FastTree: Computing Large Minimum-Evolution Trees with Profiles instead of a Distance Matrix

by Morgan N. Price, Paramvir S. Dehal, and Adam P. Arkin

Supplementary Table 1 - Topological accuracy of tree-building methods on simulated nucleotide alignments of 500 nucleotides without gaps.

Guindon and Gascuel (2003) simulated 2,000 random trees with a maximum pairwise divergence of roughly 0.4 or 1.0 substitutions/site and with significant, yet realistic, deviations from the molecular clock. They generated alignments according to the Kimura two-parameter model with a transition/transversion ratio of 2, without gaps, and without varying evolutionary rates across sites (http://www.lirmm.fr/ guindon/simul/). We measured the performance of FastTree, BIONJ+dnadist, Neighbor, and Clearcut ourselves. (Neighbor is the implementation of neighbor joining in the phylip package.) The other results are from "Shortest triplet clustering: reconstructing large phylogenies using representative sets" (L. Vinh and A. von Haeseler, BMC Bioinformatics 2005, 6:92).

Method	**Distances**	**N=24** $d \approx 0.4$	**N=24** $d \approx 1.0$	**N=96** $d \approx 1.0$
FastTree	Jukes-Cantor	0.921	0.926	0.912
FastME	Kimura	–	–	0.910
BIONJ	dnadist (F84)	0.916	0.921	0.888
BIONJ	Kimura	–	–	0.888
Neighbor	dnadist (F84)	0.914	0.918	0.885
NJ	Kimura	–	–	0.885
Clearcut	dnadist (F84)	0.893	0.897	0.874

Supplementary Table 2 - Topological accuracy of tree-building methods on simulated protein alignments with 10, 50, 250, or 1,250 sequences and no gaps.

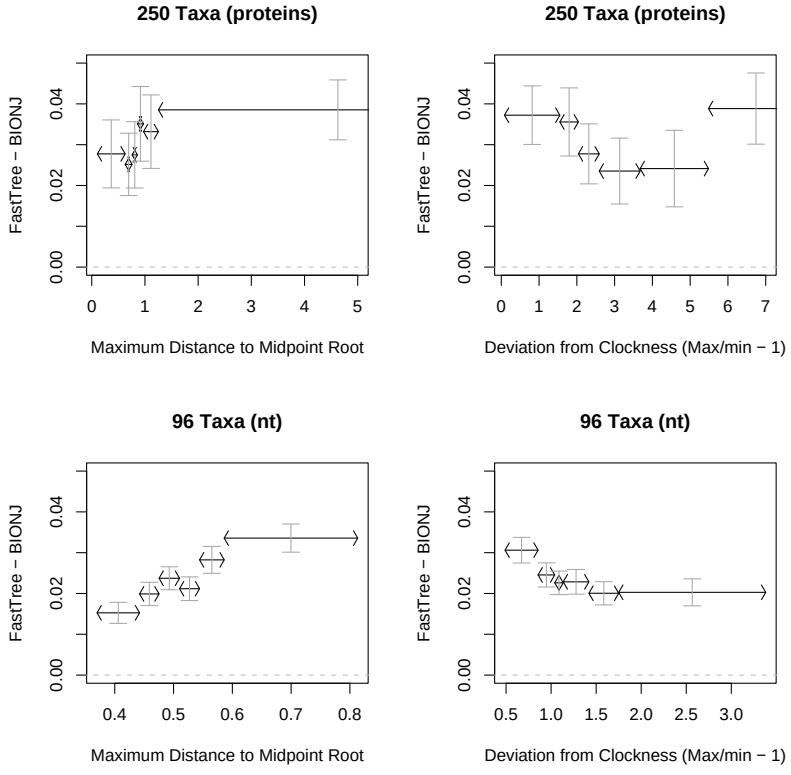
Method	Distances	N=10	N=50	N=250	N=1,250
PhyML	JTT	0.768	0.797	0.837	0.827
FastTree	log-corrected	0.752	0.800	0.841	0.827
FastME	log-corrected	0.742	0.796	0.839	0.828
BIONJ	log-corrected	0.752	0.797	0.817	0.788
BIONJ	JTT	0.714	0.794	0.831	0.799
BIONJ	JTT + Γ	0.564	0.624	0.782	0.766
QuickTree	log-corrected	0.746	0.788	0.810	0.781
QuickTree	%different	0.692	0.708	0.726	0.703
Clearcut	log-corrected	0.702	0.774	0.802	0.777
FastTree	No NNI	–	–	0.765	0.740
FastTree	Exhaustive, no NNI	–	–	0.764	0.738
BIONJ	uncorrected	–	–	0.764	0.738

Supplementary Table 3 - Relative log-likelihoods of trees inferred by variants of FastTree for genuine protein alignments of 500 sequences.

