

Computing the all-pairs quartet distance on a set of evolutionary trees

Martin Stissing ¹

Thomas Mailund ^{1,2}

Christian Nørgaard Storm Pedersen ¹

Gerth Stølting Brodal ¹

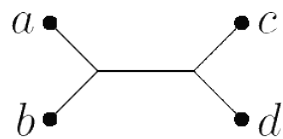
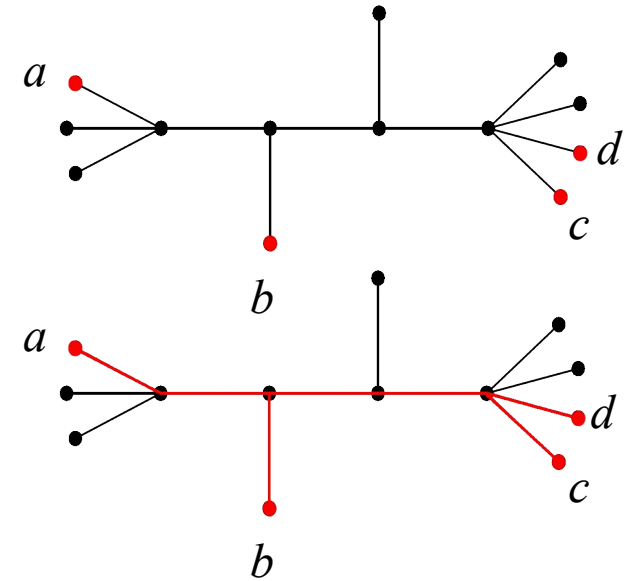
Rolf Fagerberg ³

(1) University of Aarhus, (2) Oxford University, (3) University of Southern Denmark

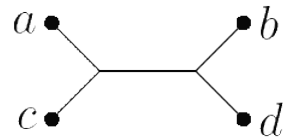
Quartets and quartet distance

Quartet: Four named species in an unrooted tree

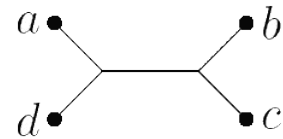
Quartet topology: The topology of the quartet induced by the tree



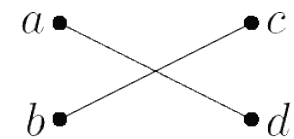
butterfly quartet



butterfly quartet



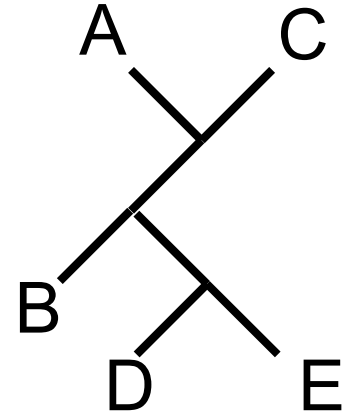
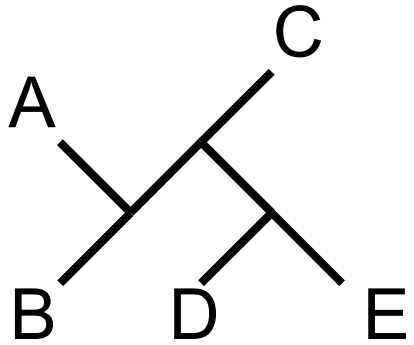
butterfly quartet



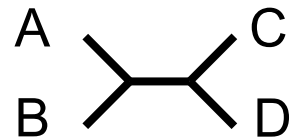
star quartet

Quartet distance: The number of quartets that differs in topology between the two trees

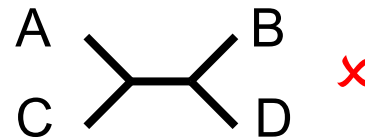
Quartets and quartet distance



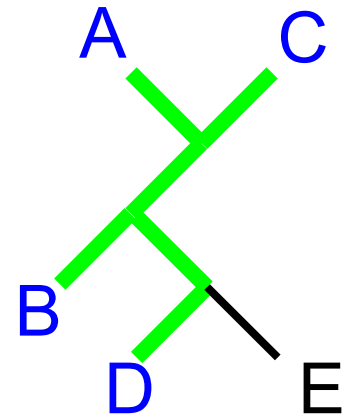
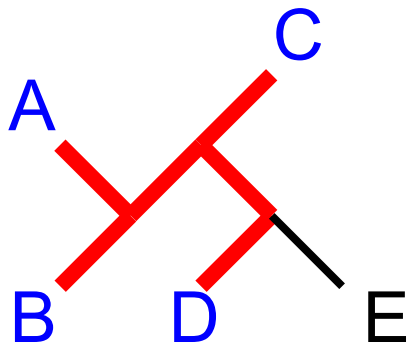
Quartets and quartet distance



ABCD



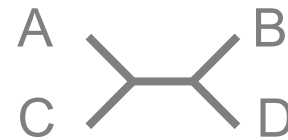
x



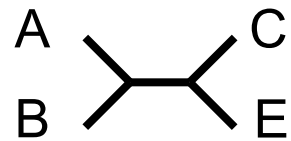
Quartets and quartet distance



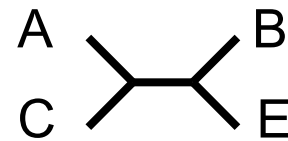
ABCD



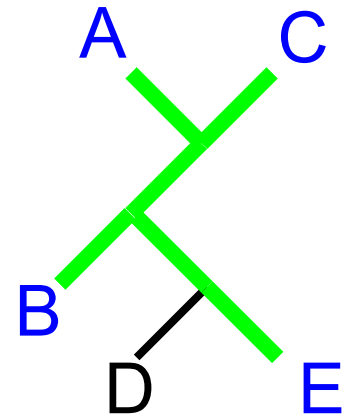
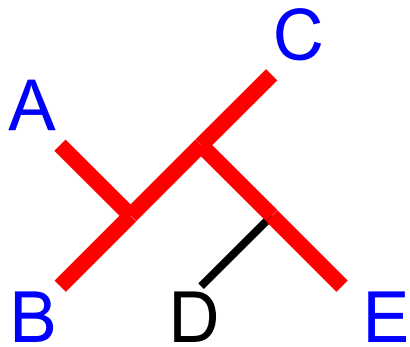
x



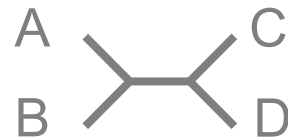
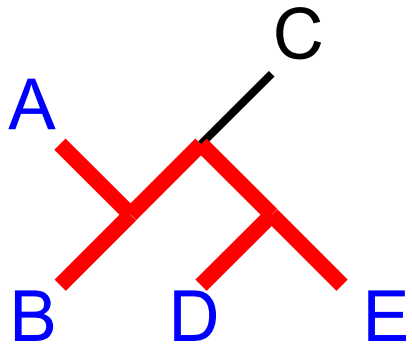
ABCE



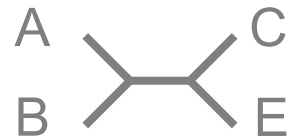
x



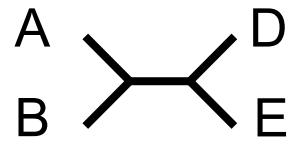
Quartets and quartet distance



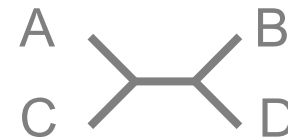
ABCD



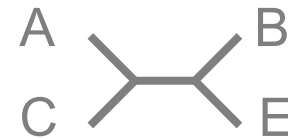
ABCE



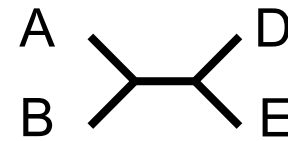
ABDE



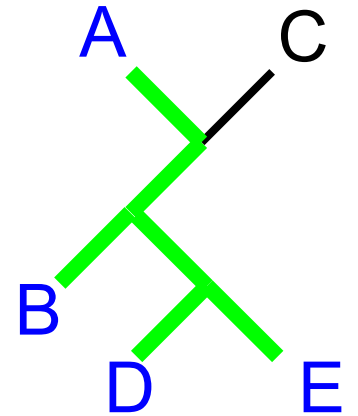
x



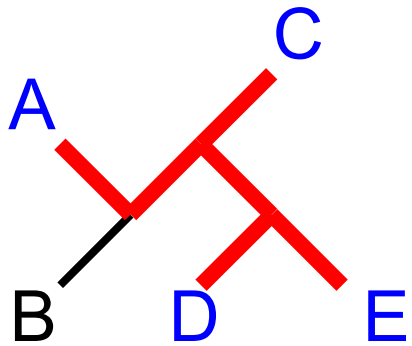
x



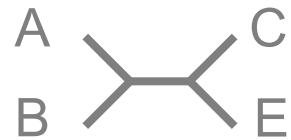
✓



Quartets and quartet distance



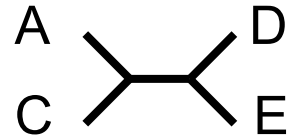
ABCD



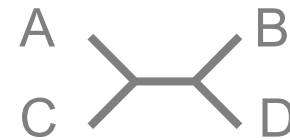
ABCE



ABDE



ACDE



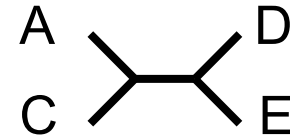
x



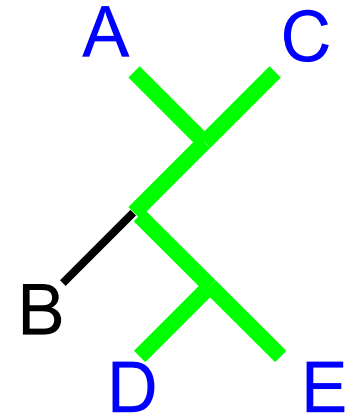
x



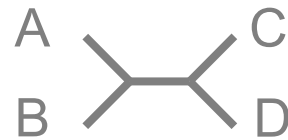
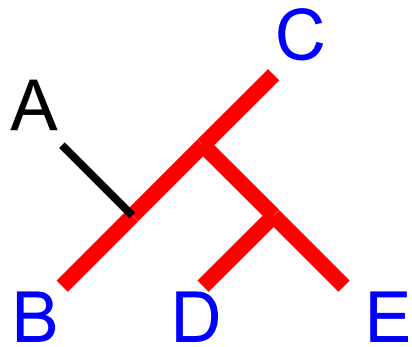
✓



✓



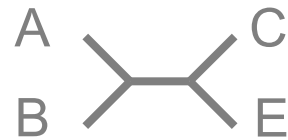
Quartets and quartet distance



ABCD



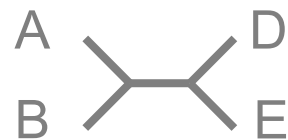
x



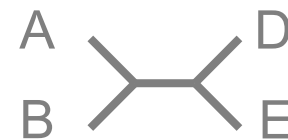
ABCE



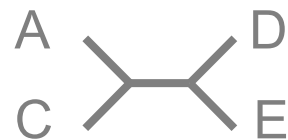
x



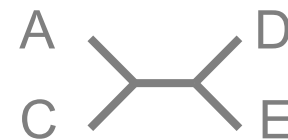
ABDE



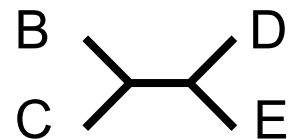
✓



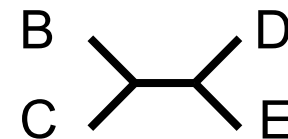
ACDE



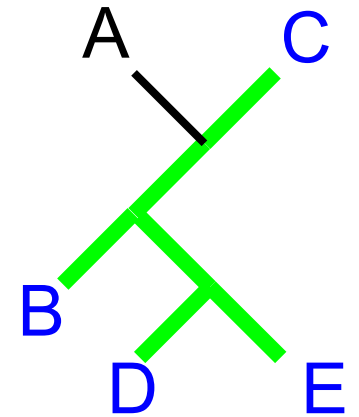
✓



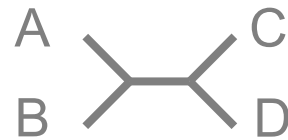
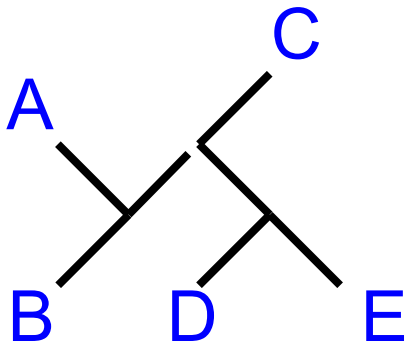
BCDE



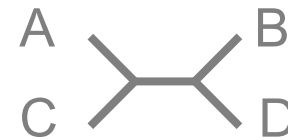
✓



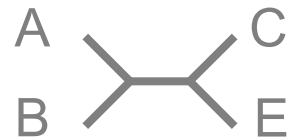
Quartets and quartet distance



ABCD



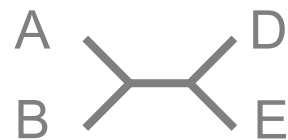
x



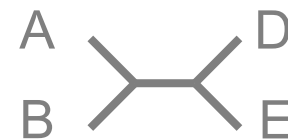
ABCE



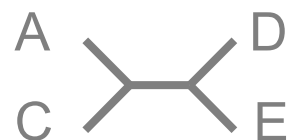
x



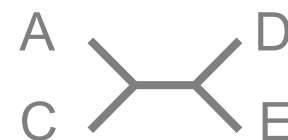
ABDE



✓



ACDE



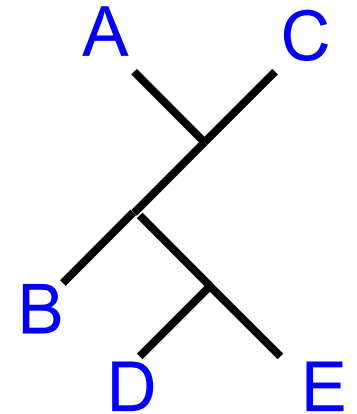
✓



BCDE



✓



Quartet distance = $\text{binom}(5,4) - 3 = 5 - 3 = 2$

Previous work

G. Estabrook, F. McMorris, and C. Meacham.
Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. *Systematic Zoology.*, 27:27-33, 1978.

