# FastTree 2 – Approximately Maximum-Likelihood Trees for Large Alignments

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

Presented by Arjun P. Athreya
April 21, 2015
CS 598AGB

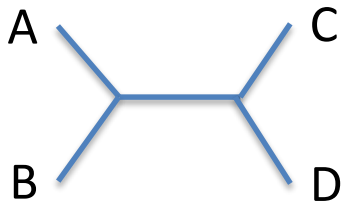
ECE ILLINOIS

ILLINOIS

# Fast Tree 2

- Five stages of computation
  - Heuristic neighbor-joining (NJ)
  - Tree length reductions
    - Nearest-neighbor interchanges (NNI)
    - Subtree-prune-regraft (SPR) moves
    - Distance model
  - Maximum Likelihood with NNIs
  - Local support values

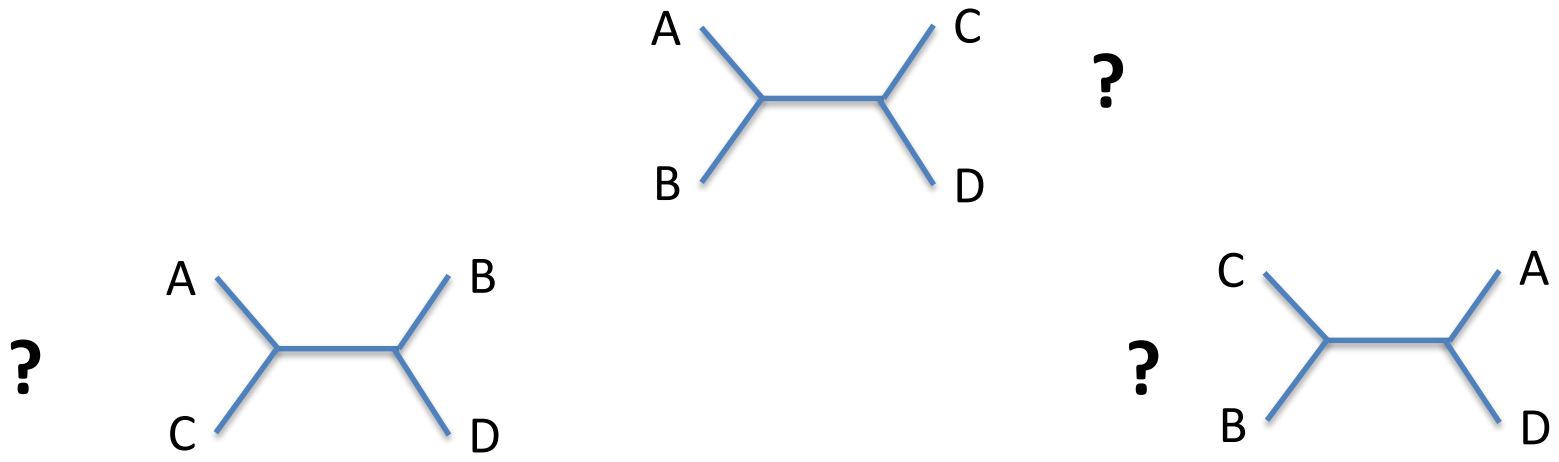
# Heuristic NJ




- Produces rough topology
- Optimization:
  - Profile for internal nodes instead of a distance-matrix (space saving!)
  - Remembers best join for each node
  - Remembers top pair-wise distances (space saving!)
  - Updates best join for a node as it traverses

