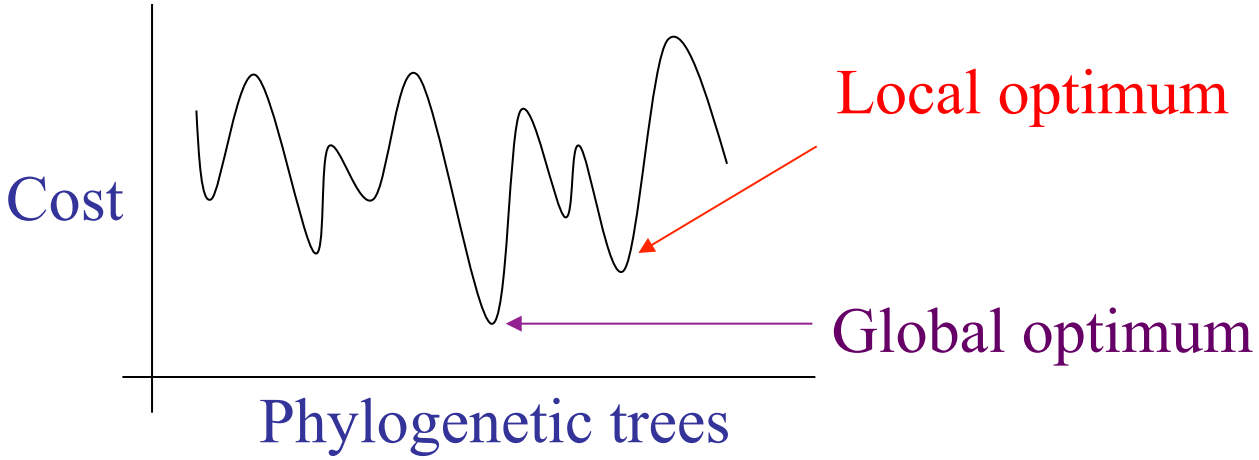
394C: Algorithms for Computational Biology

Tandy Warnow

Sept 9, 2013

Phylogenetic reconstruction methods

1. Hill-climbing heuristics for hard optimization criteria (Maximum Parsimony and Maximum Likelihood)



2. Polynomial time distance-based methods: UPGMA, Neighbor Joining, FastME, Weighbor, etc.

Performance criteria

- Running time.
- Space.
- Statistical performance issues (e.g., statistical consistency) with respect to a Markov model of evolution.
- "Topological accuracy" with respect to the underlying ***true tree.*** Typically studied in simulation.
- Accuracy with respect to a particular criterion (e.g. tree length or likelihood score), on real data.

How can we infer evolution?

While there are more than two taxa, DO

- Find the "closest" pair of taxa and make them siblings
- Replace the pair by a single taxon

Note: the input is a dissimilarity matrix, and you need to specify how to update the matrix after you replace two taxa by one taxon.

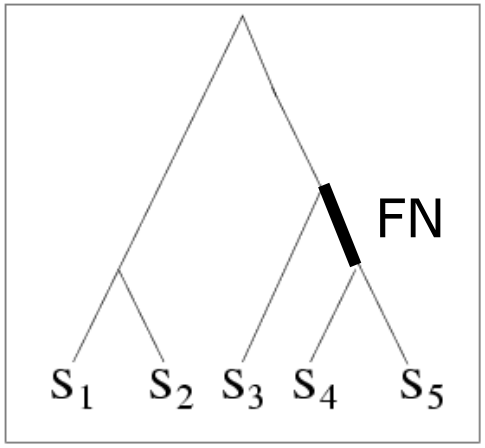
Updating the matrix

- How do we update the dissimilarity matrix, after we make two nodes x and y siblings?
- Various options, but here's one:
  - Replace the pair of siblings by a new node "xy".
  - For each remaining taxon v in the matrix, set $D(xy,v) = \frac{1}{2} (D(x,v) + D(y,v))$

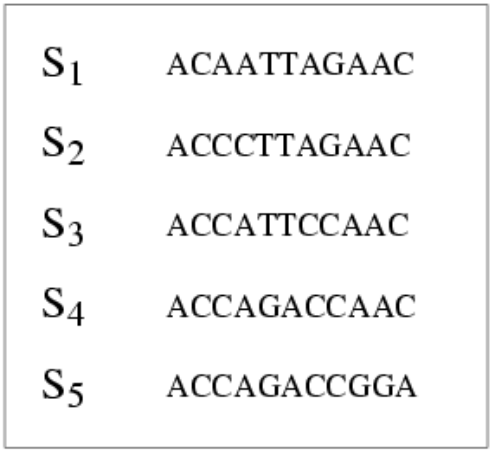
That was called "UPGMA"

- Advantages: UPGMA is polynomial time and works well under the "strong molecular clock" hypothesis.
- Disadvantages: UPGMA does not work well in simulations, perhaps because the molecular clock hypothesis does not generally apply.
- Other polynomial time methods, also distance-based, work better. One of the best of these is Neighbor Joining.