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

$O(n^3)$

M. Steel and D. Penny.
Distribution of tree comparison metrics—some new results.
Systematic Biology, 42(2):126–141, 1993.

D. Bryant, J. Tsang, P. E. Kearney, and M. Li.
Computing the quartet distance between evolutionary trees.
Proc. 11th Symp. on Discrete Algorithms (SODA), 285–286, 2000.

$O(n^2)$

G. S. Brodal, R. Fagerberg, and C. N. S. Pedersen
Computing the quartet distance in time $O(n \log n)$.
Algorithmica, 38(2): 377-395, 2003.

$O(n \log^2 n)$
 $O(n \log n)$

Previous work

G. Estabrook, F. McMorris, and C. Meacham.
Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. *Systematic Zoology.*, 27:27-33, 1978.

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.

$O(n^3)$

Algorithms are for *fully resolved trees* (binary trees) only.

Quartet distance between binary trees seems easier to compute ...

Systematic Biology, 42(2):126–141, 1993

D. Bryant, J. Tsang, P. E. Kearney, and M. Li.
Computing the quartet distance between evolutionary trees.
Proc. 11th Symp. on Discrete Algorithms (SODA), 285–286, 2000.

$O(n^3)$

G. S. Brodal, R. Fagerberg, and C. N. S. Pedersen
Computing the quartet distance in time $O(n \log n)$.
Algorithmica, 38(2): 377-395, 2003.

$O(n \log^2 n)$

$O(n \log n)$

C. Christiansen, T. Mailund, C. N. S. Pedersen, and M. Randers
Algorithms for computing the quartet distance between trees of arbitrary degree
Proc of Workshop on Algorithms in Bioinformatics (WABI), 77-88, 2005 $O(n^3)$
 $O(d^2n^2)$

G. Estabrook
Comparing
four evolutionary trees
C. Christiansen, T. Mailund, C. N. S. Pedersen, M. Randers, and M. Stissing
Fast calculation of the quartet distance between trees of arbitrary degrees
Algorithms for Molecular Biology, 1(16), 2006 $O(dn^2)$

W. Day
Expected
Proc. 18
M. Stissing, C. N. S. Pedersen, T. Mailund, G. S. Brodal, and R. Fagerberg
Computing the quartet distance between evolutionary trees of bounded degree
Proc. of APBC, 2007 $O(d^9 n \log n)$

C. R. Doucette
An efficient algorithm to compute quartet dissimilarity measures. $O(n^3)$

Algorithms are for *fully resolved trees* (binary trees) only.

Quartet distance between binary trees seems easier to compute ...

Systematic Biology, 42(2):126–141, 1993.

D. Bryant, J. Tsang, P. E. Kearney, and M. Li. $O(n^3)$
Computing the quartet distance between evolutionary trees.
Proc. 11th Symp. on Discrete Algorithms (SODA), 285–286, 2000.

G. S. Brodal, R. Fagerberg, and C. N. S. Pedersen $O(n \log^2 n)$
Computing the quartet distance in time $O(n \log n)$. $O(n \log n)$
Algorithmica, 28(2): 377-395, 2003.

QDist

QDist: Implements the $O(n^2)$ and $O(n \log^2 n)$ time algorithm for computing the quartet distance between binary trees

File Edit View Go Bookmarks Tools Help
Tech News Mac Undersiving Windows WWW Gmail

QDist

[Usage] [Installation] [Contact]

The QDist package implements the $O(n \log^2 n)$ time method for computing the quartet distance between unrooted evolutionary trees. The method is described in the paper: [Computing the Quartet Distance Between Evolutionary Trees in Time \$O\(n \log^2 n\)\$](#) , G. Brodal, R. Fagerberg, and C.N.S. Pedersen, in Proceedings of the 12th International Symposium for Algorithms and Computation (ISAAC), 2001.

QDist is described in more detail in:

- [QDist—Quartet Distance Between Evolutionary Trees](#), T. Mailund and C.N.S. Pedersen, Bioinformatics, Vol. 20, No. 10, pp 1636-1637, 2004.

USAGE

The QDist program takes as input a set of trees in newick format over the same set of species and computes the quartet distance between them. Run `qdist --help` for more info.

INSTALLATION

From Source

The QDist package is written in C++. It should compile on any Unix like system. To install the QDist package [download the source code](#) and unpack it (`tar xzf qdist-a.b.c.tar.gz`, where *a.b.c* is the version number of qdist), then run configure and make in the subdirectory `qdist-a.b.c` created during unpacking. Read the INSTALL file for more info.

From Binary

Currently, the only binary distribution is in RPM packages. Other formats might appear, if I am asked to build them (and told how to...). To install, [download the file](#) and run `rpm -U qdist-version.arch.rpm`.

CONTACT

Thomas Mailund, <mailund@birc.au.dk>, Bioinformatics Research Center, University of Aarhus.

Time-stamp: "2006-01-26 21:58:43 mailund"

Done

BIOINFORMATICS APPLICATIONS NOTE Vol. 20 no. 10 2004, pages 1636–1637
doi:10.1093/bioinformatics/bth097

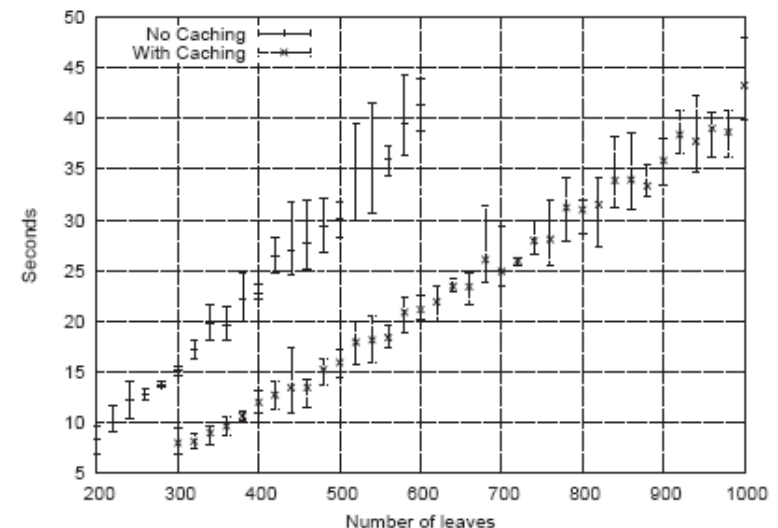


QDist—quartet distance between evolutionary trees

Thomas Mailund* and Christian N. S. Pedersen

Bioinformatics Research Center (BIRC), University of Aarhus, Ny Munkegade, Building 540, DK-8000 Århus C, Denmark

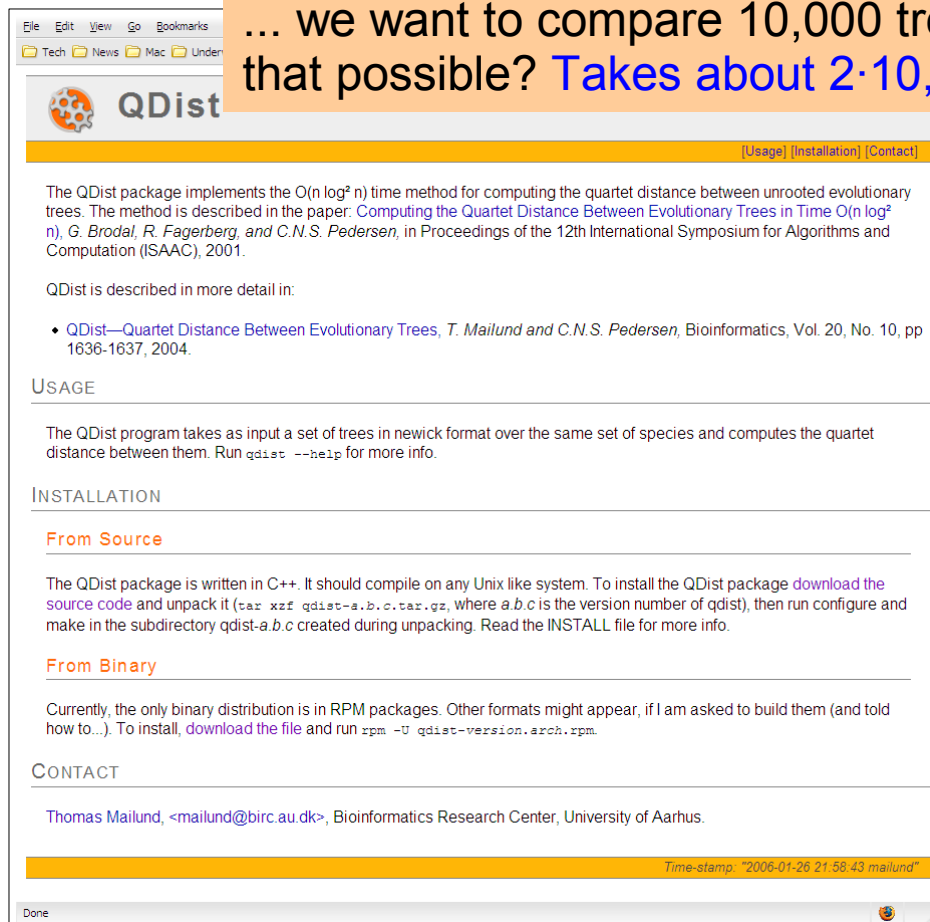
Received on November 18, 2003; revised and accepted on November 28, 2003
Advance Access publication February 12, 2004



QDist

QDist: Implements the $O(n^2)$ and $O(n \log^2 n)$ time algorithm for computing the quartet distance between binary trees

... we want to compare 10,000 trees of size 200 using Qdist, is that possible? Takes about $2 \cdot 10,000^2$ sec \approx 6 years ...



The screenshot shows the QDist website interface. At the top, there is a navigation bar with links for [Usage], [Installation], and [Contact]. The main content area includes a description of the QDist package, which implements the $O(n \log^2 n)$ time method for computing the quartet distance between unrooted evolutionary trees. It also provides a citation for the paper: "QDist—Quartet Distance Between Evolutionary Trees, T. Mailund and C.N.S. Pedersen, Bioinformatics, Vol. 20, No. 10, pp 1636-1637, 2004." Below this, there is a section for USAGE, which states that the QDist program takes as input a set of trees in newick format over the same set of species and computes the quartet distance between them. There is also an INSTALLATION section with sub-sections for "From Source" and "From Binary". The "From Source" section provides instructions on how to download the source code and unpack it. The "From Binary" section mentions that the only binary distribution is in RPM packages. At the bottom of the page, there is a CONTACT section with the email address mailund@birc.au.dk. A footer bar shows the time-stamp: "2006-01-26 21:58:43 mailund".

Vol. 20 no. 10 2004, pages 1636–1637
doi:10.1093/bioinformatics/bth097

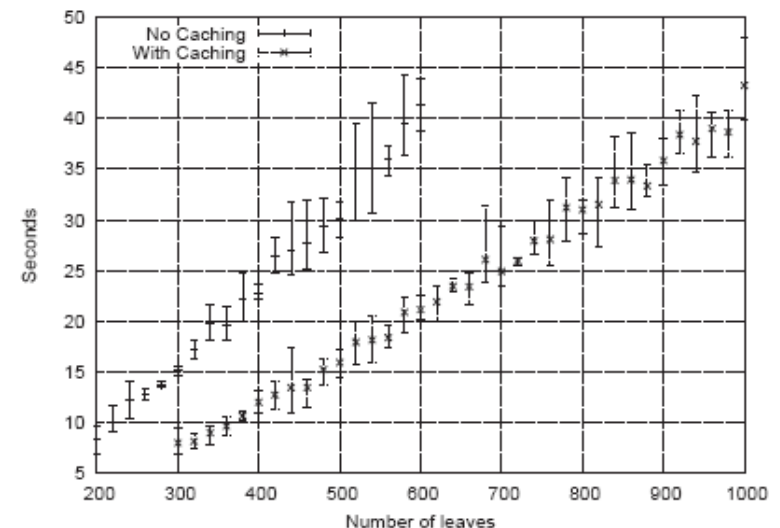


QDist—quartet distance between evolutionary trees

Thomas Mailund* and Christian N. S. Pedersen

Bioinformatics Research Center (BIRC), University of Aarhus, Ny Munkegade, Building 540, DK-8000 Århus C, Denmark

Received on November 18, 2003; revised and accepted on November 28, 2003
Advance Access publication February 12, 2004



<http://www.birc.au.dk/~mailund/qdist.html>

QDist

QDist: Implements the $O(n^2)$ and $O(n \log^2 n)$ time algorithm for computing the quartet distance between binary trees

... we want to compare 10,000 trees of size 200 using Qdist, is that possible? Takes about $2 \cdot 10,000^2$ sec \approx 6 years ...

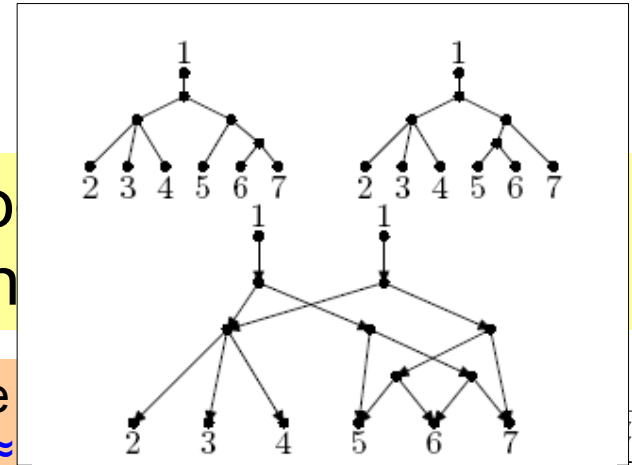
Immediate solution: Perform $O(k^2)$ comparisons between two tree using time $O(n \log^2 n)$ or $O(n^2)$ each ...

The screenshot shows the QDist website interface. The main content area contains the text: "The QDist package implements the quartet distance between binary trees. The method is described in: G. Brodal, R. Fagerberg, and S. Holm, 'Quartet Distance Between Binary Trees', Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Mathematics (SODA), SIAM, 2001." Below this, there is a section titled "QDist is described in more detail in:" with a link to "QDist—Quartet Distance Between Binary Trees, 1636-1637, 2004." The "USAGE" section states: "The QDist program takes as input two binary trees and outputs the quartet distance between them. Run 'qdist -h' for more information." The "INSTALLATION" section has two sub-sections: "From Source" and "From Binary". The "CONTACT" section lists "Thomas Mailund, <mailund@birc.au.dk>".