Method	Average Log-Lik.	Lower Lik. than FastTree
FastTree, Default settings	0.0	–
FastTree, Balanced joins	-106.4	77%
FastTree, No NNI	-402.3	100%
FastTree, Exhaustive, No NNI	-428.5	100%
BIONJ, uncorrected dist.	-514.1	>99%

Supplementary Figure 1: Relative performance of FastTree and BIONJ for fast-evolving or non-clock-like simulations

To test the relative accuracy of FastTree and BIONJ across a range of tree shapes, we subdivided the simulations by their maximum evolutionary divergence (left panels) or by their deviation from the molecular clock (right panels). Because these distributions are skewed, we split them into bins of equal size. On the x axis, each horizontal arrow shows the range of 1/6th of the simulations. The y axis shows the accuracy advantage of FastTree over BIONJ for that group of simulations, and each vertical error bar shows a 95%-confidence interval from a t test. The protein simulations (top panels) are with gaps (as in Table 1) and the nucleotide simulations (bottom panels) are from Desper and Gascuel (2002) with accelerated evolutionary rates (maximum pairwise divergence ≈ 1, as in Supplementary Table 1). To compute the divergence for these (unrooted) trees, we used the midpoint root and we measured the maximum distance from the midpoint. We quantified the deviation from clockness as the dispersion of distances from the midpoint (max/min - 1). We ran BIONJ with log-corrected distances for proteins and with dnadist (F84) distances for nucleotides.



Supplementary Note 1: Technical Details of FastTree

Distances between Nodes – No Gaps

Here we show that, if there are no gaps, then the neighbor-joining phase of FastTree, with exhaustive search for the best join, gives identical results to BIONJ with uncorrected distances.

When we join two nodes i and j, we store a profile or a frequency vector at each position l for the new parent node ij as the (weighted) average

$$\vec{P}_l(ij) = \lambda \vec{P}_l(i) + (1-\lambda)\vec{P}_l(j)$$

where λ is the weight (as in BIONJ) or $\lambda = 1/2$. Because the profile distances $\Delta(i,j)$ are linear,

$$\Delta(ij,k) = \lambda\Delta(i,k) + (1-\lambda)\Delta(j,k)$$

We first consider unweighted joins. To compute distances between joined nodes and other nodes, neighbor joining uses the formula

$$d_u(ij,k) = \frac{d_u(i,k) + d_u(j,k) - d_u(i,j)}{2}$$

and FastTree uses profiles and up-distances

$$d_u(ij,k) = \Delta(ij,k) - u(ij) - u(k)$$

$$u(ij) \equiv \frac{\Delta(i,j)}{2}$$

and $u(l) \equiv 0$ for leaves. For two leaves i and j, $\Delta(i,j) = d_u(i,j)$, so this gives the correct distance between leaves. Assume the distances are correct for all nodes so far and consider the next join ij:

$$\begin{aligned} d_u(ij,k) &= \frac{d_u(i,k) + d_u(j,k) - d_u(i,j)}{2} \\ &= \frac{\Delta(i,k) - u(i) - u(k) + \Delta(j,k) - u(j) - u(k) - \Delta(i,j) + u(i) + u(j)}{2} \\ &= \frac{\Delta(i,k) + \Delta(j,k)}{2} - \frac{\Delta(i,j)}{2} - u(k) \\ &= \Delta(ij,k) - u(ij) - u(k) \end{aligned}$$

which shows that our distances are correct for $d_u(ij,k)$ and, by induction, for all nodes.