Quantifying Error



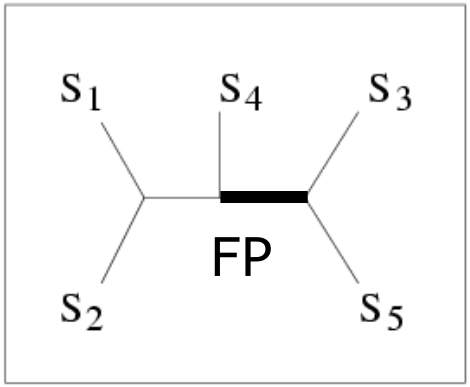
TRUE TREE



DNA SEQUENCES

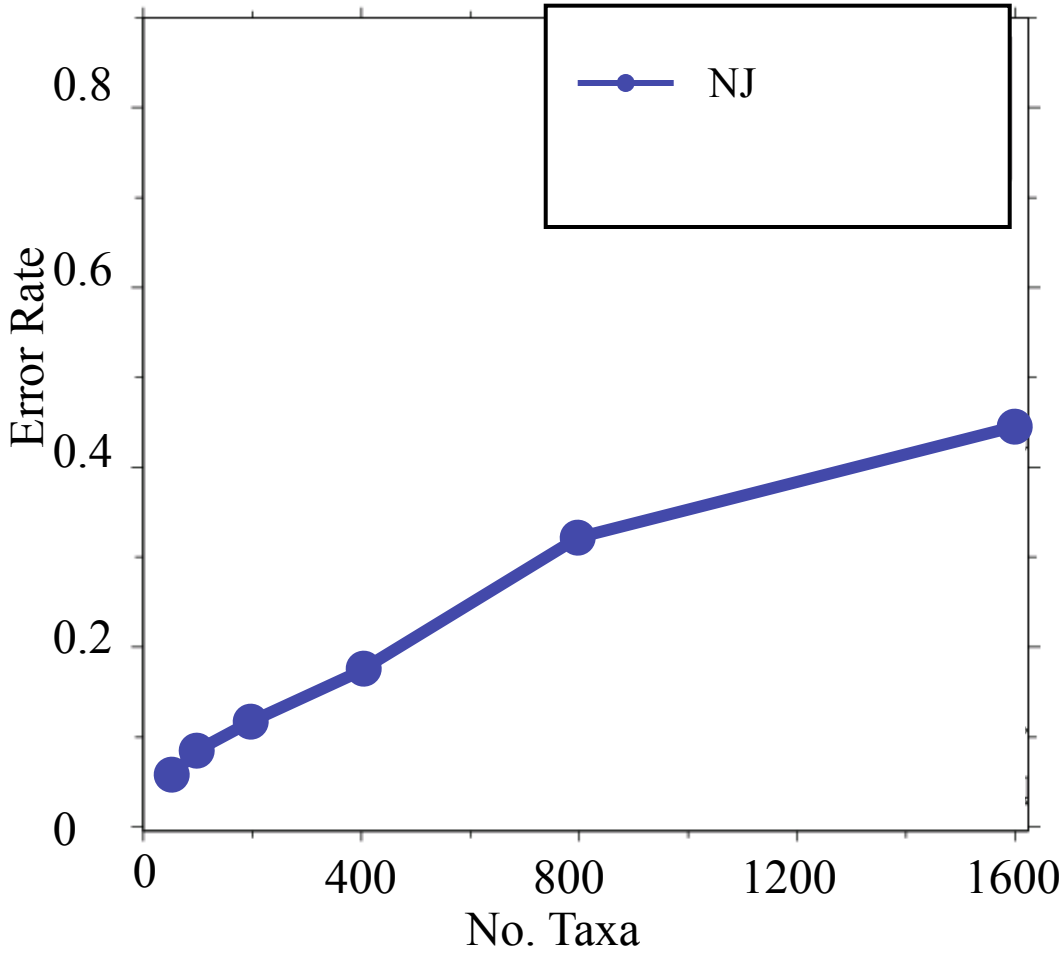
FN: false negative
(missing edge)
FP: false positive
(incorrect edge)

50% error rate



INFERRED TREE

Neighbor joining has poor performance on large diameter trees *[Nakhleh et al. ISMB 2001]*



Simulation study based upon fixed edge lengths, K2P model of evolution, sequence lengths fixed to 1000 nucleotides.

Error rates reflect proportion of incorrect edges in inferred trees.

- Other standard polynomial time methods don't improve substantially on NJ (and have the same problem with large diameter datasets).

- What about other approaches?