Two graphs are overlaid on the screenshot. The left graph, titled "Time usage for 2 random trees", plots "Time in seconds" (0 to 80) against "Number of leaves" (0 to 4000). It shows two curves: a black curve for $O(n^2)$ and a red curve for $O(n \log^2 n)$. The $O(n^2)$ curve rises steeply, while the $O(n \log^2 n)$ curve rises much more gradually. The right graph plots "Time in seconds" (0 to 3000) against "Number of leaves" (100 to 500). It shows multiple curves for different values of k (2, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100). The curves show that time increases with the number of leaves and the value of k . A legend on the left of the graph lists the k values and their corresponding colors. To the right of the graph, there is a small plot showing error bars for data points, with the x-axis ranging from 800 to 1000.

QDist

QDist: Implements the $O(n^2)$ and $O(n \log n)$ algorithms for computing the quartet distance between two trees.



for

... we want to compare 10,000 trees of size that possible? Takes about $2 \cdot 10,000^2$ sec \approx

Immediate solution: Perform $O(k^2)$ comparisons between two tree using time $O(n \log^2 n)$ or $O(n^2)$ each ...

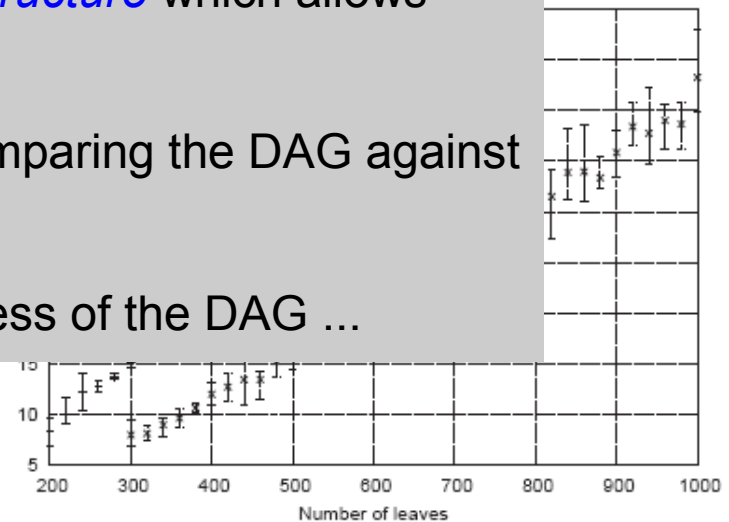
Faster solution: Exploit similarities between the input trees by merging them into a *single DAG-structure* which allows “common subtrees” to be shared.

All comparisons are performed by comparing the DAG against the input trees, or against itself.

The speed-up depends on compactness of the DAG ...

The screenshot shows the QDist website with the following content:

- QDist** logo and title.
- Description: "The QDist package implements the quartet distance between two trees. The method is described in: G. Brodal, R. Fagerberg, and T. Mailund, 'Quartet Distance Computation (ISAAC)', 2004."
- Usage: "The QDist program takes two trees as input and computes the distance between them. For more information, see the manual."
- Installation: "From Source" and "From Binary" sections.
- Contact: "Thomas Mailund, <mailund@birc.au.dk>, Bioinformatics Research Center, University of Aarhus."
- Footer: "Time-stamp: '2006-01-26 21:58:43 mailund'" and "Done".

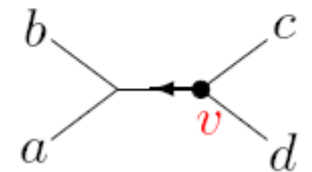
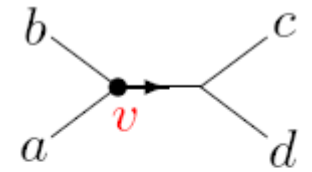


<http://www.birc.au.dk/~mailund/qdist.html>

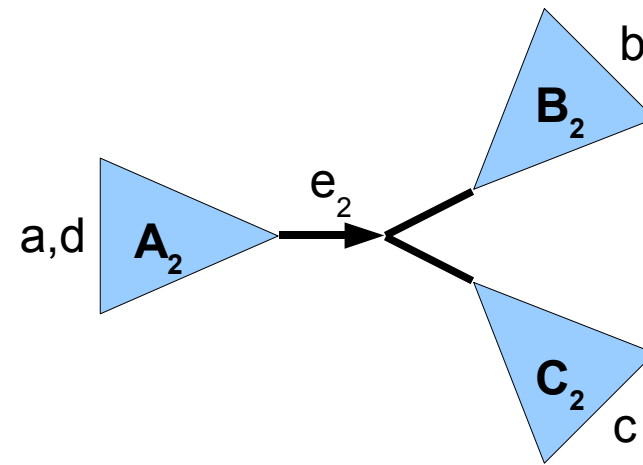
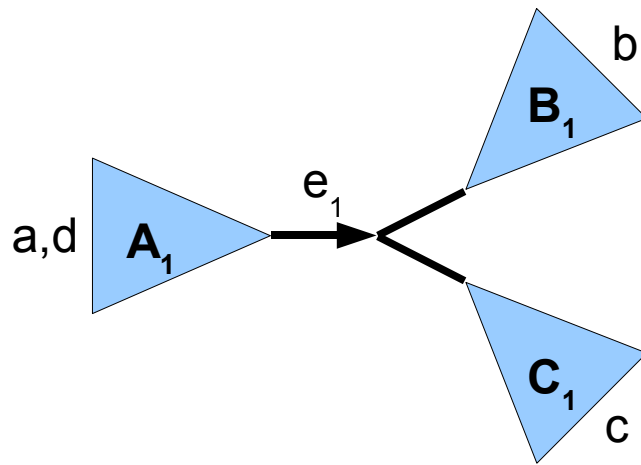
Counting shared (directed) quartets

Idea: Iterate over all pairs of edges in the two trees, and count for each pair how many *associated quartets* are shared. The problem is to define “associated” such each quartet is counted as most once ...

Directed quartets



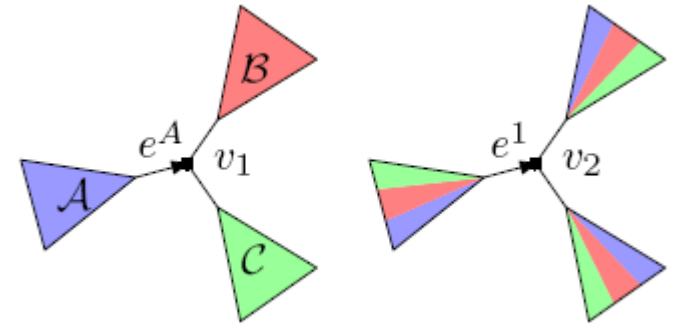
$$\binom{|A_1 \cap A_2|}{2} \cdot (|B_1 \cap B_2| \cdot |C_1 \cap C_2| + |B_1 \cap C_2| \cdot |C_1 \cap B_2|)$$



$$\binom{n}{4} - \frac{1}{2} \cdot \text{“number of identical quartets in } T_1 \text{ and } T_2\text{”}$$

Counting using colouring

Idea: Alternatively, colour leaves according to nodes in one tree and count compatible colourings in the other



$$\text{count}(e^A, e^1) = \binom{a(1)}{2} \cdot (b(2) \cdot c(3) + b(3) \cdot c(2))$$

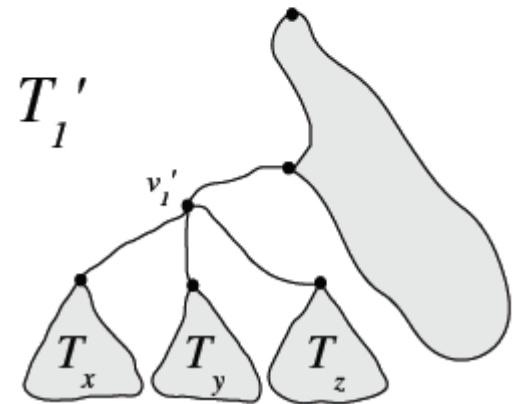
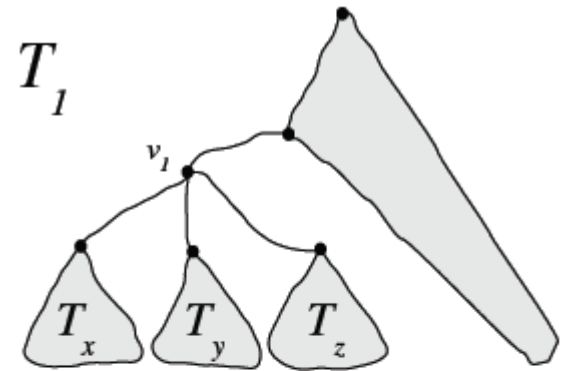
All colourings in one tree can be produced in $O(n \log n)$ –
Smaller half trick

Each colouring can be counted in amortized $O(\log n)$ –
Hierarchical decomposition and polynomial manipulations

Comparing sets of trees

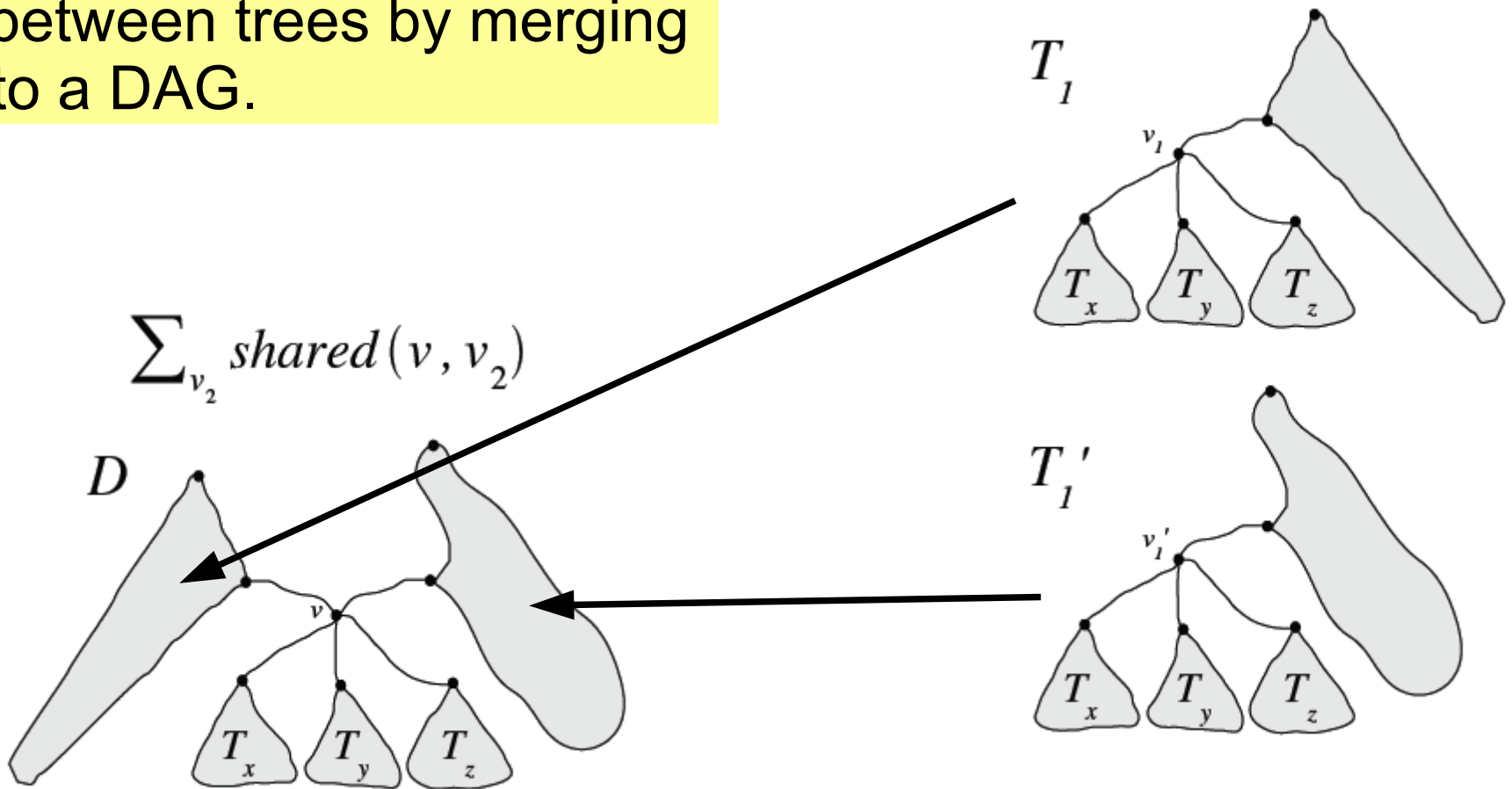
Observation: Shared sub-trees implies shared quartets.

$$\sum_{v_2} \text{shared}(v_1, v_2) = \sum_{v_2} \text{shared}(v_1', v_2)$$



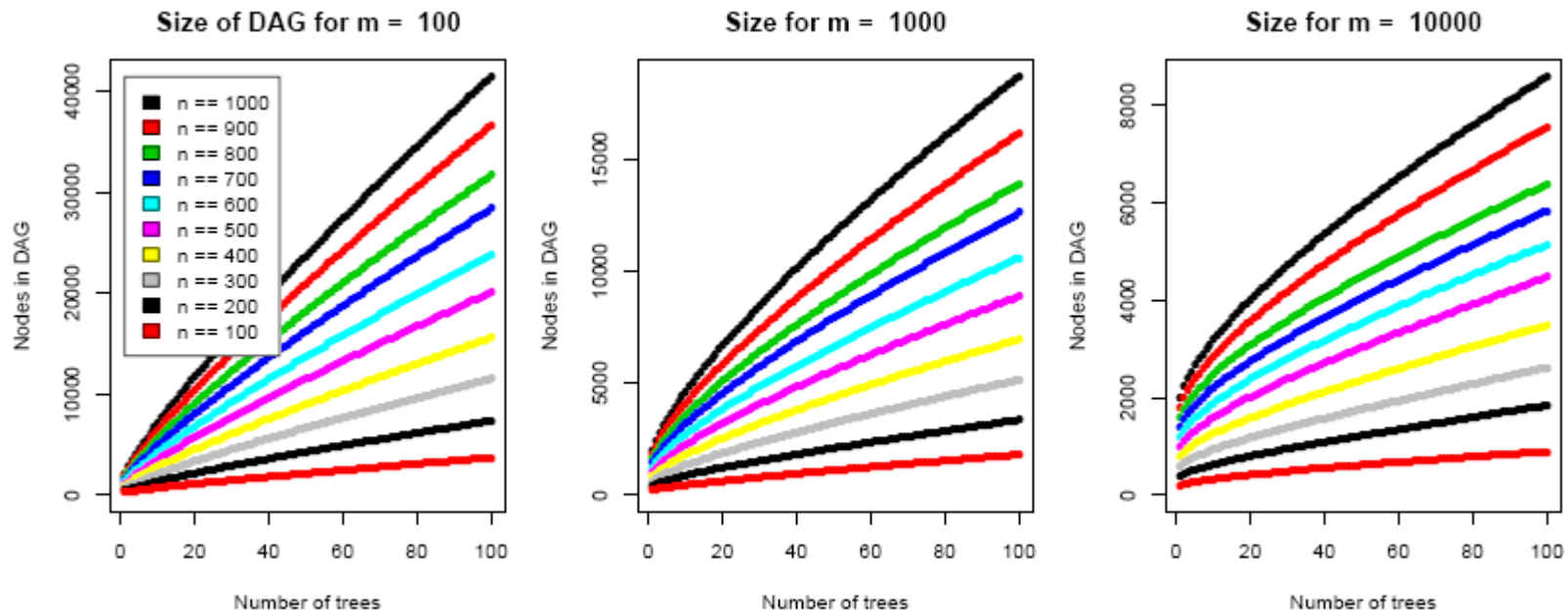
Comparing sets of trees

Observation: We can share results between trees by merging them into a DAG.



