
<td>Dolphin</td>
<td>x</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Kangaroo</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Hedgehog</td>
<td>✓</td>
<td>x</td>
<td>x</td>
<td>✓</td>
</tr>
</tbody>
</table>

#### Small Parsimony
- **Input:** character state matrix $M$
- **Task:** minimize the total cost of character states

- **Time Complexity:** $O(nm)$

- **Note:** Alignment is crucial! Sum of the distances of the endpoints of each edge can give the cost, which is $6$ (Hamming distance).

#### Large Parsimony
- **Input:** character state matrix $M$
- **Task:** find the tree $T$ and assign characters to internal nodes minimizing the total cost

- **Time Complexity:** $\text{NP-hard}$

**Exercise Time:** 10 / 31
Parsimony Reconstructing

<table>
<thead>
<tr>
<th>fur</th>
<th>AUS</th>
<th>pouch</th>
<th>land</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
<tr>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✓</td>
</tr>
</tbody>
</table>

Small Parsimony
Input: character state matrix M, rooted tree T
Task: assign characters to internal nodes minimizing total cost
\( \Rightarrow O(nm) \) time

Large Parsimony
Input: character state matrix M
Task: find tree T & assign characters to internal nodes minimizing total cost
\( \Rightarrow \text{NP-hard} \)

Note: alignment is crucial!

\[ \Rightarrow \text{cost} = \text{sum "distance" of endpoints of each edge} \]
Parsimony Reconstructing

<table>
<thead>
<tr>
<th>fur</th>
<th>AUS</th>
<th>pouch</th>
<th>land</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Small Parsimony

Input: character state matrix M, rooted tree T
Task: assign characters to internal nodes minimizing total cost
\( \Rightarrow O(nm) \) time

Large Parsimony

Input: character state matrix M
Task: find tree T & assign characters to internal nodes minimizing total cost
\( \Rightarrow \) NP-hard

Note: alignment is crucial!

Sum "distance" of endpoints of each edge
\( \Rightarrow \) cost 6 (Hamming)

Exercise

Time 10 / 31
Parsimony Reconstructing

fur    AUS    pouch    land
✓ ✓    ✓ ✓    ✓ ✓    ✓ ✓    ✓ ✓
✓ ✓    ✓ ✓    ✓ ✓    ✓ ✓    ✓ ✓
✓ ✓    ✓ ✓    ✓ ✓    ✓ ✓    ✓ ✓
✓ ✓    ✓ ✓    ✓ ✓    ✓ ✓    ✓ ✓

Small Parsimony
Input: character state matrix M,
Task: assign characters to internal
nodes minimizing total cost
\( \Rightarrow O(nm) \) time
\([Fitch'71]\)

Large Parsimony
Input: character state matrix M
Task: find tree T & assign
characters to internal nodes
minimizing total cost
\( \Rightarrow NP-hard \)
Note: alignment is crucial!

sum "distance" of endpoints of each edge
Parsimony Reconstructing

<table>
<thead>
<tr>
<th></th>
<th>fur</th>
<th>AUS</th>
<th>pouch</th>
<th>land</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="monkey.png" alt="Monkey" /></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><img src="dinosaur.png" alt="Dinosaur" /></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><img src="fish.png" alt="Fish" /></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td><img src="penguin.png" alt="Penguin" /></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

Small Parsimony

Input: character state matrix M, rooted tree T
Task: assign characters to internal nodes minimizing total cost
\[ \Rightarrow O(nm) \text{ time} \] [Fitch'71]

Large Parsimony

Input: character state matrix M
Task: find tree T & assign characters to internal nodes minimizing total cost
\[ \Rightarrow \text{NP-hard} \]
Note: alignment is crucial!

sum “distance” of endpoints of each edge \( \sim \) cost 6 (Hamming)

Exercise Time 10/31
Parsimony Reconstructing

Small Parsimony

Input: character state matrix $M$, rooted tree $T$

Task: assign characters to internal nodes minimizing total cost

Large Parsimony

Input: character state matrix $M$

Task: find tree $T$ and assign characters to internal nodes minimizing total cost
Parsimony Reconstructing

Small Parsimony

Input: character state matrix $M$, rooted tree $T$

Task: assign characters to internal nodes minimizing total cost

$\Rightarrow O(nm)$ time

[Fitch'71]

Exercise

Time

10 / 31
Parsimony Reconstructing

**Small Parsimony**

**Input:** character state matrix $M$, rooted tree $T$

**Task:** assign characters to internal nodes minimizing total cost

$\Rightarrow O(nm)$ time

(Fitch’71)

**Exercise Time**
Parsimony Reconstructing

Small Parsimony

Input: character state matrix \( M \), rooted tree \( T \)

Task: assign characters to internal nodes minimizing total cost

\( \Rightarrow \) \( O(nm) \) time

Large Parsimony

Input: character state matrix \( M \)

Task: find tree \( T \) and assign characters to internal nodes minimizing total cost

Note: alignment is crucial!
**Parsimony Reconstructing**

---

**Small Parsimony**

- **Input:** character state matrix $M$, rooted tree $T$
- **Task:** assign characters to internal nodes minimizing total cost