Maximum Parsimony

- **Input**: Set *S* of *n* aligned sequences of length k
- **Output**:
  - A phylogenetic tree T leaf-labeled by sequences in *S*
  - additional sequences of length *k* labeling the internal nodes of *T*

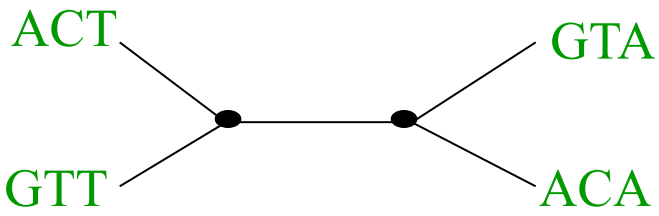
such that $\sum_{(i,j)\in E(T)} H(i,j)$

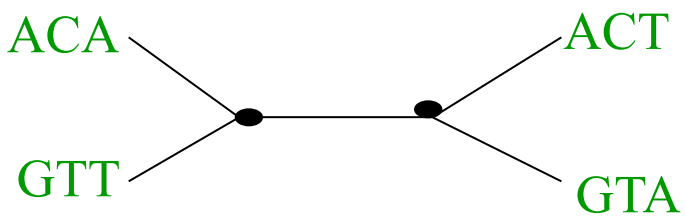
is minimized, where H(i,j) denotes the Hamming distance between sequences at nodes i and j

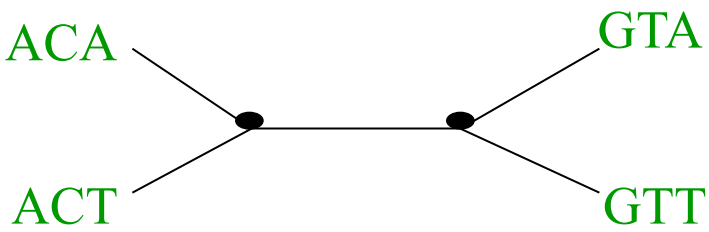
Maximum parsimony (example)

- **Input**: Four sequences
  - ACT
  - ACA
  - GTT
  - GTA
- **Question**: which of the three trees has the best MP scores?

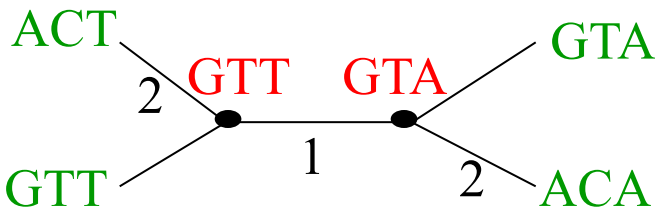
Maximum Parsimony



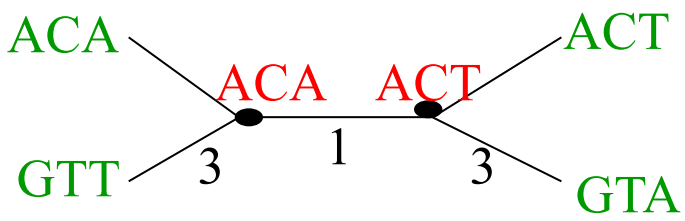




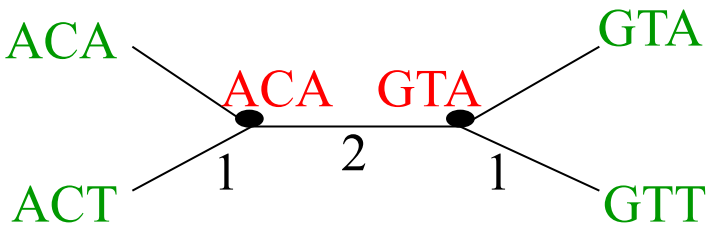
Maximum Parsimony



MP score = 5



MP score = 7

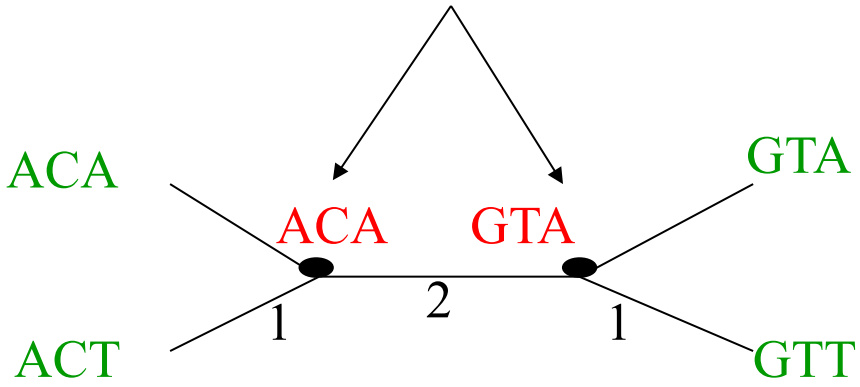


MP score = 4

Optimal MP tree

Maximum Parsimony: computational complexity

Optimal labeling can be computed in linear time O(nk)



MP score = 4

Finding the optimal MP tree is **NP-hard**

Dynamic Programming

- Fixed tree maximum parsimony has dynamic programming solutions – a simple one for unweighted maximum parsimony, and a slightly more complicated one for weighted maximum parsimony.

- What is dynamic programming?