Experiment – DAG size

The size of the DAG depends on the similarity of the input trees

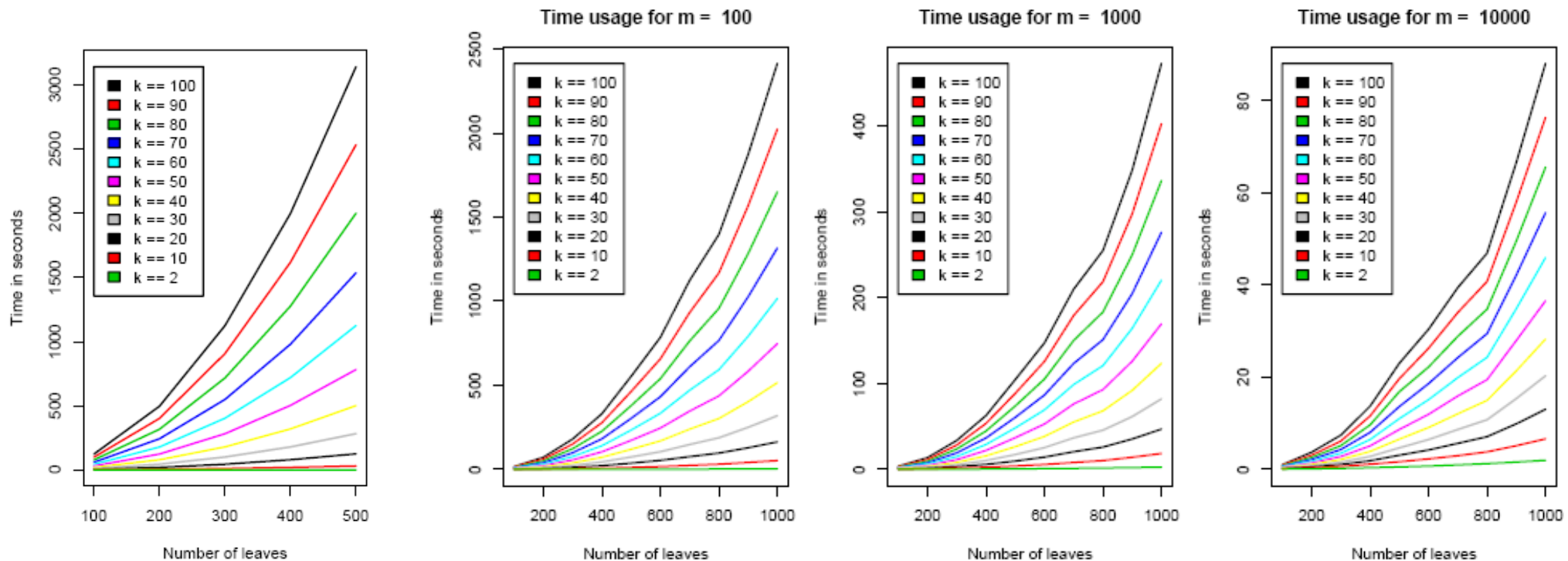


Simulation setup

Simulating one binary tree with n leaves: evolve kn sequences of length m on the k tree, construct k neighbor joining trees from the resulting kn sequences

Experiment – Running time

Running time of “DAG versus DAG” algorithm on a standard Linux PC



The straightforward “all-against-all” approach takes 3000 second for comparing 100 trees of size 500

The “DAG-DAG” approach takes between 20 and 500 seconds for the same comparisons depending on the similarity of the trees

Summary

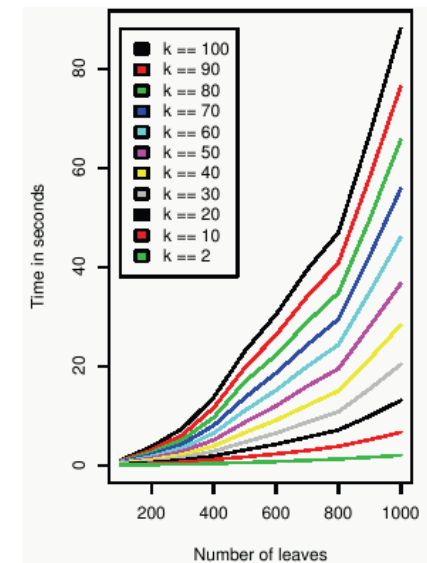
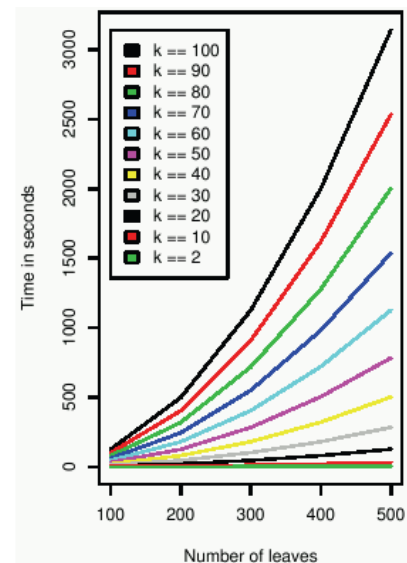
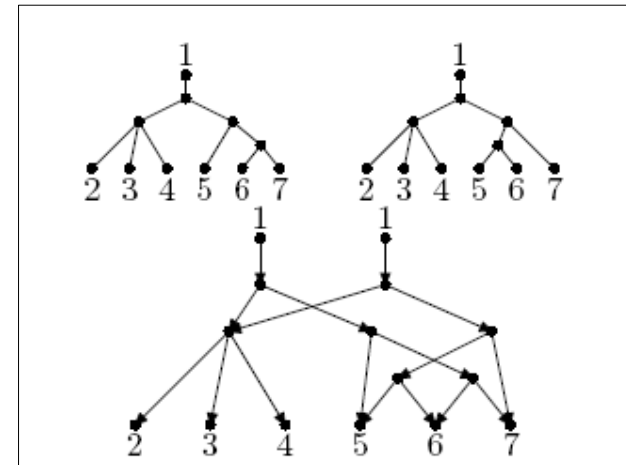
Results

Utilizing shared tree structure speeds up the computation significantly ...

Future work

Extension to non-binary trees and trees with unequal leaf sets

Applications?



Computing the quartet distance between evolutionary trees of bounded degree

Martin Stissing ¹

Christian Nørgaard Storm Pedersen ¹

Thomas Mailund ^{1,2}

Gerth Stølting Brodal ¹

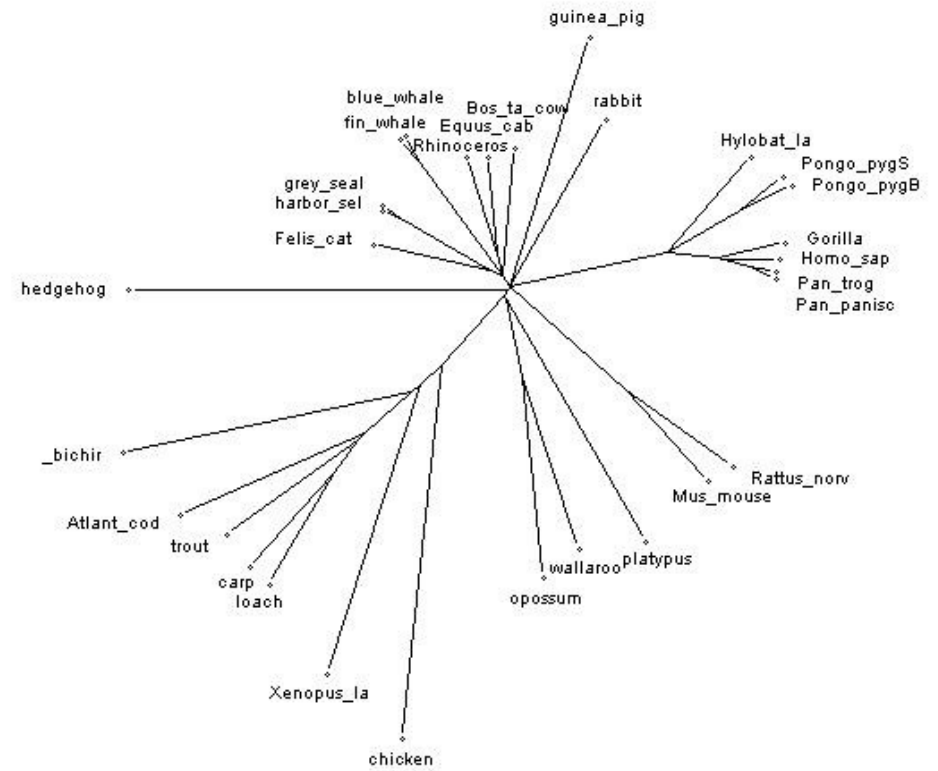
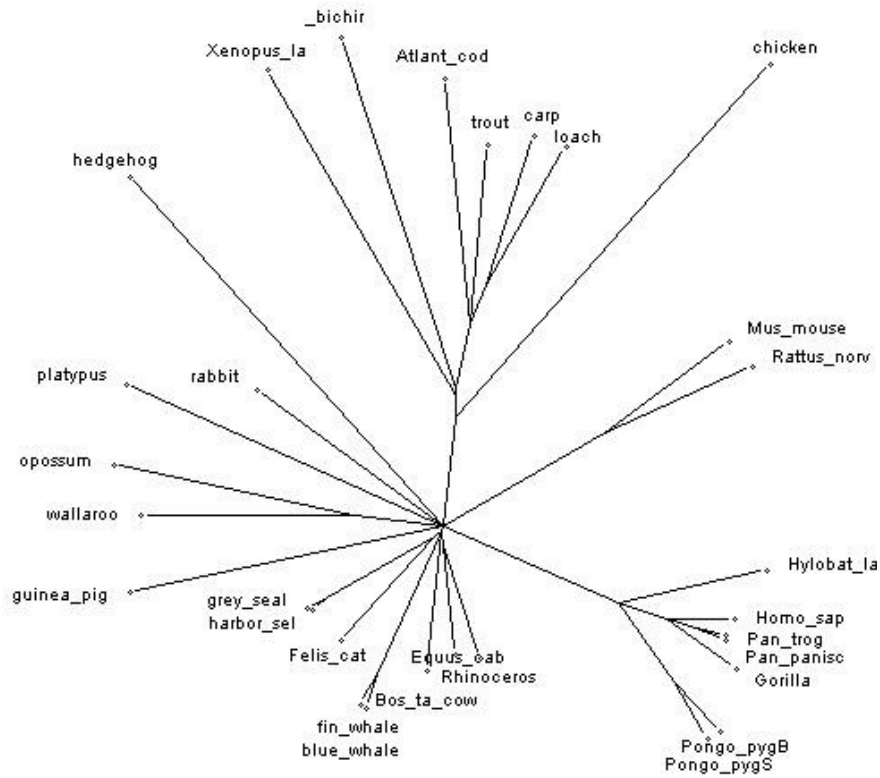
Rolf Fagerberg ³

(1) University of Aarhus, (2) Oxford University, (3) University of Southern Denmark

Comparing partially resolved trees

0.01

0.01



C. Christiansen, T. Mailund, C. N. S. Pedersen, and M. Randers
Algorithms for computing the quartet distance between trees of arbitrary degree
Proc of Workshop on Algorithms in Bioinformatics (WABI), 77-88, 2005 $O(n^3)$
 $O(d^2n^2)$

G. Estabrook
Comparing
four evolutionary trees
C. Christiansen, T. Mailund, C. N. S. Pedersen, M. Randers, and M. Stissing
Fast calculation of the quartet distance between trees of arbitrary degrees
Algorithms for Molecular Biology, 1(16), 2006 $O(dn^2)$

W. Day
Expected
Proc. 18
M. Stissing, C. N. S. Pedersen, T. Mailund, G. S. Brodal, and R. Fagerberg
Computing the quartet distance between evolutionary trees of bounded degree
Proc. of APBC, 2007 $O(d^9 n \log n)$

C. R. Doucette
An efficient algorithm to compute quartet dissimilarity measures. $O(n^3)$

Algorithms are for *fully resolved trees* (binary trees) only.

Quartet distance between binary trees seems easier to compute ...

Systematic Biology, 42(2):126–141, 1993.

D. Bryant, J. Tsang, P. E. Kearney, and M. Li. $O(n^3)$
Computing the quartet distance between evolutionary trees.
Proc. 11th Symp. on Discrete Algorithms (SODA), 285–286, 2000.

