BIOINFORMATICS APPLICATIONS NOTE Vol. 30 no. 14 2014, pages 2079–2080
doi:10.1093/bioinformatics/btu157

Phylogenetics

Advance Access publication March 20, 2014

tqDist: a library for computing the quartet and triplet distances between binary or general trees

Andreas Sand[1,2], Morten K. Holt[2], Jens Johansen[2], Gerth Stølting Brodal[2,3], Thomas Mailund[1] and Christian N. S. Pedersen[1,2,*]

[1]Bioinformatics Research Centre, [2]Department of Computer Science and [3]MADALGO, Center for Massive Data Algorithms, a Center of the Danish National Research Foundation, Aarhus University, DK-8000 Aarhus C, Denmark

Associate Editor: David Posada

ABSTRACT

Summary: tqDist is a software package for computing the triplet and quartet distances between general rooted or unrooted trees, respectively. The program is based on algorithms with running time $O(n \log n)$ for the triplet distance calculation and $O(d \cdot n \log n)$ for the quartet distance calculation, where n is the number of leaves in the trees and d is the degree of the tree with minimum degree. These are currently the fastest algorithms both in theory and in practice.

Availability and implementation: tqDist can be installed on Windows, Linux and Mac OS X. Doing this will install a set of command-line tools together with a Python module and an R package for scripting in Python or R. The software package is freely available under the GNU LGPL licence at http://birc.au.dk/software/tqDist.

Contact: cstorm@birc.au.dk

Supplementary information: Supplementary data are available at *Bioinformatics* online

Received on December 11, 2013; revised on February 21, 2014; accepted on March 17, 2014

1 INTRODUCTION

Using trees to represent evolutionary relationships is a widespread technique in many scientific fields, in particular in biology, where trees are used to represent relationships between species or between genes in a gene family (Semple and Steel, 2003). But trees are also used, for example, in linguistics, where trees may be used to describe the evolution of related languages (Gray *et al.*, 2009; Walker *et al.*, 2012), and in archaeology, where trees have been used to represent how copies of ancient manuscripts have changed over time (Buneman, 1971). A common goal in all of these fields is to reconstruct the true tree from observed data. However, depending on both the available data and the reconstruction method, the inferred trees may differ.

In such cases, tree distances are often used as a formal way to quantify the differences and to determine whether two trees are more similar than would be expected by chance. The triplet distance [introduced as the *triples distance* in Critchlow *et al.*, (1996)] and quartet distance (Estabrook *et al.*, 1985) enumerate all subsets of leaves of size three and four, respectively, and count how often the topologies induced by the three or four leaves agree in the two trees. The triplet distance is intended for rooted trees, where the triplet topology is the smallest informative subtree (for unrooted trees all subtrees with three leaves have the same topology), whereas the quartet distance is intended for unrooted trees, where the quartet topology is the smallest informative subtree. The fastest known algorithms for computing the triplet and quartet distances have time complexity $O(n \log n)$ for both distances on binary trees with n leaves and complexity $O(n \log n)$ for the triplet distance and $O(d \cdot n \log n)$ for the quartet distance on general trees, where d is the minimum degree of the two trees, and the degree of a tree is the maximum degree of a node in the tree (Brodal *et al.*, 2013; Holt *et al.*, 2014).

In this applications note, we present an efficient and easy-to-use implementation of these algorithms and show that this is the fastest implementation available for computing the triplet and quartet distances. A detailed description of the algorithms and their implementation is given in Brodal *et al.* (2013) and Holt *et al.* (2014).

2 ALGORITHM

In this section, we briefly describe the triplet distance algorithm. The quartet distance algorithm is somewhat more involved, but it also builds on the ideas presented here.

For general trees, a triplet can be either resolved (the induced topology of the triplet is a binary tree) or unresolved (the induced topology is a trifurcation). Thus, given a triplet and two trees, T_1 and T_2, the triplet can be resolved in both trees, unresolved in both trees or resolved in one and unresolved in the other. If it is resolved in both trees, the induced topologies may agree or disagree in the two trees. In the unresolved–unresolved case they always agree, and in the resolved–unresolved case they always disagree. For binary trees, the induced topologies are always resolved.

Our algorithm computes the number of triplet topology differences implicitly by computing the number of shared triplet topologies and subtracting it from the total number of triplets, $\binom{n}{3}$. The number of shared triplet topologies can, according to the classification above, be computed as the number of triplets that are unresolved in both trees plus the number of triplets that are both resolved and induce the same topology in both trees. The first of these numbers can be computed in $O(n)$ time using a simple dynamic programming algorithm. To compute the

*To whom correspondence should be addressed.

number of triplets that are resolved and induce the same topology in both trees, we associate every triplet to a unique node in T_1 and then, in a recursive traversal of T_1, count how many of the triplets associated with each node in T_1 induce the same topology in T_2. Using the *smaller-half trick* to traverse T_1 and a balanced binary tree data structure called a *hierarchical decomposition tree* to count in T_2 (Brodal *et al*., 2013), this is done in $O(n \log n)$ time. Hence, in total our algorithm runs in $O(n \log n)$ time.

3 IMPLEMENTATION

`tqDist` is an implementation of the algorithm described above together with the corresponding quartet distance algorithm. The software package is carefully implemented in C++, and interfaces for scripting in Python (via the module `pyTQDist`) and in R (via the package `rtqdist`) are also provided. The software package can be installed easily using precompiled installers or using CMake on Windows, OS X and Linux, and the source is furthermore freely available for installation on any platform supporting CMake, make and any standard C++ compiler.