Fibonacci numbers

- $F(1) = F(2) = 1$
- $F(x) = F(x-1)+F(x-2)$ if $x>2$

Calculating F(x) using recursion is exponential, but calculating F(x) using dynamic programming is O(x) time.

DP algorithm

- Dynamic programming algorithms on trees are common – there is a natural ordering on the nodes given by the tree.

- Example: computing the longest leaf-to-leaf path in a tree can be done in linear time, using dynamic programming (bottom-up).

DP algorithm for unweighted MP

- When all substitutions have the same cost, then there is a simple DP method for calculating the MP score on a fixed tree.
- Example: DNA sequences, so 4 letters (A, C, T, G). Let "Set(v)" denote the set of optimal nucleotides at node v (for an MP solution to the subtree rooted at v).

Special case for unweighted MP

- Let "Set(v)" denote the set of optimal nucleotides at node v. Then:
  - If v is a leaf, then Set(v) is {state(v)}.
  - Else we let the two children of v be w and x.
    - If Set(w) and Set(x) are disjoint, then
      Set(v) = Set(w) union Set(x)
    - Else Set(v) = Set(w) intersection Set(x)
- After you assign values to Set(v) for all v, you go to Phase 2 (picking actual states)

Special case for unweighted MP

- Assume we have computed values to Set(v) for all v. Note that Set(v) is not empty.
- Start at the root r of the tree. Pick one nucleotide from Set(r) for the state at r.
- Now visit the children x,y of r, and pick states. If the state of the parent is in Set(x), the use that state; otherwise, pick any element of Set(x).

DP for weighted MP

Single site solution for input tree T.

Root tree T at some internal node. Now, for every node v in T and every possible letter X, compute

Cost(v,X) := optimal cost of subtree of T rooted at v, given that we label v by X.

Base case: easy

General case?

DP algorithm (con't)

Cost(v,X) =

$\min_Y\{Cost(v_1,Y)+cost(X,Y)\}$ +
$\min_Y\{Cost(v_2,Y)+cost(X,Y)\}$

where v_1 and v_2 are the children of v, and Y ranges over the possible "states", and cost(X,Y) is an arbitrary cost function.

DP algorithm (con't)

We compute Cost(v,X) for every node v and every state X, from the "bottom up".

The optimal cost is

$\min_X\{\text{Cost}(\text{root},X)\}$

We can then pick the best states for each node in a top-down pass (just like the algorithm for unweighted MP).

DP algorithm (con't)

Running time? Accuracy?

How to extend to many sites?

Maximum Compatibility

Maximum Compatibility is another approach to phylogeny estimation, often used with morphological traits instead of molecular sequence data. (And used in linguistics as well as in biology.)

Input: matrix M where M_{ij} denotes the state of the species s_i for character j.

Output: tree T on which a maximum number of characters are compatible.

Characters

- A character is a partition of the set of taxa, defined by the states of the character
- Morphological examples: presence/absence of wings, presence/absence of hair, number of legs
- Molecular examples: nucleotide or residue (AA) at a particular site within an alignment

Character Compatibility

- A character c is compatible on a tree T if the states at the internal nodes of T can be set so that for every state, the nodes with that state form a connected subtree of T.
- Equivalently, c is compatible on T if the maximum parsimony score for c on T is k-1, where c has k states at the leaves of T.

Computing the compatibility score on a tree

- Given a matrix M of character states for a set of taxa, and given a tree T for that input, how do we calculate the compatibility score?
- One approach: run maximum parsimony on the input, and determine which characters are compatible.

Character compatibility

- More general problem: given matrix M of character states for a set S of taxa, find the tree with the highest character compatibility score.

- This is NP-hard, even for binary (presence/absence) characters!

Binary character compatibility

- Here the matrix is 0/1. Thus, each character partitions the taxa into two sets: the 0-set and the 1-set.
- Note that a binary character c is compatible on a tree T if and only if the tree T has an edge e whose bipartition is the same as c.

Solving binary character compatibility

- Input: matrix M of 0/1.
- Output: tree T that maximizes character compatibility
- Graph-based Algorithm:
  - Vertex set: one node v_c for each character c
  - Edge set: $(v_c, v_{c'})$ if c and c' are compatible as bipartitions (can co-exist in some tree)

Solving maximum binary character compatibility

- Vertex set: one node v_c for each character c
- Edge set: $(v_c, v_{c'})$ if c and c' are compatible as bipartitions (can co-exist in some tree)
- Note: Every clique in the graph defines a set of compatible characters.
- Hence, finding a maximum sized clique solves the maximum binary character compatibility problem.

Solution to binary character compatibility

- Max Clique is NP-hard, so this is not a fast algorithm. This algorithm shows that Maximum Character Compatibility reduces to Max Clique – not the converse.

- But the converse is also true. So Maximum Character Compatibility is NP-hard.