G. S. Brodal, R. Fagerberg, and C. N. S. Pedersen $O(n \log^2 n)$
Computing the quartet distance in time $O(n \log n)$. $O(n \log n)$
Algorithmica, 28(2): 377-395, 2003.

C. Christiansen, T. Mailund, C. N. S. Pedersen, and M. Randers
Algorithms for computing the quartet distance between trees of arbitrary degree
Proc of Workshop on Algorithms in Bioinformatics (WABI), 77-88, 2005 $O(n^3)$
 $O(d^2n^2)$

G. Estabrook, C. Christiansen, T. Mailund, C. N. S. Pedersen, M. Randers, and M. Stissing
Comparing trees: Fast calculation of the quartet distance between trees of arbitrary degrees
four evolutionary trees
Algorithms for Molecular Biology, 1(16), 2006 $O(dn^2)$

W. Day, M. Stissing, C. N. S. Pedersen, T. Mailund, G. S. Brodal, and R. Fagerberg
Expected quartet distance
Proc. 18th
Computing the quartet distance between evolutionary trees of bounded degree
Proc. of APBC, 2007 $O(d^9 n \log n)$

C. R. Doucette
An efficient algorithm to compute quartet dissimilarity measures. $O(n^3)$

Algorithms are for *fully resolved trees* (binary trees) only.

Quartet distance between binary trees seems easier to compute ...

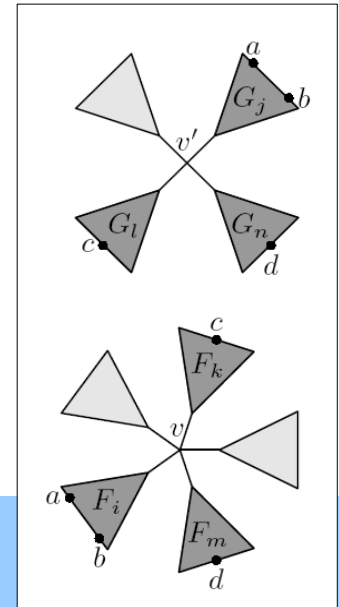
Systematic Biology, 42(2):126–141, 1993.

D. Bryant, J. Tsang, P. E. Kearney, and M. Li. $O(n^2)$
Computing the quartet distance between evolutionary trees.
Proc. 11th Symp. on Discrete Algorithms (SODA), 285–286, 2000.

G. S. Brodal, R. Fagerberg, and C. N. S. Pedersen $O(n \log^2 n)$
Computing the quartet distance in time $O(n \log n)$. $O(n \log n)$
Algorithmica, 38(2): 377-395, 2003.

Algorithms for partially resolved trees

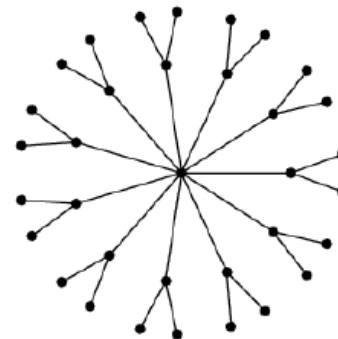
Idea: Iterate over all pairs of edges (or nodes) in the two trees, and count for each pair how many *associated quartets* are shared. The problem is to define “associated” such each quartet is counted as most once ...



Different solutions

Type	Time	Space
Center based	$O(n^3)$	$O(n)$
Edge based	$O(V V' \cdot d \cdot d')$	$O(n^2)$
Edge based	$O(n + V V' \cdot id \cdot id')$	$O(n + V V')$
Node based	$O(n + V V' \cdot \min\{id, id'\})$	$O(n + V V')$

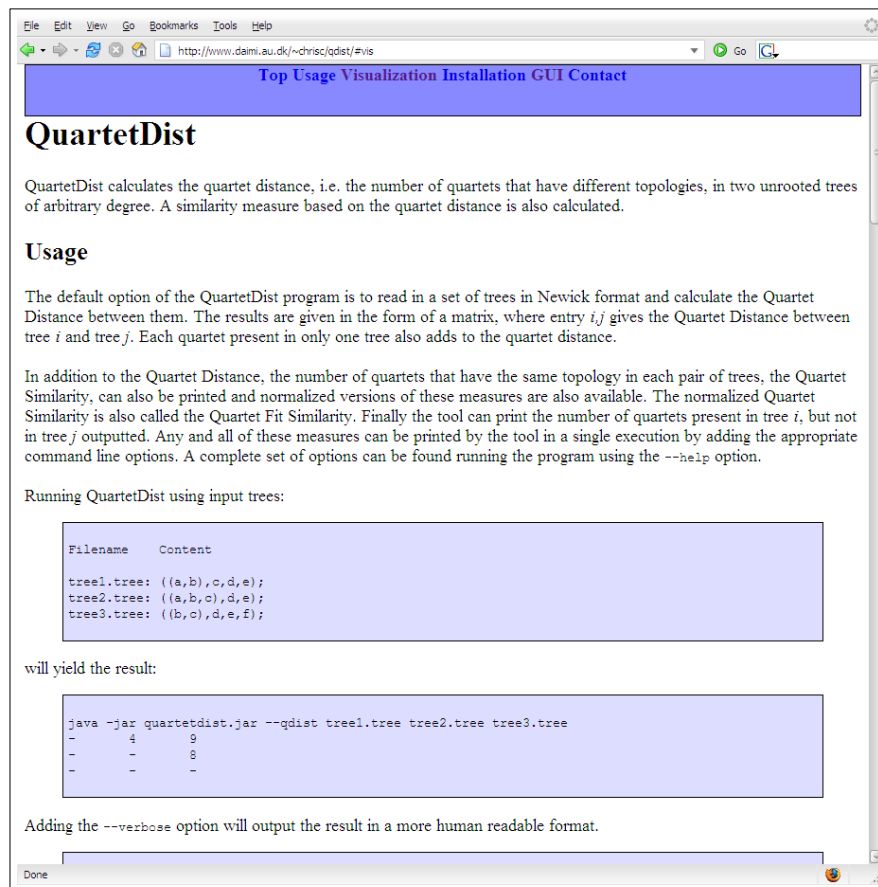
- n is the number of species/leaves
- $|V|$, $|V'|$ are number of internal nodes in T and T'
- id , id' are internal degree of T and T'



“worst case”-tree
 $|V|=id=O(n)$

QuartetDist

QuartetDist: Computes *quartet distance* (or normalized similarity, i.e. *quartet fit similarity*) between general trees



QuartetDist

QuartetDist calculates the quartet distance, i.e. the number of quartets that have different topologies, in two unrooted trees of arbitrary degree. A similarity measure based on the quartet distance is also calculated.

Usage

The default option of the QuartetDist program is to read in a set of trees in Newick format and calculate the Quartet Distance between them. The results are given in the form of a matrix, where entry i,j gives the Quartet Distance between tree i and tree j . Each quartet present in only one tree also adds to the quartet distance.

In addition to the Quartet Distance, the number of quartets that have the same topology in each pair of trees, the Quartet Similarity, can also be printed and normalized versions of these measures are also available. The normalized Quartet Similarity is also called the Quartet Fit Similarity. Finally the tool can print the number of quartets present in tree i , but not in tree j outputted. Any and all of these measures can be printed by the tool in a single execution by adding the appropriate command line options. A complete set of options can be found running the program using the `--help` option.

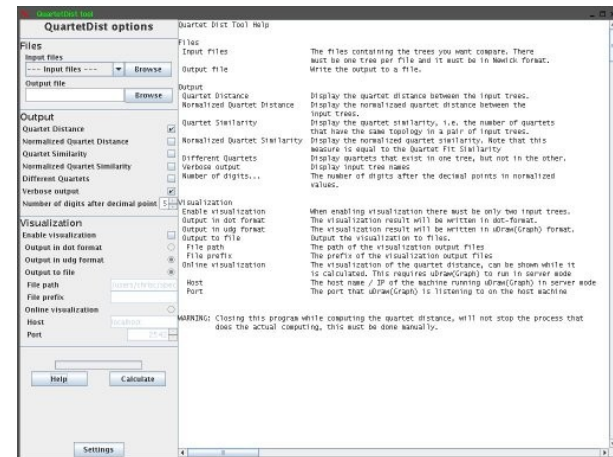
Running QuartetDist using input trees:

```
Filename  Content
tree1.tree: ((a,b),c,d,e);
tree2.tree: ((a,b,c),d,e);
tree3.tree: ((b,c),d,e,f);
```

will yield the result:

```
java -jar quartetdist.jar --qdist tree1.tree tree2.tree tree3.tree
-      4      9
-      -      8
-      -      -
```

Adding the `--verbose` option will output the result in a more human readable format.



QuartetDist options

Files

Input files:

Output file:

Output

Quartet Distance

Normalized Quartet Distance

Quartet Similarity

Normalized Quartet Similarity

Different Quartets

Verbose output

Number of digits after decimal point:

Visualization

Enable visualization

Output in dot format

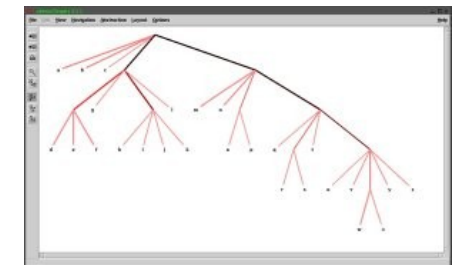
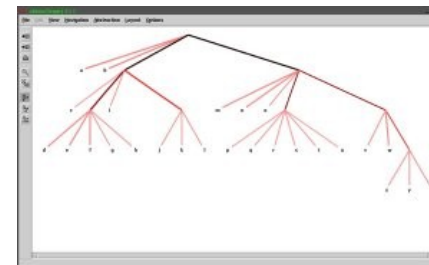
Output to file

File path

File prefix

Online visualization

Host: Port:



A simple GUI

Implementation in Java by Martin Randers and Chris Christiansen

<http://www.birc.au.dk/~chrisc/qdist/>

C. Christiansen, T. Mailund, C. N. S. Pedersen, and M. Randers
Algorithms for computing the quartet distance between trees of arbitrary degree
Proc of Workshop on Algorithms in Bioinformatics (WABI), 77-88, 2005 $O(n^3)$
 $O(d^2n^2)$

G. Estabrook
Comparing
four evolutionary trees
C. Christiansen, T. Mailund, C. N. S. Pedersen, M. Randers, and M. Stissing
Fast calculation of the quartet distance between trees of arbitrary degrees
Algorithms for Molecular Biology, 1(16), 2006 $O(dn^2)$

W. Day
Expected
Proc. 16
M. Stissing, C. N. S. Pedersen, T. Mailund, G. S. Brodal, and R. Fagerberg
Computing the quartet distance between evolutionary trees of bounded degree
Proc. of APBC, 2007 $O(d^9 n \log n)$

C. R. Doucette
An efficient algorithm to compute quartet dissimilarity measures. $O(n^3)$

Algorithms are for *fully resolved trees* (binary trees) only.

Quartet distance between binary trees seems easier to compute ...

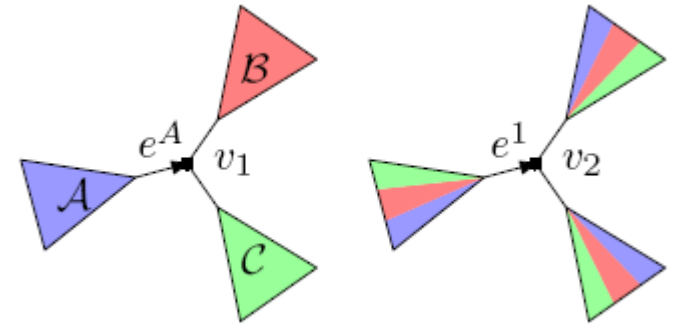
Systematic Biology, 42(2):126–141, 1993.

D. Bryant, J. Tsang, P. E. Kearney, and M. Li. $O(n^2)$
Computing the quartet distance between evolutionary trees.
Proc. 11th Symp. on Discrete Algorithms (SODA), 285–286, 2000.

G. S. Brodal, R. Fagerberg, and C. N. S. Pedersen $O(n \log^2 n)$
Computing the quartet distance in time $O(n \log n)$. $O(n \log n)$
Algorithmica, 38(2): 377-395, 2003.

Counting using colouring

Idea: Alternatively, colour leaves according to nodes in one tree and count compatible colourings in the other

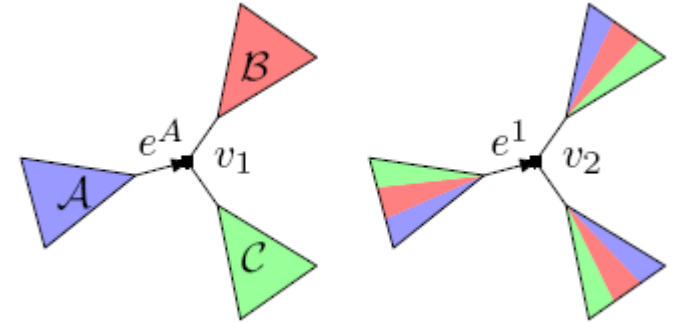


$$\text{count}(e^A, e^1) = \binom{a(1)}{2} \cdot (b(2) \cdot c(3) + b(3) \cdot c(2))$$

This approach does not explicitly consider all pairs of nodes/edges, which makes it possible to achieve a sub-quadratic running time

Counting using colouring

Idea: Alternatively, colour leaves according to nodes in one tree and count compatible colourings in the other



$$\text{count}(e^A, e^1) = \binom{a(1)}{2} \cdot (b(2) \cdot c(3) + b(3) \cdot c(2))$$

This approach does not explicitly consider all pairs of nodes/edges, which makes it possible to achieve a sub-quadratic running time

Ideas can be generalized to trees of maximum degree d . Use d colours instead of 3, use a more complex hierarchical decomposition tree ...

Counting using colouring

Idea: Alternatively, according to node and count compared in the other

General Algorithm:

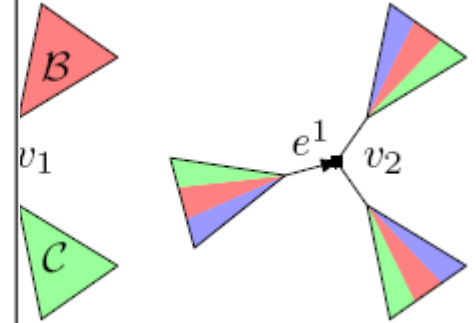
Shared = 0.