$\Rightarrow O(nm)$ time

[*(Fitch’71)*]

---

**Large Parsimony**

- **Input:** character state matrix $M$
- **Task:** find tree $T$ & assign characters to internal nodes minimizing total cost

$\Rightarrow$ NP-hard

---

**Exercise**

- Time: 10 / 31
Parsimony Reconstructing

**Small Parsimony**

Input: character state matrix \( M \), rooted tree \( T \)

Task: assign characters to internal nodes minimizing total cost

\( \Rightarrow O(nm) \) time

[Fitch'71]

**Large Parsimony**

Input: character state matrix \( M \)

Task: find tree \( T \) & assign characters to internal nodes minimizing total cost

\( \Rightarrow \) NP-hard

*Note*: alignment is crucial!
Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes.
Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes

\[ \Rightarrow \text{need model of evolution} \]
Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes

\[ \rightarrow \text{need model of evolution} \]

**Jukes & Cantor Model**

- each base evolves individually
- each base occurs with equal frequency in the genome
- constant rate $\mu$ of mutation
- each base is equally likely to be result of mutation
Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes
→ need model of evolution

Jukes & Cantor Model

- each base evolves individually
- each base occurs with equal frequency in the genome
- constant rate $\mu$ of mutation
- each base is equally likely to be result of mutation

Generalized Time Reversible Model

- each base evolves individually
- each base $X$ has a frequency $\pi_X$ to occur in the genome
- each base-substitution has its own rate of occurrence
Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes

⇝ need model of evolution

**Jukes & Cantor Model**
- each base evolves individually
- each base occurs with equal frequency in the genome
- constant rate $\mu$ of mutation
- each base is equally likely to be result of mutation

**Generalized Time Reversible Model**
- each base evolves individually
- each base $X$ has a frequency $\pi_X$ to occur in the genome
- each base-substitution has its own rate of occurrence

compute likelihood, given tree & parameters $\sim O(mn)$ time
Maximum Likelihood Reconstruction

Idea: find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes ➞ need model of evolution

**Jukes & Cantor Model**
- each base evolves individually
- each base occurs with equal frequency in the genome
- constant rate $\mu$ of mutation
- each base is equally likely to be result of mutation

**Generalized Time Reversible Model**
- each base evolves individually
- each base $X$ has a frequency $\pi_X$ to occur in the genome
- each base-substitution has its own rate of occurrence

compute likelihood, given tree & parameters ➞ $O(mn)$ time
find best tree & parameters ➞ NP-hard
Maximum Likelihood Reconstruction

**Idea:** find a tree (with branch lengths) under which evolution is most likely to have produced the observed characters/genomes

 yüzde  need model of evolution

**Jukes & Cantor Model**

- each base evolves individually
- each base occurs with equal frequency in the genome
- constant rate \( \mu \) of mutation
- each base is equally likely to be result of mutation

**Generalized Time Reversible Model**

- each base evolves individually
- each base \( X \) has a frequency \( \pi_X \) to occur in the genome
- each base-substitution has its own rate of occurrence

compute likelihood, given tree & parameters \( \Rightarrow O(mn) \) time
find best tree & parameters \( \Rightarrow \) NP-hard

\( \Rightarrow \) local search in the tree space
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another

[Diagram showing tree structures]
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) “neighboring” subtrees into another
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange
change any configuration of 4 (3) "neighboring" subtrees into another

Subtree Prune & Regraft
break any edge uv & connect v to any edge of the component of u
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange

change any configuration of 4 (3) "neighboring" subtrees into another

Subtree Prune & Regraft

break any edge uv & connect v to any edge of the component of u
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange
change any configuration of 4 (3) “neighboring” subtrees into another

Subtree Prune & Regraft
break any edge uv & connect v to any edge of the component of u
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space.

Nearest Neighbor Interchange

- Change any configuration of 4 (3) "neighboring" subtrees into another.

Subtree Prune & Regraft

- Break any edge $uv$ and connect $v$ to any edge of the component of $u$. 

---

ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space.

Nearest Neighbor Interchange

- Change any configuration of 4 (3) "neighboring" subtrees into another.

Subtree Prune & Regraft

- Break any edge $uv$ and connect $v$ to any edge of the component of $u$. 
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

**Nearest Neighbor Interchange**

change any configuration of 4 (3) “neighboring” subtrees into another

**Subtree Prune & Regraft**

break any edge uν and connect v to any edge of the component of u
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange
change any configuration of 4 (3) “neighboring” subtrees into another

Subtree Prune & Regraft
break any edge uv &
connect v to any edge of
the component of u

Tree Bisection & Reconnection
break any edge &
insert a new
reconnecting edge
"between" any 2 edges
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

**Nearest Neighbor Interchange**

change any configuration of 4 (3) "neighboring" subtrees into another

**Subtree Prune & Regraft**

break any edge uv \(\neq\) connect v to any edge of the component of u

**Tree Bisection & Reconnection**

break any edge \(\neq\) insert a new reconnecting edge "between" any 2 edges
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