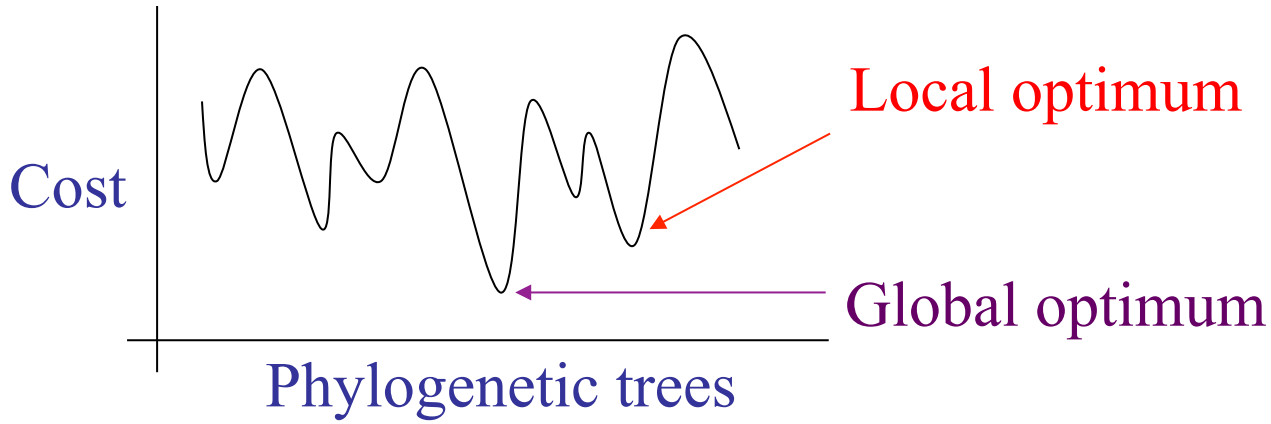
Solving NP-hard problems exactly is … unlikely

- Number of (unrooted) binary trees on *n* leaves is (2n-5)!!
- If each tree on **1000** taxa could be analyzed in **0.001** seconds, we would find the best tree in **2890 millennia**

#leaves	#trees
4	3
5	15
6	105
7	945
8	10395
9	135135
10	2027025
20	2.2×10^{20}
100	4.5×10^{190}
1000	2.7×10^{2900}

Approaches for "solving" MP/MC/ML

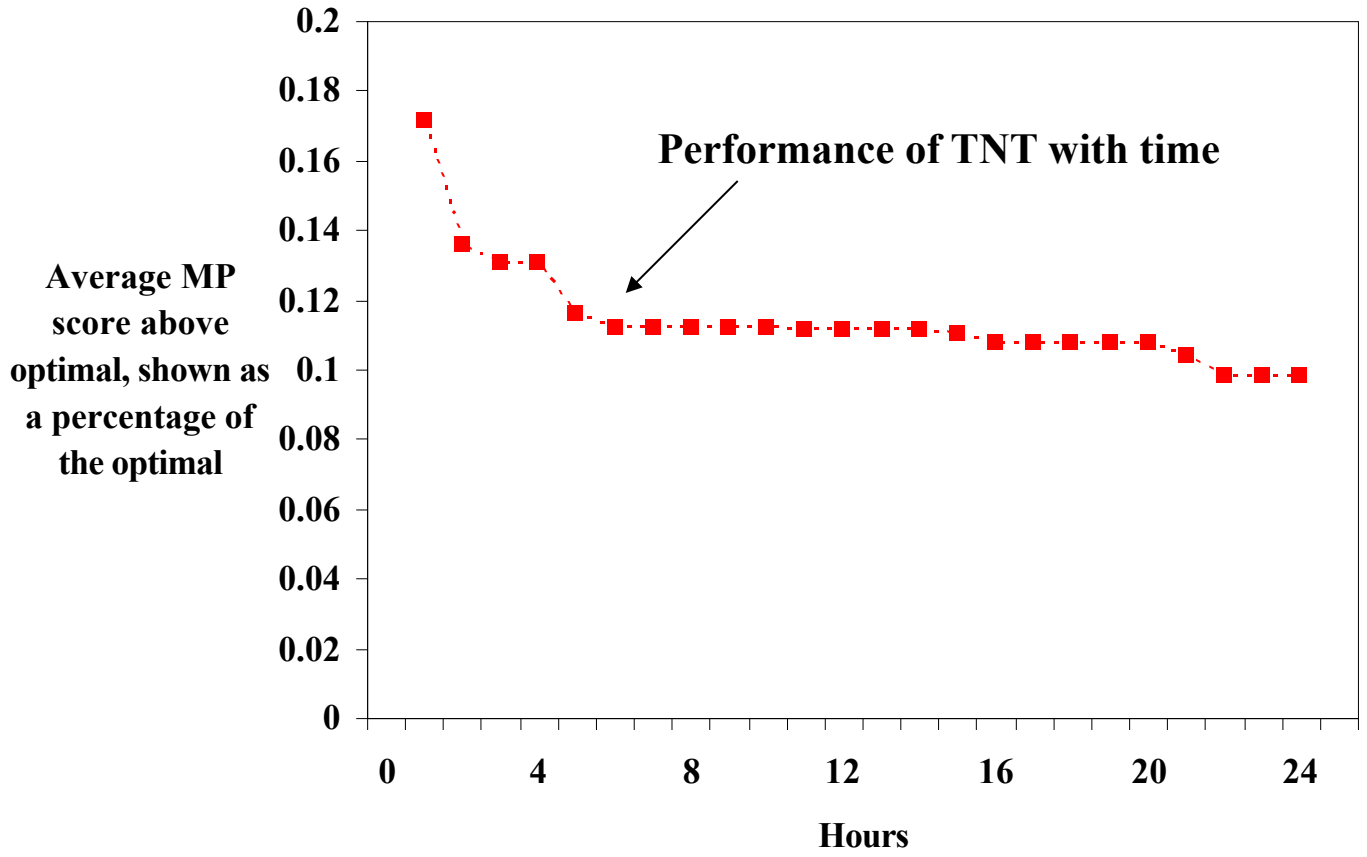
1. Hill-climbing heuristics (which can get stuck in local optima)
2. Randomized algorithms for getting out of local optima
3. Approximation algorithms for MP (based upon Steiner Tree approximation algorithms).



