Top-down phylogenetic tree reconstruction

Celine Vens[1], Eduardo Costa[1], and Hendrik Blockeel[1,2]

[1] Dept. of Computer Science, Katholieke Universiteit Leuven
[2] Leiden Institute of Advanced Computer Science, Universiteit Leiden
{Celine.Vens,Eduardo.Costa,Hendrik.Blockeel}@cs.kuleuven.be

Abstract. We propose a novel method for reconstructing phylogenetic trees. It is based on a conceptual clustering method that is an extension of the well-known decision tree learning approach. As such, the method differs from existing methods, both algorithmically and with respect to the information it uses and the assumptions it makes. It scales better in terms of the number of sequences, and as such can be used on large sets of sequences, beyond the point where other methods break down. By nature, it has a stronger tendency to construct trees in which subspecies are defined by particular polymorphisms. Experiments show that its performance is comparable to that of state-of-the-art methods.

1 Introduction

We consider the problem of deriving from a set of sequences the most likely phylogenetic tree. Many methods have been proposed for this. Among the most popular ones are maximal parsimony methods [1, 2], which aim at constructing the tree that explains the observed genetic variability with as few mutations as possible, and neighbor joining methods [3, 4], which, given an additive distance measure between sequences, construct the tree with minimal total branch length.

The neighbor joining algorithm [3] is essentially a so-called agglomerative hierarchical clustering algorithm: starting from one cluster per sequence, it iteratively merges clusters of sequences (merging the most similar ones first) until a single cluster is obtained; this yields a dendrogram that is interpreted as a phylogenetic tree.

While agglomerative clustering algorithms are among the most popular ones for clustering, many other clustering algorithms exist. Among these algorithms, one can distinguish extensional and conceptual clustering algorithms. In extensional clustering, a cluster is described by enumerating its elements, whereas in conceptual clustering, a cluster is described by listing properties of the cluster's elements, using a particular description language. This language constrains the clustering: a cluster that cannot be described in the language is not an allowable cluster.

In the context of phylogenetic analysis, a natural conceptual language would be one that refers to polymorphic locations; for instance, one cluster might be described as "all sequences having a C in position 72 and A in position 31".

With any extensional clustering method, it is of course possible to form a hierarchical clustering and next describe the clusters conceptually. However, if we assume that it must be possible to describe clusters using a particular language (based on polymorphisms), the space of all possible clusterings is greatly reduced. This enables us to use a so-called divisive clustering method, which starts from a single cluster and repeatedly divides it into subclusters until all sequences form a different cluster. Normally, given a set of N sequences, there are 2^N ways to split it into two subsets. But if we assume that the split can be described by referring to one particular polymorphic location, then the number of splits is linear in the length of the sequences, and constant in the size of the set. This makes such a divisive method computationally feasible, and potentially faster than agglomerative methods. A similar observation was made by [5], who were the first to propose a top-down clustering method for phylogenetic tree construction. Differences between their algorithm and our work are described in the related work section.

Blockeel et al. [6] discuss how a simple extension of decision tree learning leads to a divisive conceptual clustering algorithm. In this paper we investigate to what extent this approach lends itself to phylogenetic tree reconstruction.

2 The method

The conceptual clustering algorithm follows the standard decision tree induction (or: recursive partitioning) algorithm [7]. Basically, the algorithm works as follows. Given a dataset and a set of tests (where a test gives one of several possible outcomes when applied to a single data element), find the test that "optimally" partitions the dataset into subsets. Repeat this procedure on the subsets until subsets of size one are obtained.

In the context of phylogenetic tree construction, data elements are sequences, a test checks for the occurrence of a particular nucleotide or amino acid at a particular location, and an "optimal" partitioning is one that splits the set into subsets that are maximally genetically different from each other.

The computational complexity of a decision tree learning method is roughly $O(aN \log N)$ with a the number of tests and N the number of elements in the original dataset, under the assumption that a reasonably symmetric tree is built (the depth of which is logarithmic in the number of leaves) and the evaluation of a single test takes linear time in the size of the dataset. This scales much better in N than agglomerative methods, which have complexity $O(N^3)$ [4]. As such, such a divisive method may be much more efficient when analysing large sets of sequences.

There are several ways in which the quality of a single test can be evaluated. Intuitively, a test is better if the genetic distance between elements of the same subset is small and that between elements of different subsets is large. If a distance measure between sequences is given, one can simply measure the intra-subset and inter-subset variance (mean squared distance between sequences). Alternatively, one can use a criterion that is closer to the optimisation criterion

that neighbor-joining methods use, namely, constructing a phylogenetic tree with minimal total branch length. A disadvantage of the latter criterion is that it is no more linear, but quadratic, in the size of the dataset being subdivided, which increases the overal complexity of the method $O(aN^2 \log N)$.

The above algorithm (using the latter criterion) was implemented as a variant of the Clus decision tree learner[3]; we call the resulting system Clus-φ.