# Tree-length reductions : NNI

- Topology refinement

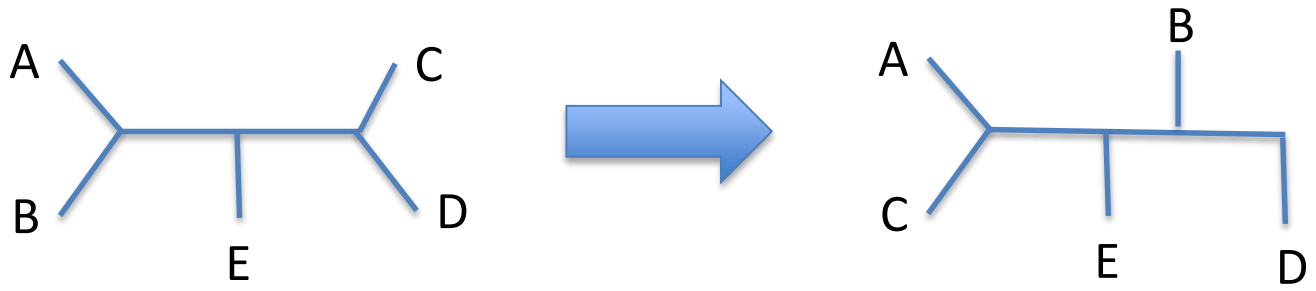



- Optimization:
  - work with profiles, than pairwise distances (space saving!)
  - 2 log(N) rounds of NNI

- Space: $O(NLa+N\sqrt{N})$ Time: $O(N\sqrt{N}\log(N)La)$

# SPR moves

- A subtree is removed from the tree, reinserted somewhere else




- Optimization:
  - Consider shortest SPRs first, and then extends the promising candidates (space savings!)
  - For each subtree, only two SPR moves (time saving!)

# Maximum Likelihood

- Improve tree-topology and branch lengths
- Jukes-Cantor model, accounts for variable rates (20 categories, geometrically distributed)
- Operation:
  - Likelihood of trees generated using NNI
  - Estimate branch lengths
- Optimizations:
  - Stop NNI if likelihood of rearrangements are not improving
  - NNI restricted to 2log(N)
  - Skip SPR in parts of tree that did not improve in recent rounds

# Results:

Metric: RF distances

FastTree outperforms other tools which don't use SPR's

**Table 1.** Topological accuracy of trees inferred from simulated alignments.

| | 250 | 1,250 | 5,000 | 78,132 |
|---|---|---|---|---|
| **Method** | **a.a.** | **a.a.** | **a.a.** | **nt.** |
| RAxML 7 (JTT+CAT, SPRs) | 90.5% | 88.4% | 88.4% | – |
| PhyML 3.0 (JTT+$\Gamma_4$, SPRs) | 89.9% | – | – | – |
| FastTree 2.0.0 (JTT+CAT or JC+CAT) | 86.9% | 83.7% | 84.3% | 92.1% |
| PhyML 3.0 (JTT+$\Gamma_4$, no SPRs) | 86.0% | – | – | – |
| FastME 2.06 (log-corrected distances, SPRs) | 80.5% | 78.8% | 77.0% | – |
| FastTree 2.0.0, no ML NNIs | 80.4% | 78.3% | 76.6% | 91.4% |
| BIONJ (ML distances) | 77.7% | 73.7% | 73.1% | – |
| Parsimony (RAxML) | 76.8% | 76.5% | 69.4% | – |
| Neighbor joining (log-corrected distances) | 76.0% | 72.6% | 71.6% | 66.1% |
| Clearcut (log-corrected distances) | 75.5% | 72.3% | 71.5% | 58.1% |

For alignments with 5,000 sequences, we used RAxML 7.2.1 with fast convergence; for smaller alignments we used RAxML 7.0.4. An earlier version of PhyML 3 took up to 4 days for individual simulations with 1,250 sequences, even without $\Gamma_4$, so we did not try to run PhyML 3 with $\Gamma_4$ on the larger simulations.

doi:10.1371/journal.pone.0009490.t001

# Results: likelihoods on biological data

- RAxML still better
- Exhaustive ML search still wins

**Table 2.** Average log-likelihood for genuine alignments with 500 sequences.

| Method | 16S | COG |
|---|---|---|
| RAxML 7.0.4 (GTR+CAT or JTT+CAT, SPRs) | −168,104 | −206,724 |
| FastTree 2.0.0 (GTR+CAT or JTT+CAT) | −168,577 | −206,993 |
| PhyML 3.0 (GTR+$\Gamma_4$ or JTT+$\Gamma_4$, no SPRs) | −168,603 | −207,156 |