MP = maximum parsimony, MC = maximum compatibility, ML = maximum likelihood

Problems with current techniques for MP

Shown here is the performance of a heuristic maximum parsimony analysis on a real dataset of almost 14,000 sequences. ("Optimal" here means best score to date, using any method for any amount of time.) Acceptable error is below 0.01%.



Observations

- The best heuristics cannot get acceptably good solutions within 24 hours on most of these large datasets.
- Large datasets may need months (or years) of further analysis to reach reasonable solutions.
- Apparent convergence can be misleading.

What happens after the analysis?

- The result of a phylogenetic analysis is often thousands (or tens of thousands) of equally good trees. What to do?
- Biologists use consensus methods, as well as other techniques, to try to infer what is likely to be the characteristics of the "true tree".

Supertree methods

- Input: collection of trees (generally unrooted) on subsets of the taxa
- Output: tree on the entire set of taxa

Basic questions:

- is the set of input trees compatible?
- can we find a tree satisfying a maximum number of input trees?

Triplet-based methods

- Triplet Compatibility: does a tree exist that satisfies all the input triplets? If so, find it. Polynomial time solvable!
- Aho, Sagiv, Szymanski, and Ullman algorithm (works on any input)

Quartet-based methods

- Quartet Compatibility: does there exist a tree compatible with all the input quartet trees? If so, find it. (NP-hard)
- Naïve Quartet Method solves Quartet Compatibility (must have a tree on every quartet)

Tree compatibility

- Unrooted trees: NP-hard
- Rooted trees: Polynomial

But rooted trees are even harder to get exactly correct than unrooted trees!

Real data

- Cannot reliably obtain accurate rooted triplets
- Cannot reliably obtain accurate quartet trees
- All input trees will have some error
- “Supertree” methods need to be able to handle error in the input trees

Quartet-based methods

- Maximum Quartet Compatibility: find a tree satisfying a maximum number of quartet trees (NP-hard)
- PTAS for case where the set contains a tree for every four leaves (Jiang et al.)
- Heuristics (Quartets MaxCut by Snir and Rao, Weight Optimization by Ranwez and Gascuel, Quartet Cleaning by Berry et al., etc.)