The software package includes four executables: `triplet_distance`, `quartet_distance`, `pairs_triplet_distance` and `pairs_quartet_distance`. The first two both take two files, each containing a single tree in the Newick format, as input and outputs the distance between the two trees. The last two executables take two files, each containing the same number of trees in the Newick format, as input and outputs a list of numbers where the *i*th number is the distance between the two trees on line *i* in the two files. Similar functions (`tripletDistance`, `quartetDistance`, `pairsTripletDistance` and `pairsQuartetDistance`) are provided in `pyTQDist` and `rtqdist`. The following short code snippet shows how the python module may be used:

```
from pyTQDist import *

td=tripletDistance("tree1.new", "tree2.new")
print "triplet distance:", td
ptd = pairsTripletDistance("trees.new")
print "list of distances:", ptd
```

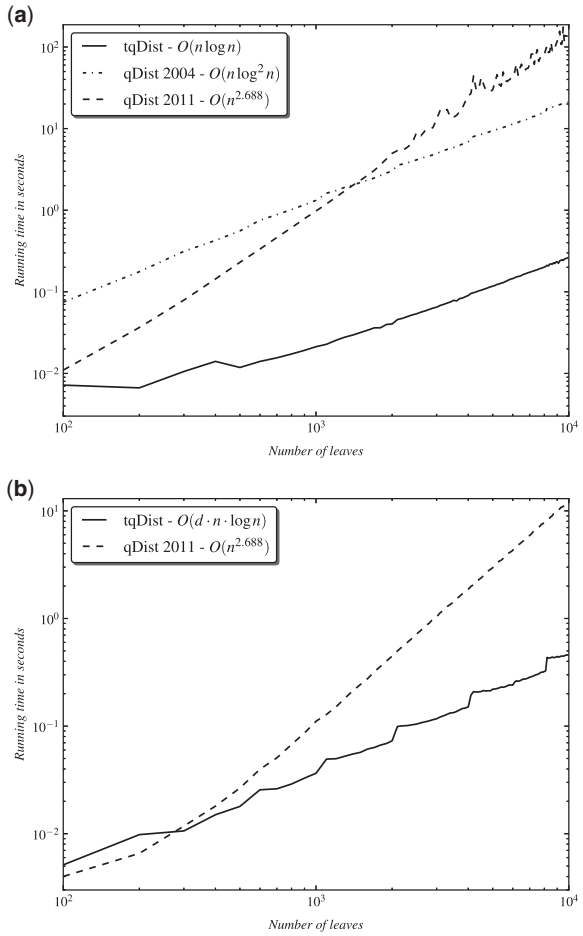

Fig. 1. Running time results for the quartet distance calculations on binary and non-binary unrooted trees. **(a)** Quartet distance running time on binary trees. **(b)** Quartet distance running time for trees of degree 128

4 PERFORMANCE

Figure 1 illustrates how the running time of our implementation of the new $O(d \log n)$ time quartet distance algorithm compares with the running times of the previously fastest algorithms: the $O(n^{2.688})$ time algorithm by Nielsen *et al*. (2011) for general trees and the $O(n \log^2 n)$ time algorithm by Brodal *et al*. (2004) for binary trees and implemented in `qDist` (Mailund and Pedersen, 2004). It is evident that our new implementation is fastest for almost all practical purposes, being up to 80 times faster for binary trees with up to 10 000 species and up to 25 times faster for trees with degree $d = 128$ and up to 10 000 species.

Funding: TM receives funding from The Danish Council for Independent Research, grant no. 12-125062.

Conflict of interest: none declared.

REFERENCES

Brodal,G.S. *et al.* (2004) Computing the quartet distance between evolutionary trees in time $O(n \log n)$. *Algorithmica*, **38**, 377–395.

Brodal,G.S. *et al.* (2013) Efficient algorithms for computing the triplet and quartet distance between trees of arbitrary degree. In: *ACM-SIAM Symposium on Discrete Algorithms (SODA)New Orleans, Louisiana, USA*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, USA, pp. 1814–1832.

Buneman,O.P. (1971) The recovery of trees from measures of dissimilarity. In: Hodson,F.R. *et al.* (eds) *Mathematics in the Archaeological and Historical Sciences*. Edinburgh University Press, Edinburgh.

Critchlow,D.E. *et al.* (1996) The triples distance for rooted bifurcating phylogenetic trees. *Syst. Biol.*, **45**, 323–334.

Estabrook,G.F. *et al.* (1985) Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. *Syst. Biol.*, **34**, 193–200.

Gray,R.D. *et al.* (2009) Language phylogenies reveal expansion pulses and pauses in pacific settlement. *Science*, **323**, 479–483.

Holt,M.K. *et al.* (2014) On the scalability of computing triplet and quartet distances. In: *Proceedings of 16th Workshop on Algorithm Engineering and Experiments (ALENEX) Portland, Oregon, USA*. Society for Industrial and Applied Mathematics (SIAM), Philadelphia, USA.

Mailund,T. and Pedersen,C.N.S. (2004) QDist–quartet distance between evolutionary trees. *Bioinformatics*, **20**, 1636–1637.

Nielsen,J. *et al.* (2011) A sub-cubic time algorithm for computing the quartet distance between two general trees. *Algorithms Mol. Biol.*, **6**, 15.

Semple,C. and Steel,M. (2003) *Phylogenetics*. Vol. 24, Oxford University Press, New York.

Walker,R.S. *et al.* (2012) Cultural phylogenetics of the Tupi language family in Lowland South America. *PLoS One*, **7**, e35025.