For all topologies, the log likelihood was computed with RAxML 7, re-optimized branch lengths and model parameters, and the GTR+$\Gamma_4$ or JTT+$\Gamma_4$ models for 16S or COG, respectively. All differences between FastTree and other methods were statistically significant ($P < 10^{-10}$) except for the comparison with PhyML on 16S rRNAs ($P = 0.07$, paired $t$ test).
doi:10.1371/journal.pone.0009490.t002

# Results: RAxML vs FastTree2

- But, FastTree found 96-98% of splits RAxML found
- Heuristics did not affect the results much and performed as expected compared to simulated data

**Table 3.** Comparison of RAxML and FastTree's log likelihoods, and the agreement of FastTree with RAxML's well-supported splits, for large genuine alignments.

| | 16S rRNA | 16S rRNA | 7 COGs |
|---|---|---|---|
| Number of sequences | 4,114 | 6,718 | 2,500 |
| RAxML 7's Log Likelihood | −325,581 | −481,259 | −1,238,666 |
| FastTree 2's Log Likelihood | −328,062 | −493,841 | −1,240,916 |
| Difference | 2,481 | 12,582 | 2,251 |
| Well-supported RAxML splits (bootstrap ≥ 0.9) | | | |
| Total in RAxML tree | 851 | 1,124 | – |
| Found by FastTree | 837 | 1,075 | – |
| Weakly-supported RAxML splits (bootstrap 0.8–0.9) | | | |
| Total in RAxML tree | 265 | 419 | – |
| Found by FastTree | 250 | 365 | – |
| Locally-supported RAxML splits (SH ≥ 0.95) | | | |
| Total in RAxML tree | 1,336 | 1,927 | 1,018 |
| Found by FastTree | 1,033 | 1,319 | 889 |

We ran RAxML with the fast convergence option. All values for COGs are averages over seven families. Log likelihoods for all topologies were computed with RAxML using $\Gamma_4$ and GTR or JTT. Global bootstrap values are from using the standard bootstrap with RAxML 7.0.4 (from [11]). SH-like local support values for RAxML's topology were computed with FastTree 2, the CAT approximation, and GTR or JTT.

doi:10.1371/journal.pone.0009490.t003

# Results: Runtime

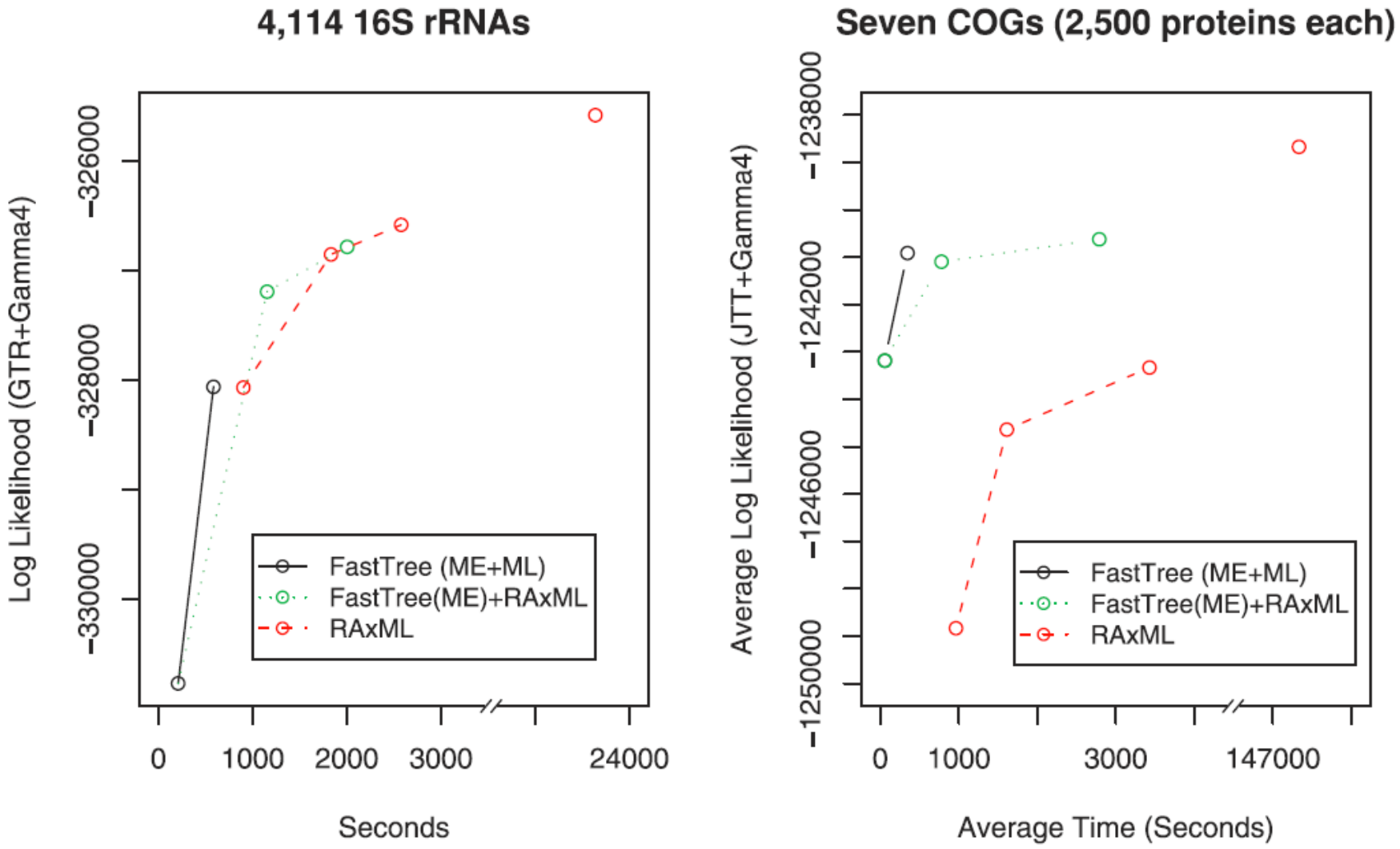
**Table 4.** Running time and memory usage on genuine alignments.