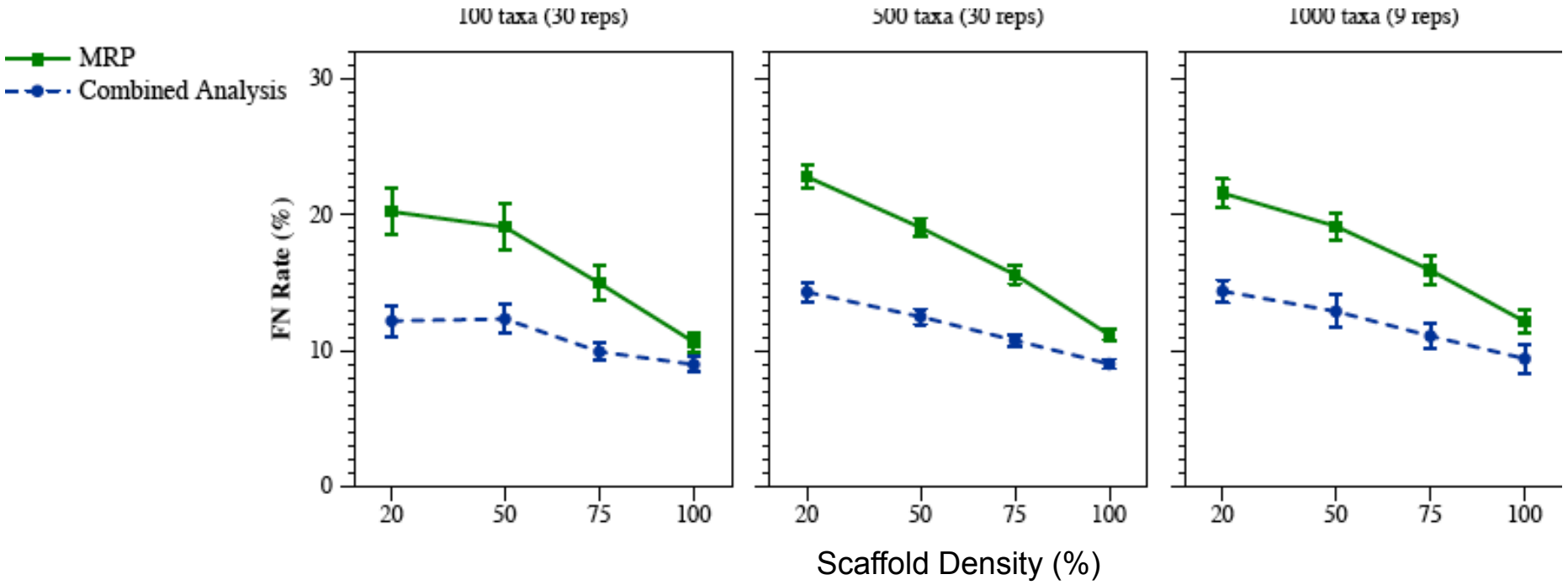
MRP

- Matrix Representation with Parsimony
- Encode each input source tree as a matrix with entries from {0,1,?}, and run maximum parsimony
- Solves “tree compatibility” exactly!

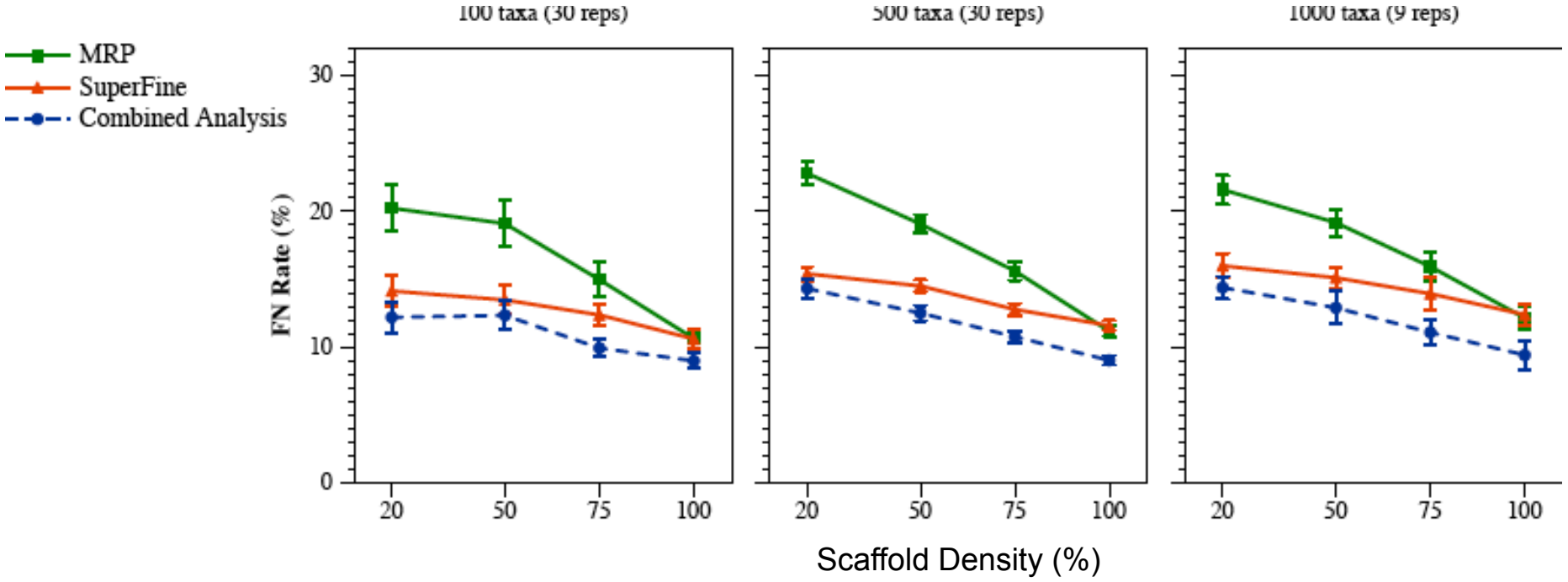
SuperFine

- Swenson et al., Systematic Biology 2012
- Supertree method “booster”
- Two-step procedure: first construct a “constraint” tree (using the strict consensus merger), then refine each polytomy using the preferred supertree method
- Improves MRP and other supertree methods