For each v_1 in T_1 do:

colour leaves according to v_1 .

Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

Quartet Distance = $\binom{n}{4} - \text{Shared}$.



$$\binom{n}{4} - \left(b(2) \cdot c(3) + b(3) \cdot c(2) \right)$$

This approach does not explicitly consider all pairs of nodes/edges, which makes it possible to achieve a sub-quadratic running time

Ideas can be generalized to trees of maximum degree d . Use d colours instead of 3, use a more complex hierarchical decomposition tree ...

Counting using colouring

Idea: Alternatively, according to node and count comparison in the other

General Algorithm:

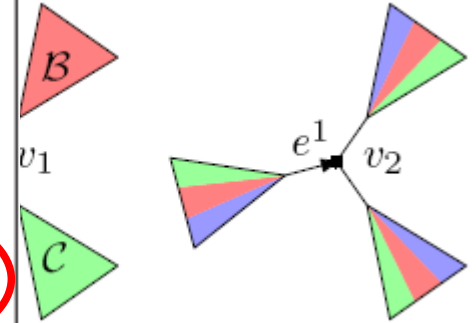
Shared = 0.

For each v_1 in T_1 do:

colour leaves according to v_1 .

Shared = Shared + $\sum_{v_2} shared(v_1, v_2)$.

Quartet Distance $\binom{n}{4} \text{Shared}$ $\binom{n}{4} (1(1) + 1(2) + 1(2) + 1(2) + 1(2))$



Efficiently calculated using hierarchical decomposition tree

This approach does not compare all pairs of nodes/edges, which makes it possible to achieve a sub-quadratic running time

Ideas can be generalized to trees of maximum degree d . Use d colours instead of 3, use a more complex hierarchical decomposition tree ...

Shared “butterfly” quartets

General Algorithm:

Shared = 0.

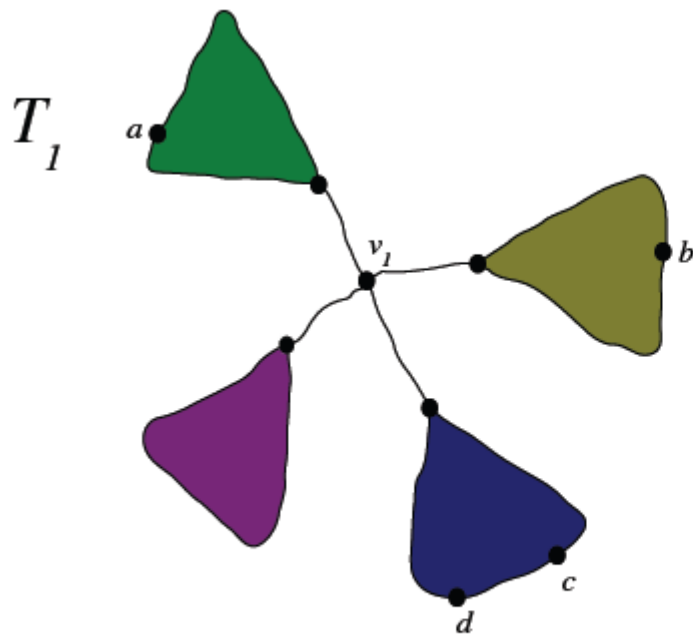
For each v_1 in T_1 do:

 colour leaves according to v_1

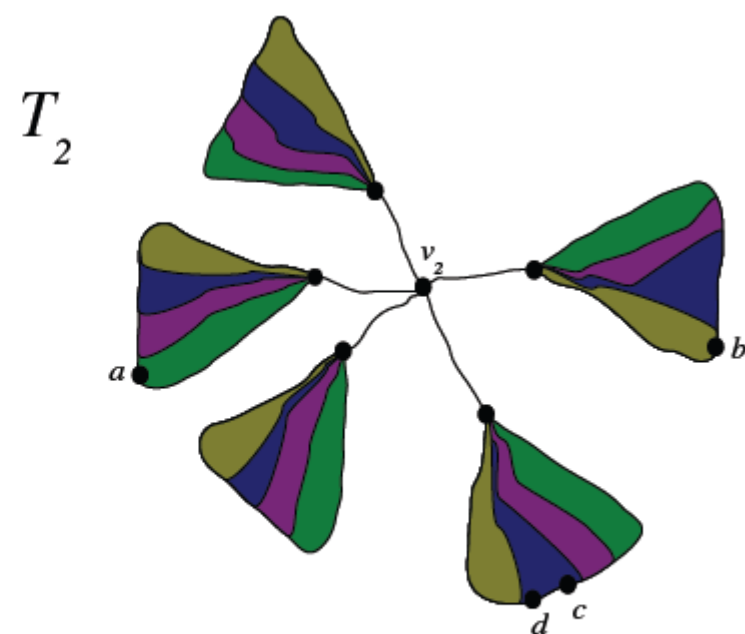
 Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

Quartet Distance = $\binom{n}{4} - \text{Shared}$.

$$B_{v_2}(c_1^1, c_2^1, \dots, c_5^4) = \sum_i \sum_j \sum_k \sum_{i'} \sum_{j'} \sum_{k'} c_{i'}^i c_{j'}^j \binom{c_{k'}}{2}$$



$ab|cd$



$ab|cd$

Shared “star” quartets

General Algorithm:

Shared = 0.

For each v_1 in T_1 do:

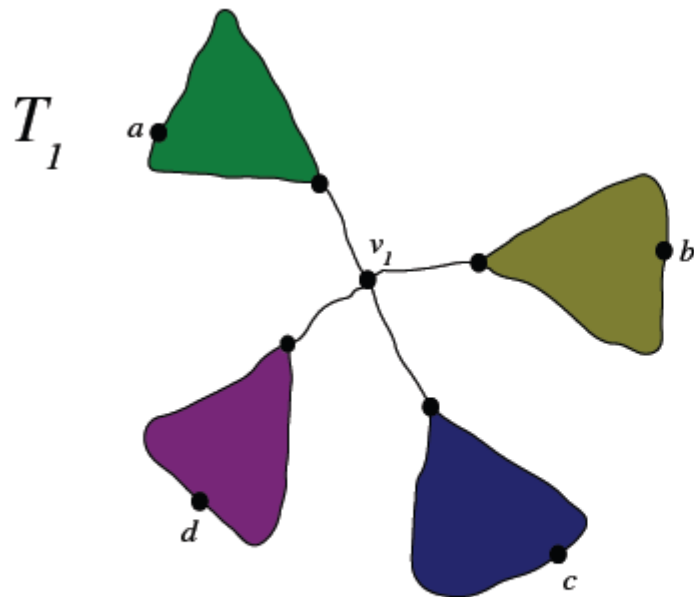
 colour leaves according to v_1 .

 Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

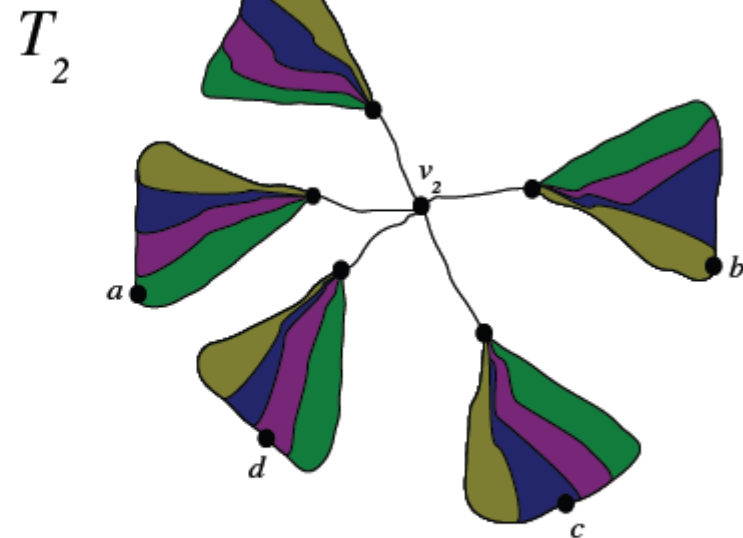
Quartet Distance = $\binom{n}{4} - \text{Shared}$.

$$S_{v_2}(c_1^1, c_2^1, \dots, c_5^4) =$$

$$\sum_i \sum_j \sum_k \sum_l \sum_{i'} \sum_{j'} \sum_{k'} \sum_{l'} c_{i'}^i c_{j'}^j c_{k'}^k c_{l'}^l$$



$abcd$



$abcd$

Colouring

General Algorithm:

Shared = 0.

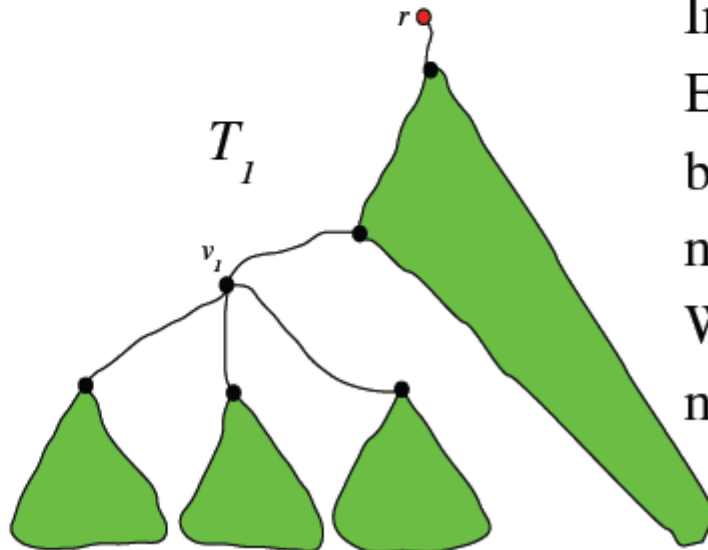
For each v_1 in T_1 do:

colour leaves according to v_1 .

Shared = Shared + $\sum_{v_2} shared(v_1, v_2)$.

Quartet Distance = $\binom{n}{4} - \text{Shared}$.

Illustration:



Invariant:

Encountering a node all leaves below are coloured **Green**, all nodes above coloured **Red**.

When done all leaves below the node are coloured **Red**.

Colouring

General Algorithm:

Shared = 0.

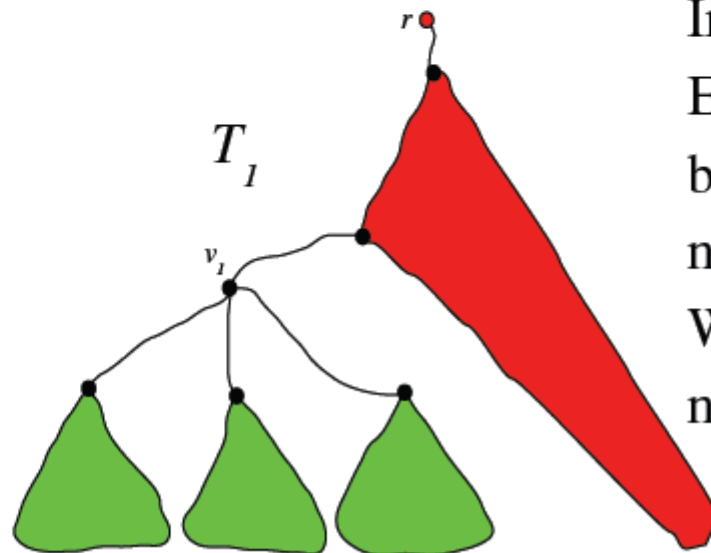
For each v_1 in T_1 do:

colour leaves according to v_1 .

Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

Quartet Distance = $\binom{n}{4} - \text{Shared}$.

Illustration:



Invariant:

Encountering a node all leaves below are coloured **Green**, all nodes above coloured **Red**.

When done all leaves below the node are coloured **Red**.

Colouring

General Algorithm:

Shared = 0.

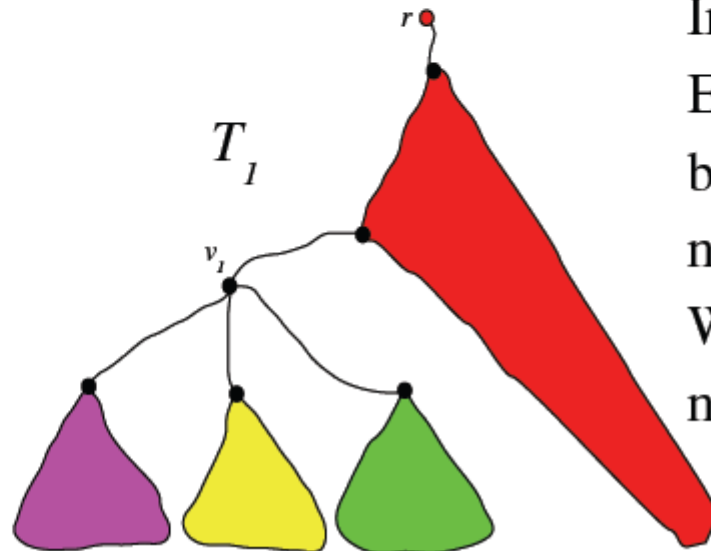
For each v_1 in T_1 do:

colour leaves according to v_1 .

Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

Quartet Distance = $\binom{n}{4} - \text{Shared}$.

Illustration:



Invariant:

Encountering a node all leaves below are coloured **Green**, all nodes above coloured **Red**.

When done all leaves below the node are coloured **Red**.

Colouring

General Algorithm:

Shared = 0.