**Nearest Neighbor Interchange**

change any configuration of 4 (3) "neighboring" subtrees into another

**Subtree Prune & Regraft**

break any edge uv & connect v to any edge of the component of u

**Tree Bisection & Reconnection**

break any edge & insert a new reconnecting edge "between" any 2 edges
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange
change any configuration of 4 (3) “neighboring” subtrees into another

Subtree Prune & Regraft
break any edge $uv \neq$ connect $v$ to any edge of the component of $u$

Tree Bisection & Reconnection
break any edge $\neq$ insert a new reconnecting edge "between" any 2 edges
ML Reconstruction - Tree Spaces

Observe: a tree and a rearrangement operation span a space

Nearest Neighbor Interchange
change any configuration of 4 (3) "neighboring" subtrees into another

Subtree Prune & Regraft
break any edge uv & connect v to any edge of the component of u

Tree Bisection & Reconnection
break any edge & insert a new reconnecting edge "between" any 2 edges
Exercise: turn into (any) caterpillar:

Exercise: how are the distances related?
Checking Robustness – Bootstrap Method

suppose: method X yields tree T from \( n \times m \) character-state matrix \( M \)

repeat \( k \) times the following experiment:

1. draw \( m \) columns from \( M \) (with repetition)
2. use \( X \) to compute \( T_i \)

Finally, for each branch of \( T \), check how often it occurs in the \( T_i \)

\( \Rightarrow \) "bootstrap value" measures robustness ("support") of each branch
Reconstruction by Gene Trees

1. Get genomes of multiple species
2. Extract "genes" using START & STOP codons
3. Cluster genes in "families" of similar genes
4. Within each family, infer a "gene tree" using dissimilarities
5. Build a consensus among the gene trees → "species tree" (Note: species tree may differ significantly from individual gene trees)
6. Reconcile all gene trees with the species tree to learn the evolution of those genes
Reconstruction by Gene Trees

A Common Method For Reconstructing Trees

1. get genomes of multiple species
2. extract "genes" using START & STOP codons
3. cluster genes in "families" of similar genes
4. within each family, infer a "gene tree" using dissimilarities
5. build a consensus among the gene trees $\rightarrow$ "species tree"
   (Note: species tree may differ significantly from individual gene trees)
6. reconcile all gene trees with the species tree to learn the evolution of those genes
Reconstruction by Gene Trees

A Common Method For Reconstructing Trees

1. get genomes of multiple species
2. extract “genes” using START & STOP codons
3. cluster genes in “families” of similar genes
4. within each family, infer a “gene tree” using dissimilarities
5. build a consensus among the gene trees \(\Rightarrow\) “species tree”
   (Note: species tree may differ significantly from individual gene trees)
6. reconcile all gene trees with the species tree to learn the evolution of those genes
Reconstruction by Gene Trees

A Common Method for Reconstructing Trees

1. Get genomes of multiple species
2. Extract “genes” using START & STOP codons
3. Cluster genes in “families” of similar genes
4. Within each family, infer a “gene tree” using dissimilarities
5. Build a consensus among the gene trees $\Rightarrow$ “species tree”
   (Note: species tree may differ significantly from individual gene trees)
6. Reconcile all gene trees with the species tree to learn the evolution of those genes
Reconstruction by Gene Trees

A Common Method For Reconstructing Trees

1. get genomes of multiple species
2. extract "genes" using START & STOP codons
3. cluster genes in "families" of similar genes
4. within each family, infer a "gene tree" using dissimilarities
5. build a consensus among the gene trees \( \rightarrow \) "species tree"
   (Note: species tree may differ significantly from individual gene trees)
6. reconcile all gene trees with the species tree to learn the evolution of those genes
Supertrees - "Build" Algorithm

Idea: find root partition and recurse (as long as there are $\geq 3$ taxa)
Supertrees - "Build" Algorithm

**Idea:** find root partition & recurse (as long as there are ≥3 taxa)
Supertrees - "Build" Algorithm

Idea: find root partition & recurse (as long as there are \( \geq 3 \) taxa)
Supertrees - "Build" Algorithm

Idea: find root partition \( \neq \) recurse (as long as there are \( \geq 3 \) taxa)

Note: always works if trees are compatible/consistent

- largest compatible subset \( \Rightarrow \) NP-hard (even for triplets)
- voting schemes (each tree votes for their clades)
- reinterpret clades as characters, combine into matrix & reconstruct
Supertrees - “Build” Algorithm

Idea: find root partition and recurse (as long as there are \( \geq 3 \) taxa)

Note: always works if trees are compatible/consistent
Supertrees - "Build" Algorithm

Idea: find root partition & recurse (as long as there are $\geq 3$ taxa)

Note: always works if trees are compatible/consistent
Supertrees - "Build" Algorithm

**Idea:** find root partition & recurse (as long as there are $\geq 3$ taxa)
Supertrees - "Build" Algorithm

Idea: find root partition & recurse (as long as there are $\geq 3$ taxa)

Note: always works if trees are compatible/consistent
Supertrees - "Build" Algorithm

Idea: find root partition & recurse (as long as there are \( \geq 3 \) taxa)

Note: always works if trees are compatible/consistent
Supertrees - "Build" Algorithm

Idea: find root partition \( \div \) recurse (as long as there are \( \geq 3 \) taxa)