False Negative Rate

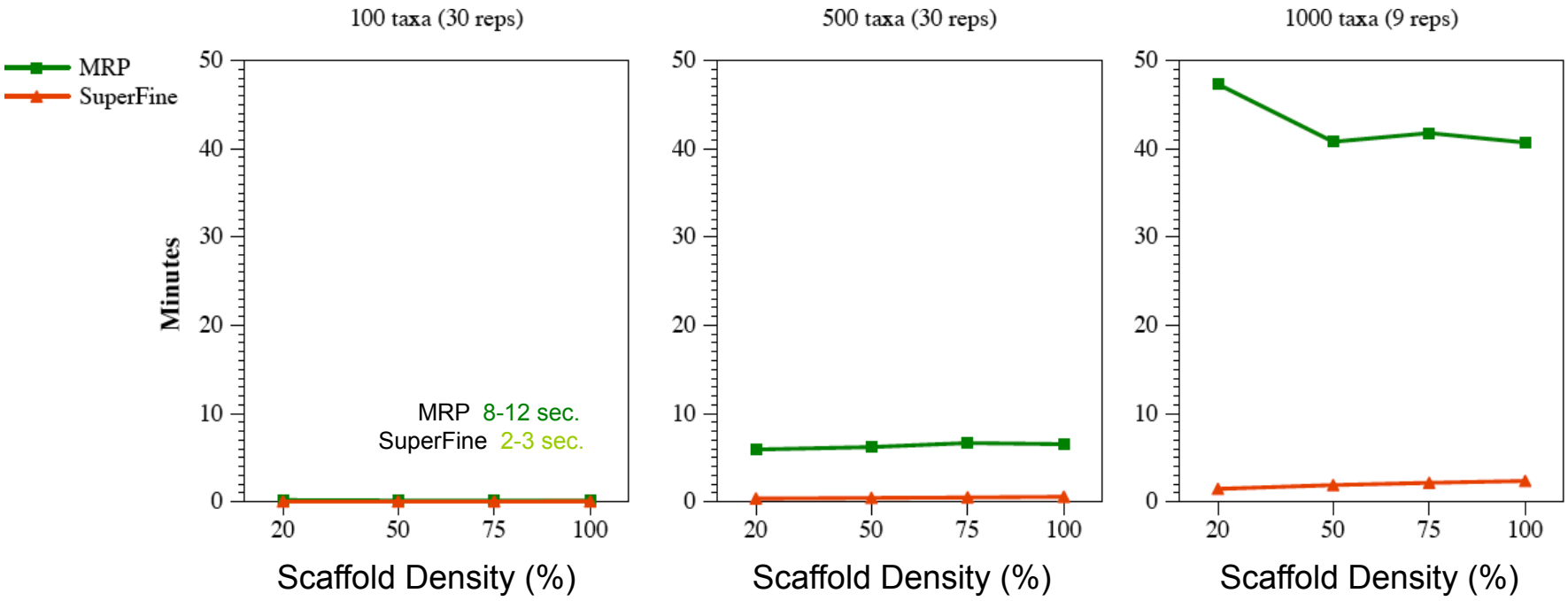


False Negative Rate



Running Time

SuperFine vs. MRP



Homework (due 9/18)

- Find some paper related to supertree or quartet-based tree estimation, read it, and write a 1-2 page discussion of what is in the paper – its claims, whether it's important, and whether you agree with the conclusions (i.e., critique the paper, don't just summarize it).
- This can be a paper that describes a new method, a paper that evaluates such a method on some data, or a paper that uses any such method to analyze some data (e.g., a biological dataset analysis).
- Google Scholar is one way to look for papers; you probably have others.

Some Quartet Tree papers to read

- "Quartets Max Cut…", by Snir and Rao, IEEE/ACM TCBB, vol. 7, no. 4, pp. 704-708
- "Quartet-based phylogenetic inference: improvements and limits", by Ranwez and Gascuel, http://mbe.oxfordjournals.org/content/18/6/1103.full.pdf
- "Short Quartet Puzzling…", by Snir and Warnow. Journal of Computational Biology, Vol. 15, No. 1, January 2008, pp. 91-103.
- "An experimental study of Quartets MaxCut and other supertree methods" by Swenson et al. Journal of Algorithms for Molecular Biology 2011, 6(7),
- "A polynomial time approximation scheme for inferring evolutionary trees from quartet topologies and its applications" by Jiang, Kearney, and Li, SICOMP 2001, http://dl.acm.org/citation.cfm?id=586889
- "Performance study of phylogenetic methods: (unweighted) quartet methods and neighbor-joining," Proceedings SODA 2001 and J. of Algorithms, 48, 1 (2003), 173-193 . (PDF)
- "Quartet Cleaning…" by Berry et al, ESA 1999, LNCS Vol. 1643, pp. 313-324.

SuperFine papers

- Swenson et al. 2012, Systematic Biology (2012) 61(2):214-227
- Nguyen, Mirarab, and Warnow, MRL and SuperFine+MRL: new supertree methods." Journal Algorithms for Molecular Biology 7:3, 2012.

The literature on supertree methods is enormous – look for something recent (last 3 years).