Quartet distance
Different trees for same set of species
Quartets and quartet distance

**Quartet**: Four named species in an unrooted tree

**Quartet topology**: The topology of the quartet induced by the tree

**Quartet distance**: The number of quartets that *don't* have the same topology in the two trees
Quartets and quartet distance
Quartets and quartet distance

A B C D
B D

A C D
B

A B C D

A B
C D

A B
c

A B
C D
e

A B C
e

A B C
e

A B
C D
e
Quartets and quartet distance
Quartets and quartet distance

Quartet ABCD is present.
Quartet ABCE is present.
Quartet ABDE is present.
Quartet ABD is missing.
Quartet ABE is missing.
Quartet BDE is present.
Quartets and quartet distance

\begin{align*}
\text{ABC} & \quad \text{ABCD} \quad \text{ABCE} \quad \text{ABDE} \quad \text{ACDE} \\
\text{BDE} & \quad \text{ABCDE} \quad \text{BCDE} \quad \text{BDE} \quad \text{CDE}
\end{align*}
Quartets and quartet distance

A
B
C
D
E

ABCD
ABCE
ABDE
ACDE
BCDE

A
B
C
D
E

A
B
C
D
E

A
B
C
D
E

A
B
C
D
E

A
B
C
D
E

A
B
C
D
E

A
B
C
D
E

A
B
C
D
E

A
B
C
D
E

A
B
C
D
E
Quartets and quartet distance

Quartet distance \(= \binom{5,4}{} - 3 = 5 - 3 = 2\)
Previous work

G. Estabrook, F. McMorris, and C. Meacham.

W. Day and C. R. Doucette.
Expected behaviour of quartet distances between undirected tree.
Proc. 18th International Numerical Taxonomy Conference, 1984

C. R. Doucette
An efficient algorithm to compute quartet dissimilarity measures. Bachelor of Science (Honours) Dissertation. Memorial University of Newfoundland, 1985

M. Steel and D. Penny.
Distribution of tree comparison metrics—some new results.

D. Bryant, J. Tsang, P. E. Kearney, and M. Li.
Computing the quartet distance between evolutionary trees.

G. S. Brodal, R. Fagerberg, and C. N. S. Pedersen
Computing the quartet distance in time $O(n \log n)$.
Details about the \( O(n^2) \) algorithm
Computing the quartet distance

We associate quartet ab|cd in T and T' with the oriented edge e s.t.:
Computing the quartet distance

Algorithm: For every pair of oriented edges \((e, e')\) in \(T\) and \(T'\), we count the number of quartets associated to \(e\) in \(T\), which is also associated to \(e'\) in \(T'\)...

\[
\text{Count}(e, e') = |\text{A}_1 \text{A}_1' \cdot \text{B}_1 \text{B}_1'| \cdot \left( \frac{|C \cap C'|}{2} \right) + |\text{A}_2 \text{B}_2' \cdot \text{B}_1 \text{A}_1'| \cdot \left( \frac{|C \cap C'|}{2} \right)
\]
Computing $|V \cap V'|$

If we assume that $1, 2, 1', 2'$ are non-empty, i.e. $e$ and $e'$ are internal edges in $T$ and $T'$, we get:

$$V \cap V' = (1 \cup 2) \cap (1' \cup 2')$$

$$= (1 \cup 2) \cap 1' \cup (1 \cup 2) \cap 2'$$

$$= (1 \cap 1') \cup (2 \cap 1') \cup (1 \cap 2') \cup (2 \cap 2')$$
Computing $|V \cap V'|$

Since $1 \cap 2 = 1' \cap 2' = \emptyset$, we get:

$$|V \cap V'| = |1 \cap 1'| + |1 \cap 2'| + |2 \cap 1'| + |2 \cap 2'|$$

... a simple recursive formulation; the table $T[e,e']$ of size $O(n^2)$ can be computed in time $O(n^2)$ using dynamic programming....
Computing $|V \cap V'|$

Special case:

$T[e,e'] = |V \cap V'| = T[e,e'_1] + T[e,e'_2]$

Base case:

$T[e,e'] = \begin{cases} 
1 & \text{if } v = v' \\
0 & \text{otherwise}
\end{cases}$
Computing the quartet distance

Algorithm: For every pair of oriented edges \((e, e')\) in \(T\) and \(T'\), we count the number of quartets assoc. to \(e\) in \(T\), which is also assoc. to \(e'\) in \(T'\)...

\[
\text{Count}(e, e') := |\text{A} \cap \text{A'}| \cdot |\text{B} \cap \text{B'}| \cdot \left( \frac{|\text{C} \cap \text{C'}|}{2} \right) + \\
|\text{A} \cap \text{B'}| \cdot |\text{B} \cap \text{A'}| \cdot \left( \frac{|\text{C} \cap \text{C'}|}{2} \right)
\]
Computing the quartet distance

Observe: every shared quartet abc1d, is counted twice...

\[\begin{align*}
\text{(1)} & \quad \text{Diagram 1} \\
\text{(2)} & \quad \text{Diagram 2}
\end{align*}\]
Computing the quartet distance

\[ \text{Quartet Dist.} = \binom{n}{4} - \frac{1}{2} \sum_{e \in E} \text{Count}(e, e') \]

Running time?