3 Evaluation

We have tested our alternative method for phylogenetic tree construction on a number of datasets. In a first set of experiments, we check how much Clus-φ trees differ from the ones returned by neighbor-joining (NJ). As a reference point, we include the difference between parsimony methods and NJ.[4]. We used the programs *neighbor* and *dnapars/protpars* from the Phylip software package [8], respectively.

In all experiments, we used the Jukes-Cantor correction formula [9] to compute the genetic distance between two DNA sequences, and the Jones-Taylor-Thornton matrix [10] for amino acid sequences. We measure differences between trees using the quartet distance [11]. This distance gives the number of quartets, i.e. subtrees induced by four leaves, that differ between two trees being compared.

Table 1 reports the quartet distance for 4 real datasets used in [12]. While the trees generated by Clus-φ are very similar to those generated by NJ, there are differences for 2 datasets. This variation is comparable to the variation between maximal parsimony and NJ.

Table 1. Quartet distance for real datasets.

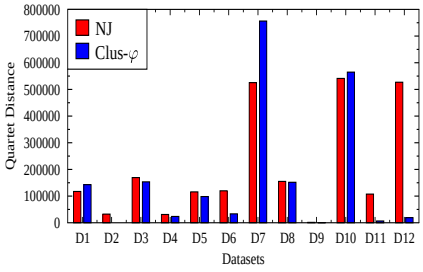
Dataset	Length	Size	NJ vs. Clus-φ	NJ vs. Pars
Chimp	896	5	0	2
hivALN	2352	14	0	156
mtDNA	1998	17	218	26
gdpAA	422	12	73	97.50

The question is then whether, in those cases where NJ and Clus-φ differ, any of them is more likely to be correct than the other one. To test this, we generated 24 datasets by simulating an evolutionary process, using the coding sequence simulation program EvolveAGene3 [13]. All the sequences that correspond to leaf nodes in the tree induced by this evolutionary simulation are used as input for phylogenetic tree reconstruction. By default, the software produces symmetric trees, i.e., binary trees that have all leaves at the same depth. However, also random tree topologies can be produced.

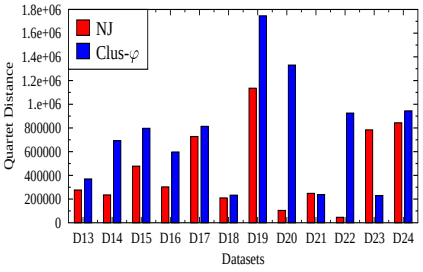
[3] `http://www.cs.kuleuven.be/~dtai/clus`.

[4] When parsimony analysis returns multiple trees, we report the average difference.

For these synthetic datasets, we can compare the trees that NJ and Clus-φ return to the known correct tree. Figure 1 shows the results separately for symmetric (left) and random (right) tree topologies. On average, NJ performs slightly better than Clus-φ; however the figure shows a large difference in the results for symmetric and random tree topologies. For symmetric trees[5], the Clus-φ tree is closer to the true tree than the tree produced by NJ in 9 out of 12 cases. For the random tree topologies, on the other hand, Clus-φ is closer to the true tree in only 2 cases; further, the differences are larger here. The reason for the difference in performance for the two topologies remains to be determined. The graphs also show a strong variation of tree quality with the dataset: the dataset has a larger influence on the overall performance than the choice of method.



(a)



(b)

D1	D2	D3	D4	D5	D6
1.22	**0**	**0.91**	**0.76**	**0.85**	**0.28**

D7	D8	D9	D10	D11	D12
1.44	**0.98**	**0.5**	1.04	**0.06**	**0.04**

(c)

D13	D14	D15	D16	D17	D18
1.34	2.95	1.67	1.97	1.12	1.11

D19	D20	D21	D22	D23	D24
1.54	12.79	**0.96**	1.41	**0.29**	1.12

(d)

Fig. 1. Quartet distance to real tree, for symmetric (left) and random (right) phylogenetic trees describing 128 DNA sequences of length 800. Graphs (a) and (b): quartet distance; Tables (c) and (d): quartet distance of Clus-ϕ relative to that of NJ.

Note that NJ and Clus-φ have a different bias with respect to tree construction. NJ provably constructs the tree with smallest possible branch length, if the real tree is purely additive, while Clus-φ uses a heuristic that makes it likely that it will approximate that tree, but there is no guarantee. On the other hand, Clus-φ is biased towards forming a phylogenetic tree where the divergence between subspecies can be attributed to particular polymorphic locations, which is not the case for NJ. Due to their different biases, NJ and Clus-φ can be expected to make mistakes independently. In such a situation, it is likely that both can be

[5] Similar results were obtained for *almost*-symmetric trees, where small perturbations have been made to the trees manually, increasing the tree depth with 1 or 2.

combined to form a single method that performs better than any of its separate components. The extent to which this holds still remains to be determined.