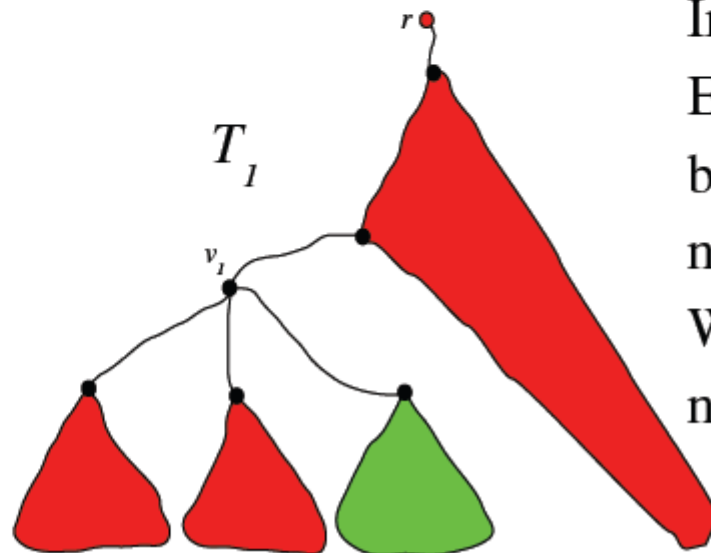
For each v_1 in T_1 do:

colour leaves according to v_1 .

Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

Quartet Distance = $\binom{n}{4} - \text{Shared}$.

Illustration:



Invariant:

Encountering a node all leaves below are coloured **Green**, all nodes above coloured **Red**.

When done all leaves below the node are coloured **Red**.

Colouring

General Algorithm:

Shared = 0.

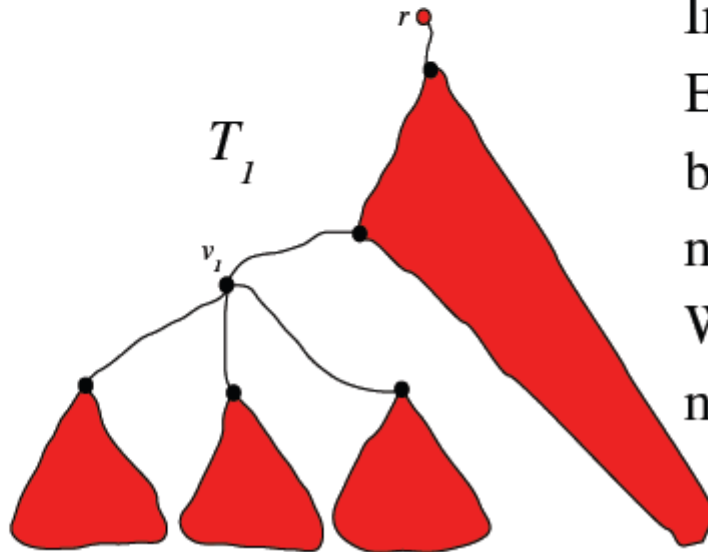
For each v_1 in T_1 do:

colour leaves according to v_1 .

Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

Quartet Distance = $\binom{n}{4} - \text{Shared}$.

Illustration:



Invariant:

Encountering a node all leaves below are coloured **Green**, all nodes above coloured **Red**.

When done all leaves below the node are coloured **Red**.

Colouring

General Algorithm:

Shared = 0.

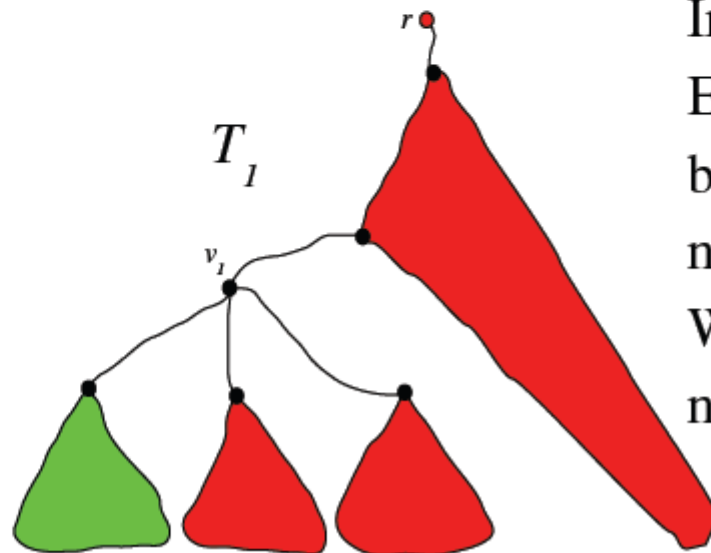
For each v_1 in T_1 do:

colour leaves according to v_1 .

Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

Quartet Distance = $\binom{n}{4} - \text{Shared}$.

Illustration:



Invariant:

Encountering a node all leaves below are coloured **Green**, all nodes above coloured **Red**.

When done all leaves below the node are coloured **Red**.

Colouring

General Algorithm:

Shared = 0.

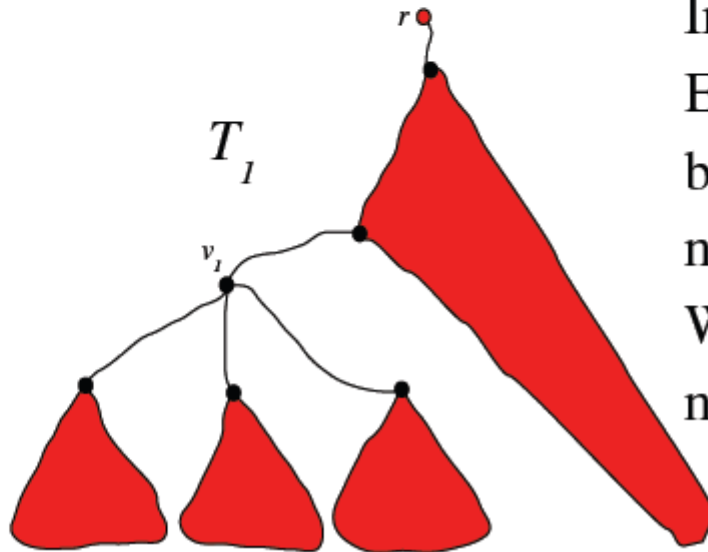
For each v_1 in T_1 do:

colour leaves according to v_1 .

Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

Quartet Distance = $\binom{n}{4} - \text{Shared}$.

Illustration:



Invariant:

Encountering a node all leaves below are coloured **Green**, all nodes above coloured **Red**.

When done all leaves below the node are coloured **Red**.

Colouring

General Algorithm:

Shared = 0.

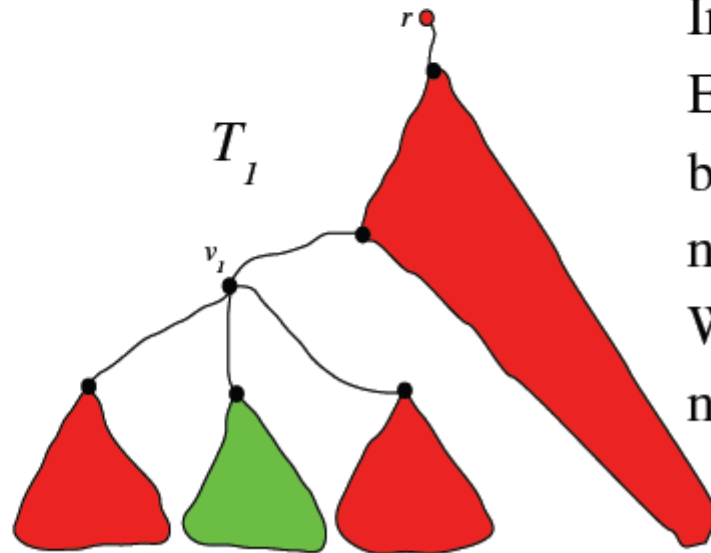
For each v_1 in T_1 do:

colour leaves according to v_1 .

Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

Quartet Distance = $\binom{n}{4} - \text{Shared}$.

Illustration:



Invariant:

Encountering a node all leaves below are coloured **Green**, all nodes above coloured **Red**.

When done all leaves below the node are coloured **Red**.

Colouring

General Algorithm:

Shared = 0.

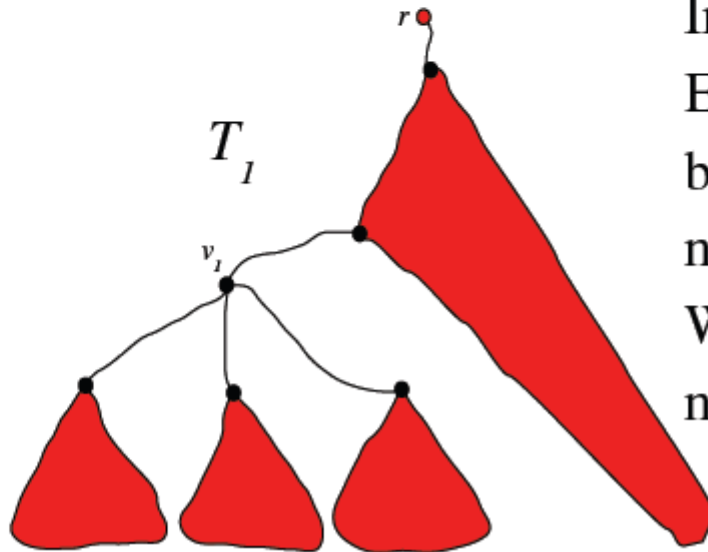
For each v_1 in T_1 do:

colour leaves according to v_1 .

Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

Quartet Distance = $\binom{n}{4}$ - Shared.

Illustration:



Invariant:

Encountering a node all leaves below are coloured **Green**, all nodes above coloured **Red**.

When done all leaves below the node are coloured **Red**.

Colouring

General Algorithm:

Shared = 0.

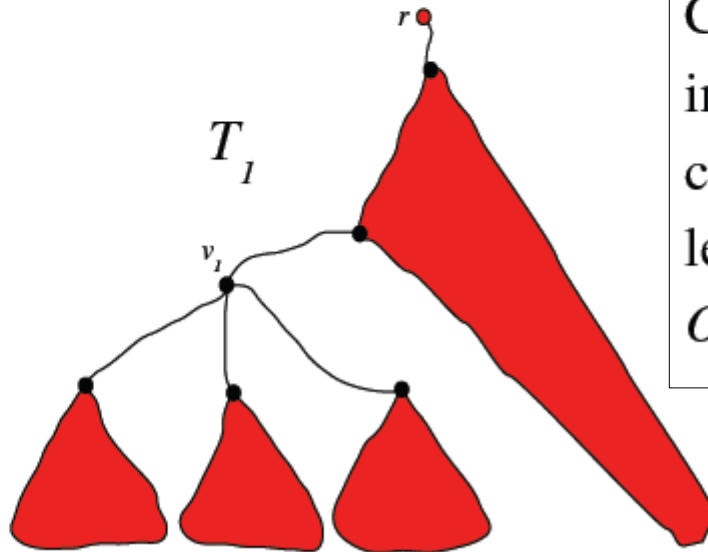
For each v_I in T_I do:

colour leaves according to v_I .

Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

Quartet Distance = $\binom{n}{4}$ - Shared.

Illustration:



Smaller-half trick

Considering a node, only leaves in smaller subtrees are coloured. This means that any leaf is coloured no more than $O(\log(n))$ times.

Colouring

General Algorithm:

Shared = 0.

For each v_I in T_I do:

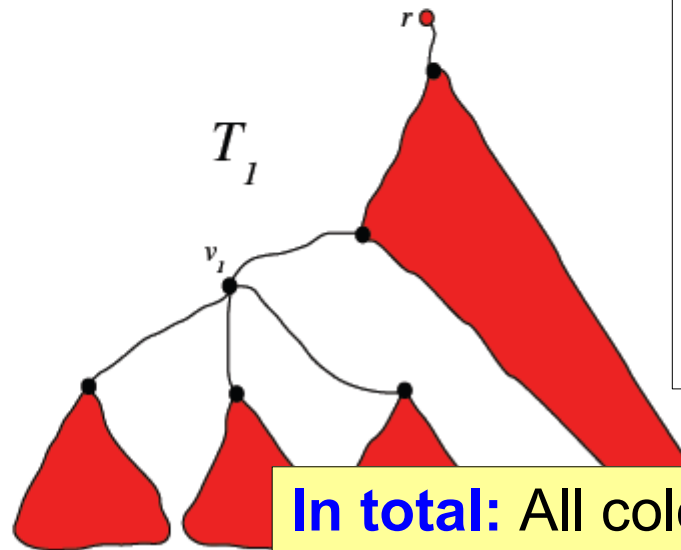
colour leaves according to v_I .

Shared = Shared + $\sum_{v_2} \text{shared}(v_1, v_2)$.

Quartet Distance = $\binom{n}{4}$ - Shared.

Smaller-half trick

Illustration:



Considering a node, only leaves in smaller subtrees are coloured. This means that any leaf is coloured no more than $O(\log(n))$ times.

In total: All colourings takes time $O(n \log n)$...

Hierarchical decomposition tree

Let T be an unrooted tree of size n with degree at most d . A **component** C in T is:

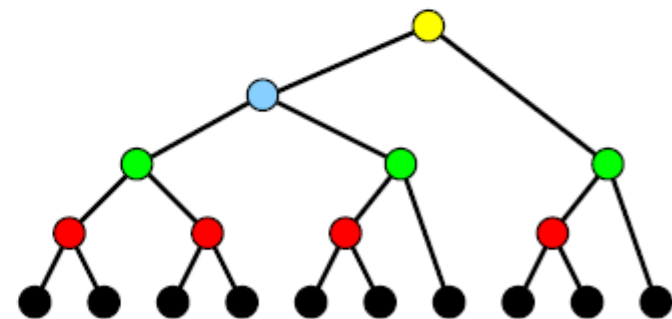
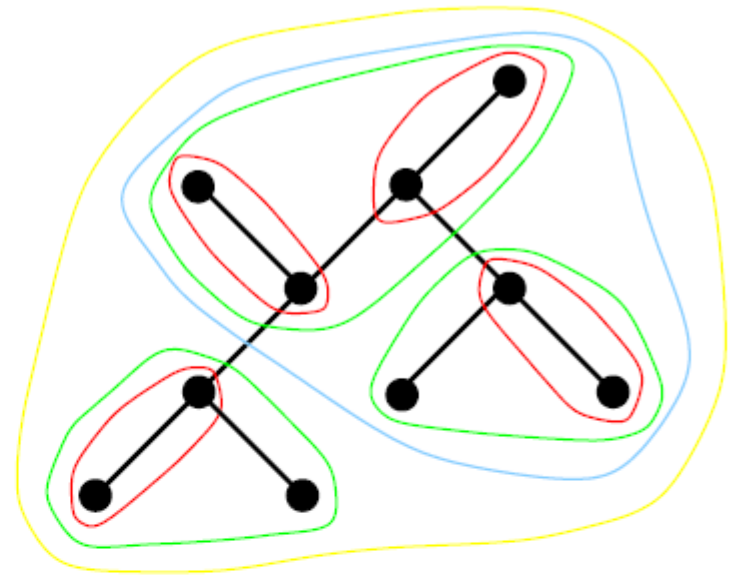
A single node in T , or

A connected subset of nodes in T with degree at most d .