For weighted joins, a similar argument shows that

$$u(ij) \equiv \lambda(u(i) + d_u(i,ij)) + (1-\lambda)(u(j) + d_u(j,ij))$$

$$d_u(i, ij) \equiv \frac{d_u(i,j) + r(i) - r(j)}{2}$$

where $r(i)$ is the "out-distance," gives the same result as the formula for BIONJ

$$d_u(ij, k) = \lambda(d_u(i,k) - d_u(i,ij)) + (1-\lambda)(d_u(j,k) - d_u(j,ij))$$

For BIONJ, we also need to compute the "variances" that are used to computed the weights of the joins. The variance values for pairs of leaves are the same as the distance values, or $V(l_1, l_2) = d_u(l_1, l_2)$, and the BIONJ formula for variances of internal nodes is

$$V(ij, k) = \lambda V(i,k) + (1-\lambda)V(j,k) - \lambda(1-\lambda)V(i,j)$$

where λ is the weight of i in ij. These variance values can be computed from profile-distances by using a "variance correction" $\nu(i)$ analogous to the up-distances. Let $\nu(i) \equiv 0$ for leaves and

$$V(i,j) = \Delta(i,j) - \nu(i) - \nu(j)$$

$$\nu(ij) \equiv \lambda\nu(i) + (1-\lambda)\nu(j) + \lambda(1-\lambda)V(i,j)$$

Given these variances, BIONJ weights the join of i, j so as to minimize the variance of the distance estimates for the new node ij, using the formula

$$\lambda = \frac{1}{2} + \frac{\sum_{k \neq i,j} (V(j,k) - V(i,k))}{2(n-2)V(i,j)}$$

FastTree computes the numerator with

$$\begin{aligned}\sum_{k \neq i,j} (V(j,k) - V(i,k)) &= \sum_{k \neq i,j} (\Delta(j,k) - \Delta(i,k) - \nu(j) + \nu(i) + \nu(k) - \nu(k)) \\ &= (n-2)(\nu(i) - \nu(j)) + \sum_{k \neq i,j} \Delta(j,k) - \sum_{k \neq i,j} \Delta(i,k)\end{aligned}$$

where n is the number of active nodes before the join takes place. FastTree computes this in $O(La)$ time by using the total profile to compute sums of $\Delta(i,k)$, as explained in the next section.

Out-distances

For either neighbor joining with unweighted joins or BIONJ, we need to compute the out-distances $r(i)$ so that we can compute the neighbor-joining criterion

$$d_u(i,j) - r(i) - r(j)$$

where

$$r(i) \equiv \frac{\sum_{j \neq i} d_u(i,j)}{n-2}$$

In the absence of gaps, the average profile distance between a node and all other nodes can be inferred from the total profile T:

$$\begin{aligned}\sum_{j \neq i} d_u(i,j) &= \sum_{j \neq i} (\Delta(i,j) - u(i) - u(j)) \\ &= \sum_j \Delta(i,j) - \Delta(i,i) - (n-1)u(i) - \sum_{j \neq i} u(j) \\ &= n\Delta(i,T) - \Delta(i,i) - (n-1)u(i) + u(i) - \sum_j u(j)\end{aligned}$$

which can be computed in $\mathrm{O}(La)$ time if we store the total profile T and the total of all the up-distances.

Handling Gaps

Now, consider what happens if there are gaps. The profiles' frequency vectors do not include gaps, but the profile distance is weighted at each position:

$$\Delta(A,B) \equiv \frac{\sum_{l=1}^{L} \Delta_l(A,B) w_l(A) w_l(B)}{\sum_{l=1}^{L} w_l(A) w_l(B)}$$

$$\Delta_l(A,B) \equiv \sum_\alpha \sum_\beta f_{Al}(\alpha) f_{Bl}(\beta) D(\alpha,\beta)$$

where D is the dissimilarity matrix on characters, $f_{Al}(\alpha)$ is the frequency of character α in the profile of A at position l, Δ_l is the distance between the profiles at position l, and $w_l(A)$ is the proportion of non-gaps in the profile of A at position l. The proportion of non-gaps for an internal node is just the weighted average of the values for its children.

In the presence of gaps, the previous formula for the out-distance is not a good approximation because highly gapped sequences contribute little to the total profile. Instead, we need to take the weights of the comparisons into account. Let $T - i$ be the total profile with the contribution from i removed. Then

$$\sum_{i \neq j} \Delta(i,j) \approx (n-1)\Delta(i, T-i)$$

$$\Delta(i, T-i) = \frac{\sum_{j \neq i} \sum_{l=1}^{L} \Delta_l(i,j) w_l(i) w_l(j)}{\sum_{j \neq i} \sum_{l=1}^{L} w_l(i) w_l(j)}$$

$$\Delta(i, T) = \frac{\sum_{j} \sum_{l=1}^{L} \Delta_l(i,j) w_l(i) w_l(j)}{\sum_{j} \sum_{l=1}^{L} w_l(i) w_l(j)}$$

which leads to a formula for $\Delta(i, T-i)$ in terms of $\Delta(i, T)$ and $\Delta(i, i)$ and the total weights of those comparisons. In practice, this gives a good approximation for the out-distances in the presence of gaps (data not shown).