Finally, to analyze how NJ and Clus-φ scale in the number of sequences, synthetic datasets containing sequences of the same length (300 nucleotides), but increasing number of sequences were generated. Figure 2 shows run times for NJ and Clus-φ. Clus-φ is slower than NJ when analysing a relatively small number of sequences, but clearly outperforms NJ for large datasets. The break-even point in our experiments is around 850 sequences.

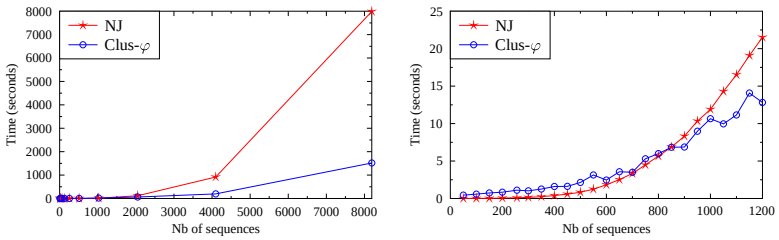


Fig. 2. Running times for Clus-φ and NJ; the right graph zooms in on the interval 0-1200 of the left graph.

4 Related work

Among other methods for phylogenetic tree construction, the most closely related one is the PTDC (Phylogeny by Top Down Clustering) algorithm [5]. The algorithm shares with our proposal the idea of recursively partitioning clusters by looking for the two subclusters with largest inter-cluster distance. The main differences are: (1) our approach makes the link to the decision tree learning framework, which allows us to profit from the highly developed state of the art in that area; (2) PTDC's heuristic maximises the average of all pairwise distances between sequences of different clusters, making it similar to the UPGMA algorithm, from which it inherits some undesirable characteristics such as sensitivity to unequal substitution rates in different lineages [14], whereas Clus-φ uses a heuristic that approximates the neighbor joining criterion; (3) while PTDC splits clusters based on equality of subsequences, Clus-φ creates splits based on a single, most informative, position; this gives it an efficiency advantage.

5 Conclusions

We have proposed a novel method for reconstruction of phylogenetic trees. The method differs strongly from existing methods, both from an algorithmic point

of view and from the point of view of the information it uses; it has a stronger tendency to construct a tree in which subspecies can be defined by referring to particular polymorphisms. The method scales much better in terms of the number of sequences, and remains feasible for sets of thousands of sequences, where other methods fail. Its performance is close to that of the state-of-the-art neighbor-joining method.

Acknowledgments

Celine Vens and Eduardo Costa are supported by the Research Foundation - Flanders (FWO Vlaanderen) and the GOA Probabilistic Logic Learning.

References

1. Kluge, A., Farris, J.: Quantitative phyletics and the evolution of anurans. Systematic Zoology **40** (1969) 446–457
2. Fitch, W.: Toward defining the course of evolution: Minimum change for a specific tree topology. Systematic Zoology **10** (1971) 406–416
3. Saitou, N., Nei., M.: The neighbor-joining method: a new method for reconstructing phylogenetic trees. Mol. Biol. Evol. **4**(4) (1987) 406–425
4. Studier, J., Keppler, K.: A note on the Neighbor-Joining algorithm of Saitou and Nei. Mol. Biol. Evol. **5**(6) (1988) 729–731
5. Arslan, A.N., Bizargity, P.: Phylogeny by top down clustering using a given multiple alignment. In: Proceedings of the 7th IEEE Symposium on Bioinformatics and Biotechnology (BIBE 2007), Vol. II. (2007) 809–814
6. Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: Proceedings of the 15th International Conference on Machine Learning. (1998) 55–63
7. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann series in Machine Learning. Morgan Kaufmann (1993)
8. Felsenstein, J.: Phylip (phylogeny inference package) version 3.68 (2008) http://evolution.genetics.washington.edu/phylip.html.
9. Jukes, T., Cantor, C.: Evolution of protein molecules. In Munro, H., ed.: Mammalian Protein Metabolism. Academic Press, New York (1969) 21–132
10. Jones, D., Taylor, W., Thornton, J.: The rapid generation of mutation data matrices from protein sequences. Computer Applications in the Biosciences **8** (1992) 275–282
11. Estabrook, G., McMorris, F., Meacham, C.: Comparison of undirected phylogenetic trees based on subtrees of four evolutionary units. Systematic Zoology **34** (1985) 193–200
12. Salemi, M., Vandamme, A.: The Phylogenetic Handbook: A Practical Approach to DNA and Protein Phylogeny. Cambridge University Press (2003)
13. Hall, B.: Simulating DNA Coding Sequence Evolution with EvolveAGene3. Molecular Biology and Evolution **25**(4) (2008) 688–695
14. Sourdis, J., Krimbas, C.: Accuracy of phylogenetic trees estimated from DNA sequence data. Molecular Biology and Evolution **4** (1987) 159–166