| | Distinct | | FastTree 2.0.0 | | | RAxML 7 | PhyML 3 |
|---|---|---|---|---|---|---|---|
| **Alignment** | **Sequences** | **Positions** | **Model** | **Hours** | **GB** | **Hours** | **Hours** |
| 16S rRNA, subsets | 500 | 1,287 nt. | GTR | 0.02 | – | 2.2 | 2.9 |
| COGs, subsets | 500 | 65–1,009 a.a. | JTT | 0.02 | – | 5.2 | 7.2 |
| COGs, subsets | 2,500 | 197–384 a.a. | JTT | 0.11 | – | 61 | – |
| Efflux permeases | 8,362 | 394 a.a. | JTT | 0.25 | 0.35 | 197 | >1,200 |
| 16S rRNAs, families | 15,011 | 1,287 nt. | GTR | 0.66 | 0.56 | 64 | >2,000 |
| ABC transporters | 39,092 | 214 a.a. | JTT | 1.02 | 0.96 | – | – |
| 16S rRNAs, all | 237,882 | 1,287 nt. | JC | 21.8 | 5.8 | – | – |

All runs used a single thread of execution. All runs accounted for variable rates across sites, using CAT for RAxML 7 and FastTree 2 or $\Gamma_4$ for PhyML 3. All FastTree runs include local SH-like supports and all RAxML runs include branch lengths under $\Gamma_4$. RAxML and PhyML were run without support values (no bootstrap). For random subsets of 500 16S rRNAs or for COGs, we show average running times. For alignments with over 1,000 sequences, we used RAxML 7.2.1's fast convergence option.
doi:10.1371/journal.pone.0009490.t004

Would take years!

# Results: Likelihood over time




RAxML with same starting tree as FastTree shows similar improvement in likelihood with time

# Conclusion

- FastTree2 makes intelligent decisions on improving speed while maintaining pretty good accuracy
- Impact of heuristics, computational tricks do not impact results a lot
- RAxML is still a winner for accuracy, but at the cost of time (may never complete for large datasets)
  - Personal experience on running FastTree 2 and RAxML for course project, 1 minute vs 30 minutes on small amino acid data