Note: always works if trees are compatible/consistent
Supertrees - "Build" Algorithm

Idea: find root partition & recurse (as long as there are \( \geq 3 \) taxa)

Note: always works if trees are compatible/consistent

incompatible?
- largest compatible subset
  \( \leadsto \) NP-hard (even for triplets)
- voting schemes
  (each tree votes for their clades)
- reinterpret clades as characters, combine into matrix & reconstruct
Consensi of Non-Agreeing Trees

**Strict Consensus**

- Ochromonas
- Symbiodinium
- Prorocentrum
- Loxodes
- Tetrahymena
- Spirotrichonema
- Tracheloraphis
- *Euplotes*
- Gruberia

**Consensus Subtree**

- Symbiodinium
- Prorocentrum
- Loxodes
- Tetrahymena
- Spirotrichonema
- Tracheloraphis
- Gruberia
- *Ochromonas*

**Majority Consensus**

- Ochromonas
- Symbiodinium
- Prorocentrum
- Loxodes
- Tetrahymena
- Spirotrichonema
- *Euplotes*
- Tracheloraphis
- Gruberia
Reconstruction by Gene Trees

A Common Method For Reconstructing Trees

1. Get genomes of multiple species
2. Extract “genes” using START & STOP codons
3. Cluster genes in “families” of similar genes
4. Within each family, infer a “gene tree” using dissimilarities
5. Build a consensus among the gene trees \( \Rightarrow \) “species tree”
   (Note: species tree may differ significantly from individual gene trees)
6. Reconcile all gene trees with the species tree to learn the
evolution of those genes
Reconstruction by Gene Trees

A Common Method For Reconstructing Trees

1. get genomes of multiple species
2. extract "genes" using START & STOP codons
3. cluster genes in "families" of similar genes
4. within each family, infer a "gene tree" using dissimilarities
5. build a consensus among the gene trees "species tree"
   (Note: species tree may differ significantly from individual gene trees)
6. reconcile all gene trees with the species tree to learn the evolution of those genes
The History of a Gene Family

Recall

gene = "functional element" of DNA, clustered into gene-families

each family yields a tree depicting its history $\Rightarrow$ "gene tree"
consensus of the gene trees yields "species tree"
But: what really happened???

Mouse
Dog
Bat
Rat

19 / 31
The History of a Gene Family

Recall

gene = "functional element" of DNA, clustered into gene-families

each family yields a tree depicting its history \( \rightsquigarrow \) "gene tree"
consensus of the gene trees yields "species tree"
But: what really happened???
Reconciliation

Embedding Rules

gene tree $G$, species tree $S$
- mapping $\rho : V(G) \rightarrow V(S)$
- $\ell$ is leaf in $G \rightsquigarrow \rho(\ell)$ "corresponds" to $\ell$ ($a \rightarrow A$, $a' \rightarrow A$, etc.)
- $u \in V(G)$ is called duplication if $\rho(u) = \rho(c)$ for any child $c$ of $u$ in $G$
- all non-leaves of $G$ that not duplications are called speciations
- each edge $uv$ of $G$ incurs a loss-cost equal to the number of edges in the $\rho(u)$-$\rho(v)$-path in $S$ minus 1 if $v$ is a speciation or 0 if $v$ is a duplication
Reconciliation

DL-model
- ○ = speciation
- △ = duplication
- × = loss
Reconciliation

Goal: embed gene tree into species tree (extant genes must map to their species)

Max. Likelihood
find most probable embedding (computationally expensive)

Parsimony
find embedding minimizing #events (possibly weighted)

DL-model
○ = speciation
△ = duplication
× = loss
Reconciliation

Parsimonious Reconciliation

Input: species tree $S$, gene tree $G$, $\delta, \lambda \in \mathbb{N}$

Task: embed $G$ in $S$, minimizing the weighted sum of events

Result: LCA-assignment solves this optimally in $O(|S|+|G|)$

DL-model

- $\bigcirc$ = speciation (0)
- $\triangle$ = duplication ($\delta$)
- $\times$ = loss ($\lambda$)
Reconciliation

Parsimonious Reconciliation

Input: species tree $S$, gene tree $G$, $\delta, \lambda, \tau \in \mathbb{N}$

Task: embed $G$ in $S$, minimizing the weighted sum of events

Result: LCA-assignment solves this optimally in $O(|S|+|G|)$
Reconciliation

Parsimonious Reconciliation

Input: species tree $S$, gene tree $G$, $\delta, \lambda, \tau \in \mathbb{N}$

Task: embed $G$ in $S$, minimizing the weighted sum of events

Result: LCA-assignment solves this optimally in $O(|S|+|G|)$

DTL-model

- $\bigcirc =$ speciation $(0)$
- $\bigtriangleup =$ duplication $(\delta)$
- $\times =$ loss $(\lambda)$
- $\bigdiamond =$ transfer $(\tau)$
Reconciliation

Parsimonious Reconciliation

Input: species tree $S$, gene tree $G$, $\delta, \lambda, \tau \in \mathbb{N}$

Task: embed $G$ in $S$, minimizing the weighted sum of events

Result: LCA-assignment solves this optimally in $O(|S|+|G|)$

Events only between co-existing species $\rightarrow$ time constraints $\rightarrow$ NP-hard