Gaps also complicate the interpretation of the "variances" used for weighted joins. In principle, the variances should be divided by the number of non-gap positions in the comparison, as distances that are computed from more positions are more reliable. However, if we do that, the formula for variances given in the previous section becomes unreliable (data not shown). Instead, we implicitly weight less-gapped sequences more highly because the less-gapped member of a join contributes more strongly to the profile.

To compute the weight for each join in the presence of gaps, we need to estimate $\sum_{k \neq i,j} \Delta(i,k)$. To do this, split the profile-distance into its numerator and its weight or denominator

$$\Delta(i,j) = \frac{N(i,j)}{w(i,j)}$$

Then

$$\begin{aligned} \sum_{k \neq i,j} \Delta(i,k) &\approx \sum_{k \neq i,j} \Delta(i,k) \frac{(n-2) \cdot w(i,k)}{\sum_{k \neq i,j} w(i,k)} \\ &= \frac{(n-2) \cdot N(i, T-i-j)}{w(i, T-i-j)} \end{aligned}$$

where $T - i - j$ represents the total profile with i and j removed, and the terms can be computed with

$$N(i, T-i-j) = n \cdot N(i,T) - N(i,i) - N(i,j)$$

$$w(i, T-i-j) = n \cdot w(i,T) - w(i,i) - w(i,j)$$

Restrictions on the Top-hits Heuristic

To ensure accuracy, FastTree sorts the sequences before computing the top-hits lists, and it restricts the top-hits heuristic to "close enough" sequences. FastTree sorts the sequences by how many gaps they contain and by their out-distances. Fewer gaps is better, as this makes the distances more reliable, and a smaller out-distance is better, as this implies that the sequence is more likely to be close to other nodes.

Before it estimates the top-hits of B from the top-hits of A, FastTree requires that $d_u(A,B) \leq 0.75 \cdot d_u(A, H_{2m})$, where H_{2m} is A's $2m$-th best hit. (Note that we are using uncorrected distances here, not the neighbor-joining criterion, even though the top-hits lists are sorted by the neighbor-joining criterion.) This restriction applies to the initial computation of top hits but not to the "refreshes" that are used to update the top hits during neighbor joining. The factor of 0.75 represents a compromise between speed and accuracy. If we used a factor of 0.5, then the top-hits list should include the true top hit because of the triangle inequality $d_u(A,B) \leq d_u(A,C) + d_u(C,B)$. On the other hand, for a perfectly balanced tree with clock-like evolution, we expect the distance of hit m to be proportionate to $\log_2 m$ and the distance of hit $2m$ to proportionate to $1 + \log_2 m$, so with a factor of $\log_2 m/(1 + \log_2 m)$ we would expect $O(m)$ close neighbors of A.

FastTree also requires that A and B have similar patterns of gaps. This is to avoid cases where the sequences overlap in only a few positions, so that they might be identical or nearly so (for the positions considered) even though they have very different top-hits lists. Specifically, we require that the total number of non-gap positions in the comparison between A and B must be at least $1 - d_u/2$ times the number of non-gap positions in B or at least $1 - 2d_u/3$ times the average number of shared positions between A and its top $2m$ hits, where d_u is the maximum distance allowed between A and B.

Refreshing the Top-hits Lists

If the top-hit list has shrunk too much or is too old, then FastTree does a "refresh" – it recomputes the top-hit lists for the new joined node and updates the top-hit lists of the new node's top hits. A top-hit list is too short if it has less than $0.8m$ entries, where 0.8 is a tuning parameter and m is the size of the top-hit lists. FastTree also records a top-hits "age" of each join, and it sets the age of a new node to one plus the maximum of its childrens' ages. If the age is above $1 + log_2(m)$ then it does a refresh.