\( O(n^2) \)
Comparing partially resolved trees
The ideas of Tsang's $O(n^2)$ time algorithm for fully resolved tree can be generalized. An immediate generalization yields and $O(n^2 d^4)$ time algorithm. Doucette's work from 1985 can be used for a $O(n^3)$ time algorithm ...
Idea: Iterate over all pairs of edges (or nodes) in the two trees, and count for each pair how many associated quartets are shared. The problem is to define “associated” such that each quartet is counted as most once ...

### Different solutions

<table>
<thead>
<tr>
<th>Type</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center based</td>
<td>$O(n^3)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Edge based</td>
<td>$O(</td>
<td>V</td>
</tr>
<tr>
<td>Edge based</td>
<td>$O(n+</td>
<td>V</td>
</tr>
<tr>
<td>Node based</td>
<td>$O(n+</td>
<td>V</td>
</tr>
</tbody>
</table>

- $n$ is the number of species/leaves
- $|V|, |V'|$ are number of internal nodes in T and T'
- $id, id'$ are internal degree of T and T'

“worst case”-tree $|V|=id=O(n)$
Algorithms for partially resolved trees

A sub-cubic time algorithm for computing the quartet distance between two general trees (2011).

Thomas Mailund, Jesper Nielsen and Christian N.S. Pedersen

In this paper we develop an $O(n^{2+\alpha})$, where $\alpha = (\omega-1)/2$ and $O(n^\omega)$ is the time it takes to multiply two $n \times n$ matrices. Using the Coppersmith-Winograd algorithm, where $\omega = 2.376$, this yields a running time of $O(n^{2.688})$. The running time is thus independent of the degrees of the inner nodes of the input trees, and this is the first sub-cubic time algorithm with this property.

<table>
<thead>
<tr>
<th>Type</th>
<th>Time</th>
<th>Space</th>
</tr>
</thead>
<tbody>
<tr>
<td>Center based</td>
<td>$O(n^3)$</td>
<td>$O(n)$</td>
</tr>
<tr>
<td>Edge based</td>
<td>$O(</td>
<td>V</td>
</tr>
<tr>
<td>Edge based</td>
<td>$O(n+</td>
<td>V</td>
</tr>
<tr>
<td>Node based</td>
<td>$O(n+</td>
<td>V</td>
</tr>
</tbody>
</table>

- $n$ is the number of species/leaves
- $|V|$, $|V'|$ are number of internal nodes in $T$ and $T'$
- $id$, $id'$ are internal degree of $T$ and $T'$

"worst case"-tree

$|V|=id=O(n)$
Algorithms for partially resolved trees

A sub-cubic time algorithm for computing the quartet distance between two general trees (2011).

Thomas Mailund, Jesper Nielsen and Christian N.S. Pedersen

In this paper we develop an $O(n^{2+\alpha})$, where $\alpha = (\omega - 1)/2$ and $O(n^\omega)$ is the time it takes to multiply two $n \times n$ matrices. Using the Coppersmith-Winograd algorithm, where $\omega = 2.376$, this yields a running time of $O(n^{2.688})$. The running time is thus independent of the degrees of the inner nodes of the input trees, and this is the first sub-cubic time algorithm with this property.

A practical $O(n \log^2 n)$ time algorithm for computing the triplet distance on binary trees.

Andreas Sand, Gerth Brodal, Rolf Fagerberg, Christian N.S. Pedersen and Thomas Mailund

In this paper we develop an $O(n \log^2 n)$ time algorithm for computing the triplet distance between binary trees.

- $n$ is the number of species/leaves
- $|V|, |V'|$ are number of internal nodes in $T$ and $T'$
- $\text{id}, \text{id}'$ are internal degree of $T$ and $T'$

"worst case"-tree $|V|=\text{id}=O(n)$
Algorithms for partially resolved trees

A sub-cubic time algorithm for computing the quartet distance between two general trees (2011).

Thomas Mailund, Jesper Nielsen and Christian N.S. Pedersen

In this paper we develop an $O(n^{2+\alpha})$, where $\alpha = (\omega - 1)/2$ and $O(n^\omega)$ is the time it takes to multiply two $n \times n$ matrices. Using the Coppersmith-Winograd algorithm, where $\omega = 2.376$, this yields a running time of $O(n^{2.688})$. The running time is thus independent of the degrees of the inner nodes of the input trees, and this is the first sub-cubic time algorithm with this property.

A practical $O(n\log^2 n)$ time algorithm for computing the triplet distance on binary trees.

Andreas Sand, Gerth Brodal, Rolf Fagerberg, Christian N.S. Pedersen and Thomas Mailund

Efficient Algorithms for Computing the Triplet and Quartet Distance Between Trees of Arbitrary Degree (2012).

Gerth Brodal, Rolf Fagerberg, Thomas Mailund, Christian N.S. Pedersen and Andreas Sand

In this paper we develop an $O(n \log n)$ time algorithm for computing the triplet distance between trees of arbitrary degree, and a $O(dn \log n)$ time algorithm for computing the quartet distance between trees of degree $d$. 

- $n$ is the number of species/leaves
- $|V|$, $|V'|$ are number of internal nodes in $T$ and $T'$
- id, id' are internal degree of $T$ and $T'$
Other kinds of distances?
Subtree-Prune-Regraft

$\text{SPR-Dist}(T_1, T_2) =$

Minimum number of SPR-op's that transform $T_1$ into $T_2$

NP-complete for unrooted binary trees allowing unrestricted SPR-operations ...
Nearest-Neighbor interchange

NNI-Dist($T_1, T_2$) = Minimum number of NNI-op's that transform $T_1$ into $T_2$

NP-complete for unrooted binary trees ...