DTL-model

- $\bigcirc$ = speciation (0)
- $\triangle$ = duplication ($\delta$)
- $\times$ = loss ($\lambda$)
- $\blacklozenge$ = transfer ($\tau$)
Reconciliation

Parsimonious Reconciliation

Input: species tree $S$, gene tree $G$, $\delta, \lambda, \tau \in \mathbb{N}$

Task: embed $G$ in $S$, minimizing the weighted sum of events

Result: LCA-assignment solves this optimally in $O(|S| + |G|)$

T events only between co-existing species $\sim$ time constraints $\sim$ NP-hard

Idea: dated species tree $\sim O(|S|^2|G|)$ [Doyon et al.’10]

DTL-model

- $\bigcirc$ = speciation (0)
- $\bigtriangleup$ = duplication ($\delta$)
- $\times$ = loss ($\lambda$)
- $\bigotimes$ = transfer ($\tau$)

Diagram
Comparing Phylogenetic Trees

**Distance Measures**
- Nearest Neighbor Interchange
- Subtree Prune & Regraft
- Tree Bisection & Reconnection
Comparing Phylogenetic Trees

Distance Measures
- Nearest Neighbor Interchange
- Subtree Prune & Regraft
- Tree Bisection & Reconnection
- Robinson-Foulds distance
- quartet/triplet distance
Agreement Forests

Definition

A forest $F$ is called agreement forest of trees $T_1$ and $T_2$ if $F$ can be obtained from $T_1$ and $T_2$ by removing edges.
Agreement Forests

**Definition**

A forest $F$ is called agreement forest of trees $T_1$ and $T_2$ if $F$ can be obtained from $T_1$ and $T_2$ by removing edges.
Agreement Forests

**Definition**

A forest $F$ is called agreement forest of trees $T_1$ and $T_2$ if $F$ can be obtained from $T_1$ and $T_2$ by removing edges.
Agreement Forests

**Definition**

A forest $F$ is called agreement forest of trees $T_1$ and $T_2$ if $F$ can be obtained from $T_1$ and $T_2$ by removing edges.
Agreement Forests

**Definition**
A forest $F$ is called agreement forest of trees $T_1$ and $T_2$ if $F$ can be obtained from $T_1$ and $T_2$ by removing edges.

**Theorem** [Allen & Steel, ’01]

$\text{TBR-distance}(T_1, T_2) = \#\text{trees in smallest agreement forest} - 1$

NP-hard to compute

**Theorem** [Bordewich & Semple, ’04]

$\text{rSPR-distance}(T_1, T_2) = \#\text{trees in smallest rooted agreement forest} - 1$

NP-hard to compute
Robinson-Foulds Distance

Definition

\[ RF(T_1, T_2) = \# \text{splits/clusters occurring in exactly one of } T_1 \text{ and } T_2 \]

= edge-contraction distance a common tree

Note: observe relation to NNI: \[ RF(T_1, T_2) \leq 2 \text{ NNI}(T_1, T_2) \]

trivial splits, \( \{A,B\} \mid \{C,D,E,F\} \), \( \{E,F\} \mid \{A,B,E,F\} \), \( \{C,D\} \mid \{A,B,E,F\} \)

trivial splits, \( \{A,B\} \mid \{C,D,E,F\} \), \( \{E,F\} \mid \{A,B,C,D\} \), \( \{A,B,D\} \mid \{C,E,F\} \)
Robinson-Foulds Distance

Definition

\[ RF(T_1, T_2) = \# \text{splits/clusters occurring in exactly one of } T_1 \text{ and } T_2 \]

= edge-contraction distance a common tree

Note: observe relation to NNI: \( RF(T_1, T_2) \leq 2 \ NNI(T_1, T_2) \)

trivial splits, \( \{AB\}, \{CD, DE, EF\}, \{EF\}, \{AB, E, F\}, \{CD\}, \{AB, E, F\} \)

trivial splits, \( \{AB\}, \{CD, DE, EF\}, \{EF\}, \{AB, C, D\}, \{AB, D\}, \{CE, EF\} \)
Robinson-Foulds Distance

**Definition**

\[ RF(T_1, T_2) = \#\text{splits/clusters occurring in exactly one of } T_1 \text{ and } T_2 \]

= edge-contraction distance a common tree

**Note:** observe relation to NNI: \[ RF(T_1, T_2) \leq 2 \times \text{NNI}(T_1, T_2) \]
Robinson-Foulds Distance

**Definition**

\[ RF(T_1, T_2) = \#\text{splits/clusters occurring in exactly one of } T_1 \text{ and } T_2 \]