To refresh, we compare the new node AB to all $n-1$ other active nodes, and we save the top m hits. Then, we compare the close neighbors of AB (the top m hits) to the top $2m$ hits of AB, and we update the close neighbors' top-hit lists by merging. The age of every updated top-hits list is set to zero. A single refresh takes $\mathrm{O}(nLa + m^2La) = \mathrm{O}(NLa)$ time and ensures that the top-hit lists of $\mathrm{O}(m = \sqrt{N})$ other nodes reach size m and are up to date. Thus, there are $\mathrm{O}(\sqrt{N})$ refreshes and they take a total of $\mathrm{O}(N\sqrt{N}La)$ time. We also note that

if the top hit list has size equal to the remaining number of active nodes (e.g. because $n \leq m$), then the top-hit lists are exhaustive. At that point there is only $O(m^2La = NLa)$ work remaining.

Compact Representation of Profiles

To save memory, FastTree represents sequences as characters (1 byte per position) rather than profiles ($4a + 4$ bytes per position, because it stores a single-precision floating-point value for each eigenfrequency plus an overall weight). Similarly, when FastTree computes a profile that is constant at a position (or all-gaps, with weight=0), it stores a character and a weight, but not the eigenfrequency vector. This requires an additional overhead of 1 byte per position for varying positions, but still yields a large savings in space requirements, especially for amino acid alignments ($a = 20$).

Supplementary Note 2 – Technical Details of Testing FastTree

To generate simulated protein alignments, we used the 310 COGs with at least 1,000 distinct sequences in MicrobesOnline. Given an actual family and alignment, we removed duplicate sequences, we selected the desired number of family members at random, and we removed alignment positions that were over 25% gaps. For alignments of up to 1,250 sequences, we inferred a maximum-likelihood phylogeny using PhyML 3.0 with the JTT model and no rate variation across sites (Guindon and Gascuel 2003). Given the topology, we inferred the evolutionary rate of each site using proml from the phylip package, gamma-distributed rates (8 categories), and a coefficient of variation of 1 (http://evolution.genetics.washington.edu/phylip.htm). The inferred rate categories were biased downwards (the average rate was less than one), so we normalized the rates so that their average was 1. We then used the branch lengths (from proml) and the evolutionary rate of each site to simulate an ungapped alignment with Rose (Stoye et al., 1998). Finally, we re-introduced the gaps that were in the genuine alignment so that the simulated alignment had the same pattern of gaps. Thus, both the topology and the placement of gaps should be realistic.

For simulations of 5,000 sequences, the above approach was not computationally feasible. (Even for 1,250 sequences, many of the PhyML jobs had not completed after a week.) Instead, we inferred the topology and branch lengths using FastTree and we assigned the evolutionary rate of each site randomly among 16 categories. These categories approximated a gamma distribution with a coefficient of variation of 0.7. For comparison, the coefficient of variation of the inferred rates for the genuine alignments of 10-250 sequences was typically 0.6-0.8. Also, before selecting sequences from the genuine alignment, we made a 99% non-redundant subset of sequences with CD-HIT (W. Li et al.,

Bioinformatics 18:77-82). This avoided inferring a tree with many very-short branch lengths and hence simulating large numbers of non-unique sequences

To compare the inferred trees to the true trees, we used perl scripts or phylip's treedist. So that treedist would run on trees of 5,000 leaves, we modified treedist.c to set maxgrp=100,000. (Without this change, it crashes on large trees.) Similarly, to run consense on resamples of the COG2814 and PF00005 trees, we modified consense.c to set maxgrp=1,000,000. This initializes its hashtable, which it uses to look up splits, to roughly 30-fold larger than the standard setting, and should speed it up.

To test tree inference on large genuine alignments, we used FastTree 1.0.0, BIONJ from http://www.lirmm.fr/~w3ifa/MAAS/BIONJ/BIONJ.c, QuickTree 1.1, Clearcut 1.0.8, RapidNJ 1.0.0, FastME 1.1, PhyML 3.0, RAxML VI version 1.0, and protdist and seqboot from version 3.65 of the phylip package. We also tried to run quick-join 1.0.10 (T.Mailund et al., BMC Bioinformatics 2006) and scoredist (Sonnhammer and Hollich 2005) on COG2814, but these required too much memory. Executables were obtained from the authors' web sites or were compiled with gcc version 3.4.6 and the -O2 optimization setting. When using protdist to estimate maximum likelihood distances, we replaced negative distances (from non-overlapping sequences) and distances above 3 substitutions per site with 3, as in scoredist. This improved the likelihood of the trees inferred by BIONJ from protdist distances.

To test the accuracy of the local bootstrap, we used FastTree 1.0.4.