A **hierarchical decomposition tree** $H(T)$ of T is a rooted binary tree:

A leaf is simple component (single node in T).

An internal node is a composite component of its children.



Hierarchical decomposition tree

Let T be an unrooted tree of size n with degree at most d . A **component C** in T is:

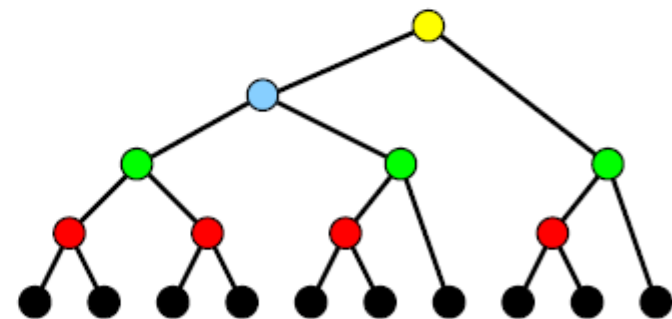
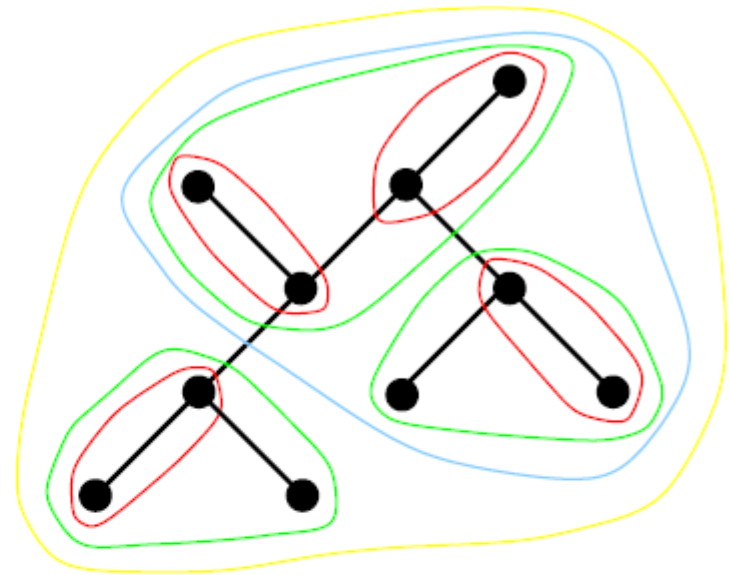
A single node in T , or

A connected subset of nodes in T with degree at most d .

A **hierarchical decomposition tree $H(T)$** of T is a rooted binary tree:

A leaf is simple component (single node in T).

An internal node is a composite component.



Lemma: A hierarchical decomposition tree $H(T)$ of T of height $O(d \log n)$ can be constructed in time $O(dn)$

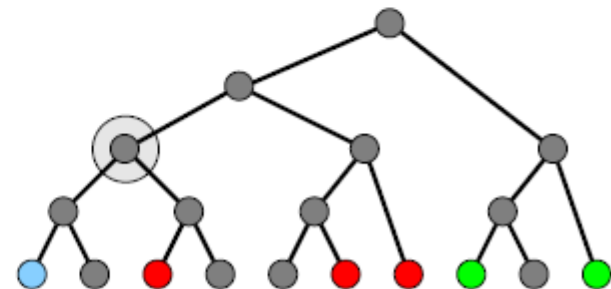
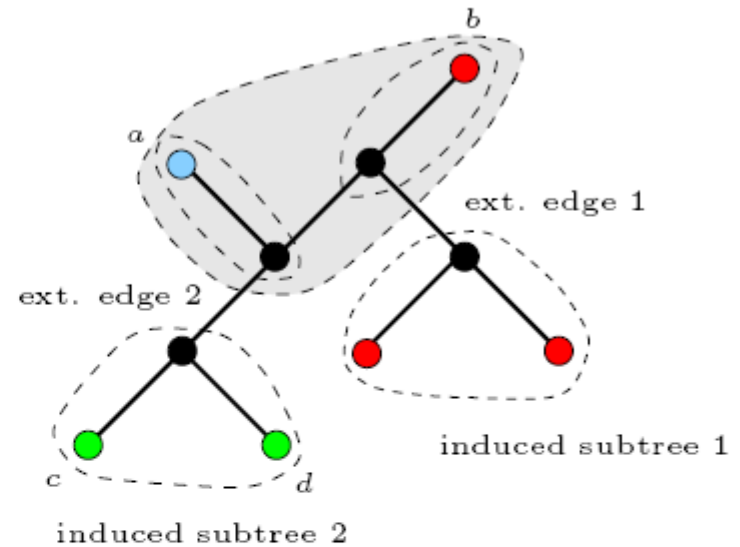
Hierarchical decomposition tree

Annotation of node C in $H(T)$

A d -tuple denoting the number of leaves in C coloured in each of the d colours.

A function F of $d \times$ “degree of C ” variables – representing counts of each colour in the remaining leaves of T

The function F yields the number of quartets associated to nodes in C and compatible with colouring ...

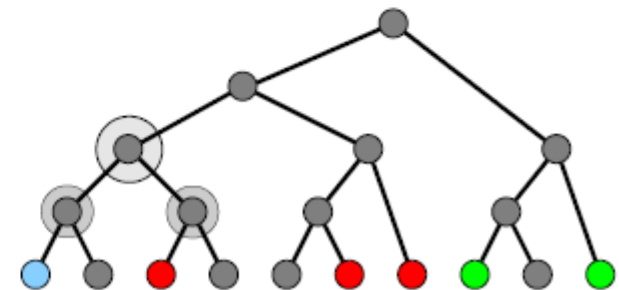
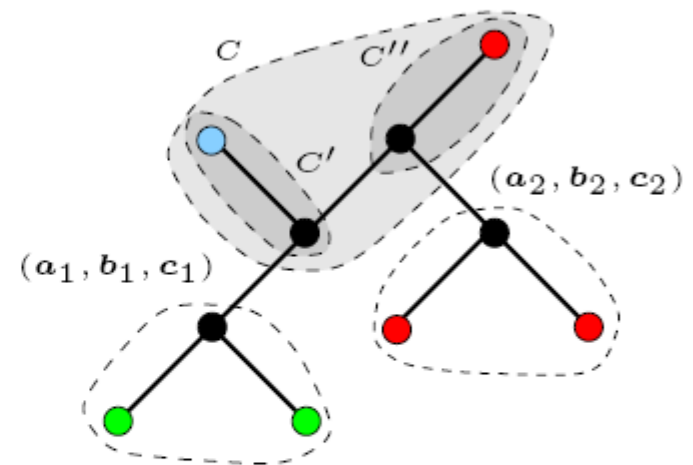


Hierarchical decomposition tree

Annotation is done bottom-up: For a leaf, it depends on whether it is a leaf or node in T . For a node, it depends on the type of composition ...

Example:

- $(a' + a'', b' + b'', c' + c'')$
- $F(a_1, b_1, c_1, a_2, b_2, c_2)$
= $F'(a_2 + a'', b_2 + b'', c_2 + c'', a_1, b_1, c_1)$
+ $F''(a_1 + a', b_1 + b', c_1 + c', a_2, b_2, c_2)$



Hierarchical decomposition tree

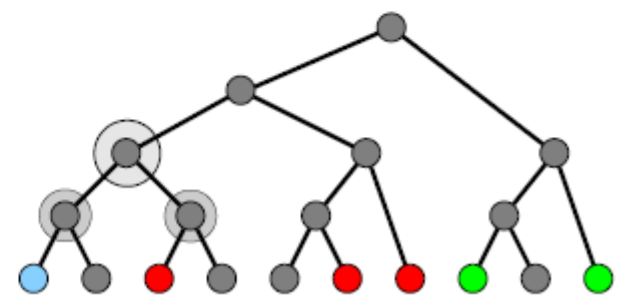
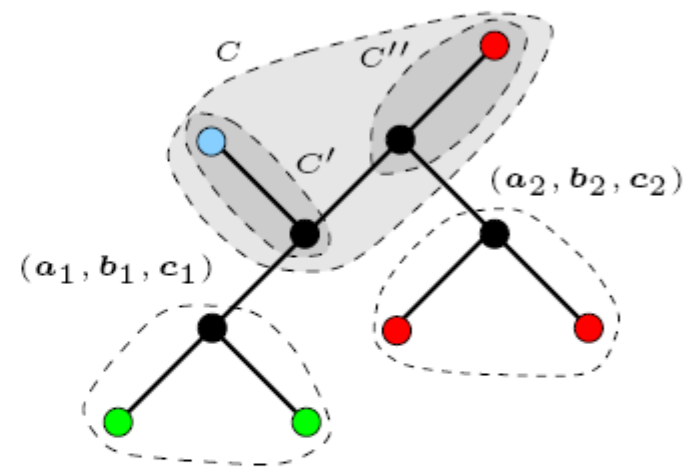
$$S_{v_2}(c_1^1, c_2^1, \dots, c_5^4) = \sum_i \sum_j \sum_k \sum_l \sum_{i'} \sum_{j'} \sum_{k'} \sum_{l'} c_{i'}^i c_{j'}^j c_{k'}^k c_{l'}^l$$

$$B_{v_2}(c_1^1, c_2^1, \dots, c_5^4) = \sum_i \sum_j \sum_k \sum_{i'} \sum_{j'} \sum_{k'} c_{i'}^i c_{j'}^j \binom{c_{k'}^k}{2}$$

or node in T . For a node, it depends on the

Example:

- $(a' + a'', b' + b'', c' + c'')$
- $F(a_1, b_1, c_1, a_2, b_2, c_2)$
 $= F'(a_2 + a'', b_2 + b'', c_2 + c'', a_1, b_1, c_1)$
 $+ F''(a_1 + a', b_1 + b', c_1 + c', a_2, b_2, c_2)$



Hierarchical decomposition tree

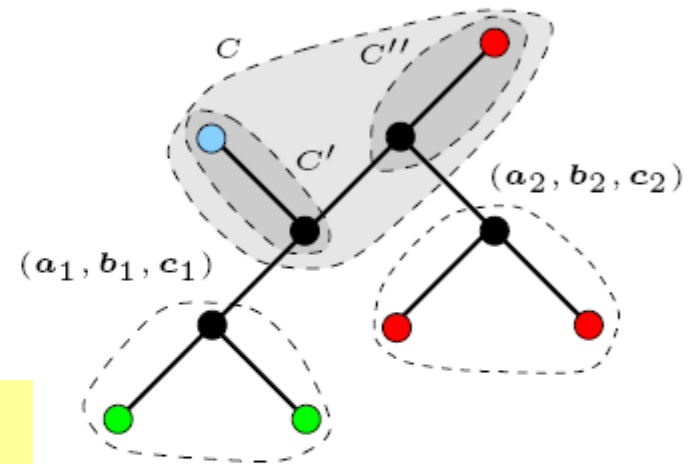
$$S_{v_2}(c_1^1, c_2^1, \dots, c_5^4) = \sum_i \sum_j \sum_k \sum_l \sum_{i'} \sum_{j'} \sum_{k'} \sum_{l'} c_{i'}^i c_{j'}^j c_{k'}^k c_{l'}^l$$

$$B_{v_2}(c_1^1, c_2^1, \dots, c_5^4) = \sum_i \sum_j \sum_k \sum_{i'} \sum_{j'} \sum_{k'} c_{i'}^i c_{j'}^j \binom{c_{k'}^k}{2}$$

or node in T . For a node, it depends on the

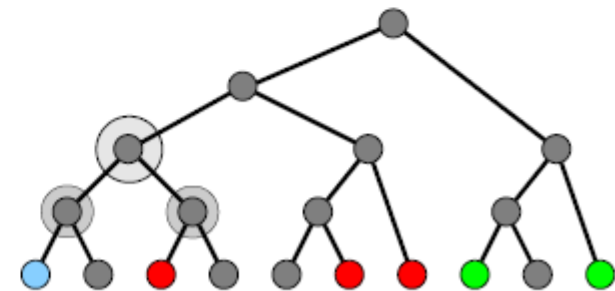
Example:

- $(a' + a'', b' + b'', c' + c'')$
- $F(a_1, b_1, c_1, a_2, b_2, c_2)$
 $= F'(a_2 + a'', b_2 + b'', c_2 + c'', a_1, b_1, c_1)$
 $+ F''(a_1 + a', b_1 + b', c_1 + c', a_2, b_2, c_2)$



Observation: F is a polynomial of at most d^2 variables of degree at most 4, i.e. It can be manipulated in time $O(d^8)$...

Lemma: Initial annotation takes time $O(d^9 n)$. Updating the annotation when changing a colour takes amortized time $O(d^9 \log n)$...



Hierarc

on tree

$$S_{v_2}(c_1^1, c_2^1, \dots, c_5^4) = \sum_i \sum_j \sum_k \sum_l \sum_{i'}$$

or node in T . For

General Algorithm:
 Shared = 0.
 For each v_1 in T_1 do:
 colour leaves according to v_1 .
 Shared = Shared + $\sum_{v_2} shared(v_1, v_2)$.
 Quartet Distance = $\binom{n}{4}$ - Shared.

$$) = \sum_{j'} \sum_{k'} c_{i'}^j c_{j'}^k \binom{c_{k'}}{2}$$

Ex

Summing it all up

-
- Colouring leaves and updating $H(T)$ takes amortized time $O(\log n \times d^9 \log n)$ per node v_1 in T_1 . Counting takes time $O(1)$ per node v_1 in T_1 . In total: $O(d^9 n \log^2 n)$.

... with “contractions” in $H(T)$ the total time can be improved to **$O(d^9 n \log n)$** ...

Obs

d^2 variables of degree at most 4, i.e. it can be manipulated in time $O(d^8)$...

Lemma: Initial annotation takes time $O(d^9 n)$. Updating the annotation when changing a colour takes amortized time $O(d^9 \log n)$...