= edge-contraction distance a common tree

**Note:** observe relation to NNI: \( RF(T_1, T_2) \leq 2 \ NNI(T_1, T_2) \)

**Note:** splits correspond to clusters when rooted at last leaf

---

**Day's Algorithm (common clusters in } O(n)\) [Day'85]**

1. relabel all leaves s.t. leaves continuous in \( T_1 \)
2. each vertex in \( T_1 \) knows:
   - \( L \): smallest leaf in cluster
   - \( R \): largest leaf in cluster
   \( \leadsto T_1 \)'s clusters in table \([L, R]\) \((O(n)\) sparse set/perfect hashing\)
3. each vertex in \( T_2 \) knows \( L \neq R \neq \text{size } N \) of its cluster
4. each vertex in \( T_2 \) only checks for \([L, R]\) if \( R - L = N - 1 \)
   \( \text{lookup in } T_1 \)'s table in } O(1) \text{ (average) time)
Day’s Algorithm (common clusters in $O(n)$) [Day’85]

1. relabel all leaves s.t. leaves continuous in $T_1$
2. each vertex in $T_1$ knows:
   - $L$: smallest leaf in cluster
   - $R$: largest leaf in cluster
   - $\rightarrow$ $T_1$’s clusters in table $[L, R]$ ($O(n)$ sparse set/perfect hashing)
3. each vertex in $T_2$ knows $L \neq R \neq$ size $N$ of its cluster
4. each vertex in $T_2$ only checks for $[L, R]$ if $R - L = N - 1$
   (lookup in $T_1$’s table in $O(1)$ (average) time)
Robinson-Foulds Distance

Day's Algorithm (common clusters in $O(n)$) [Day'85]

1. relabel all leaves s.t. leaves continuous in $T_1$
2. each vertex in $T_1$ knows:
   - $L$: smallest leaf in cluster
   - $R$: largest leaf in cluster
   - $T_1$'s clusters in table $[L, R]$ ($O(n)$ sparse set/perfect hashing)
3. each vertex in $T_2$ knows $L \neq R \neq$ size $N$ of its cluster
4. each vertex in $T_2$ only checks for $[L, R]$ if $R - L = N - 1$
   (lookup in $T_1$'s table in $O(1)$ (average) time)
Robinson-Foulds Distance

Day's Algorithm (common clusters in $O(n)$) \cite{Day85}

1. relabel all leaves s.t. leaves continuous in $T_1$
2. each vertex in $T_1$ knows:
   - $L$: smallest leaf in cluster
   - $R$: largest leaf in cluster
   \rarr; $T_1$'s clusters in table $[L, R]$ ($O(n)$ sparse set/perfect hashing)
3. each vertex in $T_2$ knows $L \neq R \neq$ size $N$ of its cluster
4. each vertex in $T_2$ only checks for $[L, R]$ if $R - L = N - 1$
   (lookup in $T_1$'s table in $O(1)$ (average) time)
Robinson-Foulds Distance

Day's Algorithm (common clusters in $O(n)$) [Day'85]

1. relabel all leaves s.t. leaves continuous in $T_1$
2. each vertex in $T_1$ knows:
   - $L$: smallest leaf in cluster
   - $R$: largest leaf in cluster
   $\Rightarrow T_1$’s clusters in table $[L, R]$ ($O(n)$ sparse set/perfect hashing)
3. each vertex in $T_2$ knows $L \neq R \neq$ size $N$ of its cluster
4. each vertex in $T_2$ only checks for $[L, R]$ if $R - L = N - 1$
   (lookup in $T_1$’s table in $O(1)$ (average) time)
Quartet/Triplet Distance

**Definition**

\[ Q/T(T_1, T_2) = \#\text{quartets/triplets occur. in exactly one of } T_1 \text{ and } T_2 \]
Quartet/Triplet Distance

**Definition**

\[ Q/T(T_1, T_2) = \#\text{quartets/triplets occur. in exactly one of } T_1 \text{ and } T_2 \]

**computing Q-distance (binary trees)** [Bryant et al.’00]

1. each edge \( uv \) has 4 sets (2 clusters for each of \( u \neq v \))
2. quartet \( AB|CD \) "belongs" to edge \( e \) if \( e \) splits \( AB|CD \) and \( e \) touches \( AB\)-path or \( CD\)-path, \( \Rightarrow \) each split is owned exactly once
3. \( \forall uv \in T_1 \neq ab \in T_2: \) intersect 4 sets of \( uv \) with split of \( ab \) in \( T_2 \)
4. sizes of intersections can be precomputed bottom-up in \( O(n^2) \) time

**State of the Art**

- count conflict quartets/triplets \( \Rightarrow O(n \log n) \) time [Brodal et al.’13]
- enumerate conflict quartets \( \Rightarrow O(n^2 + d) \) [Bryant et al.’00]
- enumerate conflict triplets \( \Rightarrow O(n + d) \) [Weller’17]
Quartet/Triplet Distance

**Definition**

$Q/T(T_1, T_2) = \#\text{quartets/triplets occur. in exactly one of } T_1 \text{ and } T_2$

**Computing Q-distance (Binary trees)** [Bryant et al.’00]

1. each edge uv has 4 sets (2 clusters for each of u ≠ v)
2. quartet AB|CD "belongs" to edge e if e splits AB|CD ≠ e touches AB-path or CD-path ⇨ each split is owned exactly once
3. $\forall$ uv∈ $T_1$ ≠ ab∈ $T_2$: intersect 4 sets of uv with split of ab in $T_2$
4. sizes of intersections can be precomputed bottom-up in $O(n^2)$ time

**State of the Art**

- count conflict quartets/triplets ⇻ $O(n \log n)$ time [Brodal et al.'13]
- enumerate conflict quartets ⇻ $O(n^2 + d)$ [Bryant et al.'00]
- enumerate conflict tripletts ⇻ $O(n + d)$ [Weller'17]
Phylogenetic Networks

Observation
Trees cannot capture hybridization
Phylogenetic Networks

**Observation**

Trees cannot capture hybridization \( \leadsto \) phylogenetic network
Phylogenetic Networks

Observation
Trees cannot capture hybridization \(\Rightarrow\) phylogenetic network

Definition

- **evolutionary network** \(N =\) rooted DAG, leaves labeled (taxa)
- **reticulations** \(R =\) vertices of in-degree \(\geq 2\)
- **binary** = all inner vertices degree 3
- **block** = component without cut-vertex
- **display** \(T =\) subdivision of \(T\) is a subgraph
Phylogenetic Networks

**Observation**
Trees cannot capture hybridization $\Rightarrow$ phylogenetic network

**Definition**
evolutionary network $N =$ rooted DAG, leaves labeled (taxa)
reticulations $R =$ vertices of in-degree $\geq 2$
binary = all inner vertices degree 3
block = component without cut-vertex
display $T =$ subdivision of $T$ is a subgraph
Split Networks

split = bipartition of set of taxa

splits $A|B \neq X|Y$ incompatible if both $A \neq B$ intersect both $X \neq Y$

Convex Hull Algorithm [Holland et al.,'04]

c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)
**Split Networks**

split = bipartition of set of taxa

splits A|B ≠ X|Y incompatible if both A ∩ B intersect both X ∩ Y

Convex Hull Algorithm [Holland et al.,'04]

c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)
Split Networks

split = bipartition of set of taxa

splits $A \cap B \neq X \cap Y$ incompatible if both $A \cap B$ intersect both $X \cap Y$

Convex Hull Algorithm [Holland et al.,'04]

c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)
Split Networks

split = bipartition of set of taxa
splits $A \mid B \neq X \mid Y$ incompatible if both $A \cap B$ intersect both $X \cap Y$

Convex Hull Algorithm [Holland et al.,’04]

c.f.: Neighbor Net [Bryant & Moulton,’03] (for circular splits)
Split Networks

split = bipartition of set of taxa
splits $A|B \neq X|Y$ incompatible if both $A \cap B$ intersect both $X \cap Y$

Convex Hull Algorithm [Holland et al., '04]

c.f.: Neighbor Net [Bryant & Moulton, '03]
(for circular splits)
Split Networks

split = bipartition of set of taxa
splits $A \backslash B \neq X \backslash Y$ incompatible if both $A \backslash B$ intersect both $X \backslash Y$

Convex Hull Algorithm [Holland et al., ’04]

c.f.: Neighbor Net [Bryant & Moulton, ’03]
(for circular splits)
Split Networks

split = bipartition of set of taxa

splits $A|B \neq X|Y$ incompatible if both $A \cap B$ intersect both $X \cap Y$

Convex Hull Algorithm [Holland et al.,'04]

c.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)
Split Networks

split = bipartition of set of taxa
splits $A \upharpoonright B \neq X \upharpoonright Y$ incompatible if both $A \upharpoonright B$ intersect both $X \upharpoonright Y$

Convex Hull Algorithm [Holland et al., ‘04]

c.f.: Neighbor Net [Bryant & Moulton, ‘03]
(for circular splits)
Split Networks

split = bipartition of set of taxa
splits A|B & X|Y incompatible if both A ∩ B intersect both X ∩ Y

Convex Hull Algorithm [Holland et al.,'04]

C.f.: Neighbor Net [Bryant & Moulton,'03]
(for circular splits)
Consensus Split Networks

**Strategy**

1. list all splits of all input trees
2. extend splits to full taxa using "Z-closure"
3. build consensus
Consensus Split Networks

**Strategy**
1. list all splits of all input trees
2. extend splits to full taxa using "Z-closure"
3. build consensus

**Experimental Study** – 106 gene trees (yeast)

[Rokas et al.’03, Holland et al.’04]
Consensus Split Networks

**Strategy**

1. list all splits of all input trees
2. extend splits to full taxa using "Z-closure"
3. build consensus

**Experimental Study** – 106 gene trees (yeast)

[Rokas et al.’03, Holland et al.’04]
Consensus Split Networks

**Strategy**
1. list all splits of all input trees
2. extend splits to full taxa using "Z-closure"
3. build consensus

**Experimental Study** – 106 gene trees (yeast)

[Rokas et al.’03, Holland et al.’04]
Consensus Split Networks

**Strategy**

1. list all splits of all input trees
2. extend splits to full taxa using "Z-closure"
3. build consensus

**Experimental Study** – 106 gene trees (yeast)

[Rokas et al.’03, Holland et al.’04]
Rooted Network Reconstruction

**Observation**

rooted network: cluster of $u \subseteq$ cluster of $v \iff u \leq v$

$\Rightarrow$ rooted network is hasse diagram of its clusters
Rooted Network Reconstruction

Observation
rooted network: cluster of u ⊆ cluster of v ↔ u ≤ v
⇒ rooted network is hasse diagram of its clusters

Example
\{a,b,c,d\}, \{c,d,e,f,g,h\}, \{c,d,e,f,g\}, \{e,f,g,h\}, \{c,d,e\}, \{e,f,g\}, \{a,b\}, \{c,d\}, \{f,g\}
Observation

rooted network: cluster of $u \subseteq$ cluster of $v \iff u \leq v$

\Rightarrow rooted network is Hasse diagram of its clusters

Example

$\{a,b,c,d\}, \{c,d,e,f,g,h\}, \{c,d,e,f,g\}, \{e,f,g,h\}, \{c,d,e\}, \{e,f,g\}, \{a,b\}, \{c,d\}, \{f,g\}$
Rooted Network Reconstruction

Observation

rooted network: cluster of $u \subseteq$ cluster of $v \iff u \leq v$

$\implies$ rooted network is Hasse diagram of its clusters

Example

$\{a,b,c,d\}, \{c,d,e,f,g\}, \{c,d,e,f,g,h\}, \{c,d,e\}, \{e,f,g\}, \{a,b\}, \{c,d\}, \{f,g\}$

![Hasse diagram of clusters](image)
Rooted Network Reconstruction

**Observation**
rooted network: cluster of $u \subseteq$ cluster of $v \iff u \leq v$

$\Rightarrow$ rooted network is *hasse diagram* of its clusters

**Example**

\{
\{a, b, c, d\}, \{c, d, e, f, g, h\}, \{c, d, e, f, g\}, \{e, f, g, h\}, \{c, d, e\}, \{e, f, g\}, \{a, b\}, \{c, d\}, \{f, g\}\}

c.f. "cluster popping" [Huson & Rupp, '08]
Rooted Network Reconstruction

**Observation**

rooted network: cluster of $u \subseteq$ cluster of $v \iff u \leq v$

$\iff$ rooted network is hasse diagram of its clusters

**Problem**

may produce more reticulations than necessary to explain the data
Rooted Network Reconstruction

**Observation**

rooted network: cluster of $u \subseteq$ cluster of $v \iff u \leq v$

$\Rightarrow$ rooted network is hasse diagram of its clusters

**Problem**

may produce more reticulations than necessary to explain the data

**Hybridization Number**

Input: set of trees $T$, int $k$

Question: Is there a network with $\leq k$ reticulations displaying all trees in $T$?
Rooted Network Reconstruction

**Observation**

rooted network: \( u \subseteq \text{cluster of } v \iff u \leq v \)

\( \Rightarrow \) rooted network is **hasse diagram** of its clusters

**Problem**

may produce more reticulations than necessary to explain the data

**Hybridization Number**

**Input:** set of trees \( T \), int \( k \)

**Question:** Is there a network with \( \leq k \) reticulations displaying all trees in \( T \)?

\( \Rightarrow \) **NP-hard** for 2 trees [Bordewich & Semple,'07]
Rooted Network Reconstruction

**Observation**

rooted network: cluster of $u \subseteq$ cluster of $v \iff u \leq v$

$\Rightarrow$ rooted network is hasse diagram of its clusters

**Problem**

may produce more reticulations than necessary to explain the data

**Hybridization Number**

**Input:** set of trees $T$, int $k$

**Question:** Is there a network with $\leq k$ reticulations displaying all trees in $T$?

$\Rightarrow$ NP-hard for 2 trees [Bordewich & Semple,’07]

**Note:** $HN(T_1,T_2) = \text{max. acyclic agreement forest} - 1$ [Baroni et al.,’05]
Networks Display Trees

Observation

A network may display up to $2^{|R|}$ trees.
Networks Display Trees

Observation

A network may display up to $2^{\left|R\right|}$ trees.
Networks Display Trees

Observation

A network may display up to $2^{\lfloor R \rfloor}$ trees. But: how to decide if a given tree is displayed?
Networks Display Trees

Tree Containment

Input: a network \( N \), a tree \( T \)

Question: Does \( N \) display \( T \)?
Networks Display Trees

Tree Containment

Input: a network $N$, a tree $T$

Question: Does $N$ display $T$?

$\Rightarrow$ NP-hard (from Disjoint Paths) [Kanj et al.'08]
Networks Display Trees

Tree Containment

Input: a network $N$, a tree $T$

Question: Does $N$ display $T$?

\[ \Rightarrow \text{NP-hard (from Disjoint Paths)} \quad [\text{Kanj et al.'08}] \]

Note: linear time on reticulation visible $N$ \[ [\text{Gunawan,'18}][\text{Weller,'18}] \]
Networks Display Trees

Tree Containment

Input: a network $N$, a tree $T$

Question: Does $N$ display $T$?

$\rightsquigarrow$ NP-hard (from Disjoint Paths) [Kanj et al.'08]

Note: linear time on reticulation visible $N$ [Gunawan,'18][Weller,'18]
Small Taxonomy of Network Classes

c.f. "Who is Who in Phylogenetic Networks" (http://phylnet.univ-mlv.fr/)
Thanks & Enjoy Part III