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*

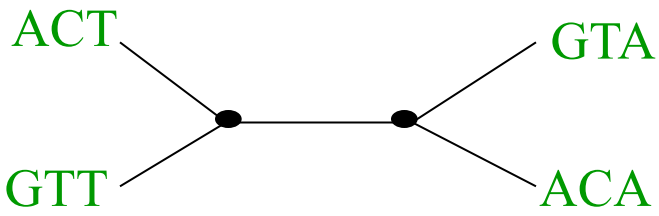
such that $\sum_{(i,j)\in E(T)} H(i,j)$

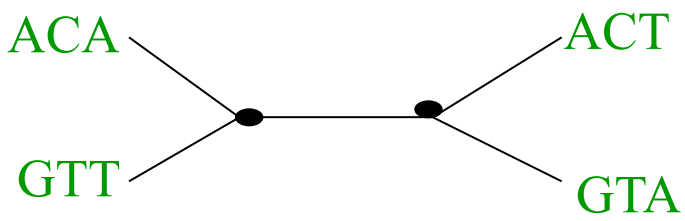
is minimized, where H(i,j) denotes the Hamming distance between sequences at nodes i and j

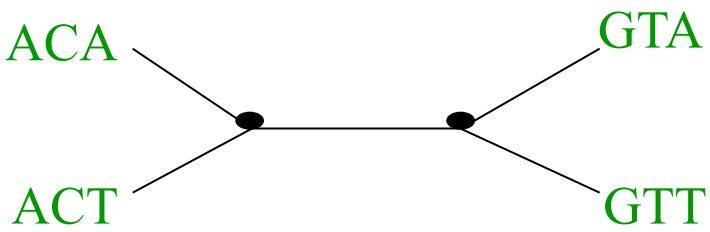
Maximum parsimony (example)

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

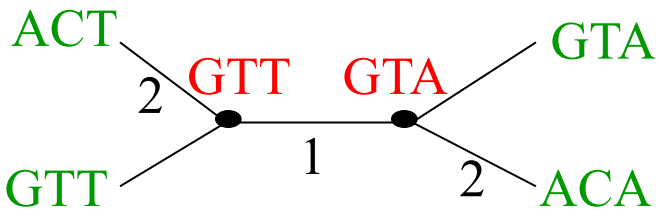
Maximum Parsimony



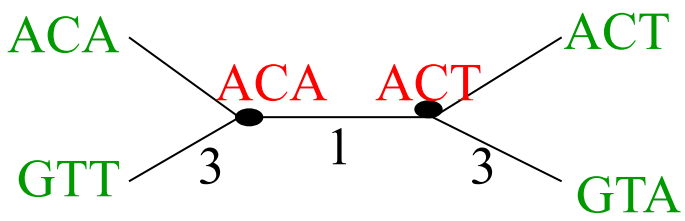




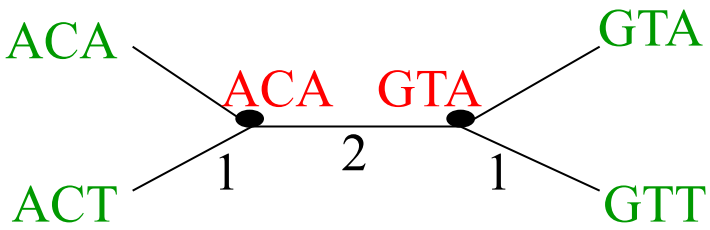
Maximum Parsimony



MP score = 5



MP score = 7

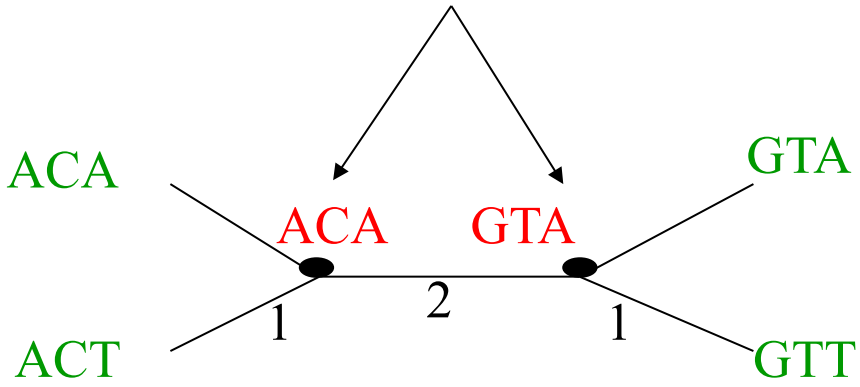


MP score = 4

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**

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

Homoplasy

- Homoplasy is back-mutation or parallel evolution of a character.
- A character labelling the leaves of a tree T is “compatible” on a tree T if you can assign states to the internal nodes so that there is no homoplasy.
- For a binary character, this means the character changes only once on the tree.

Testing Compatibility on a tree

- It is trivial to test if a binary character is compatible on a tree (polynomial time): label all the internal nodes on any 0-0 path by 0, and on any 1-1 path by 1, and see if there are any conflicts.
- Just as easy for multi-state characters, too!

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.

Multi-state character compatibility

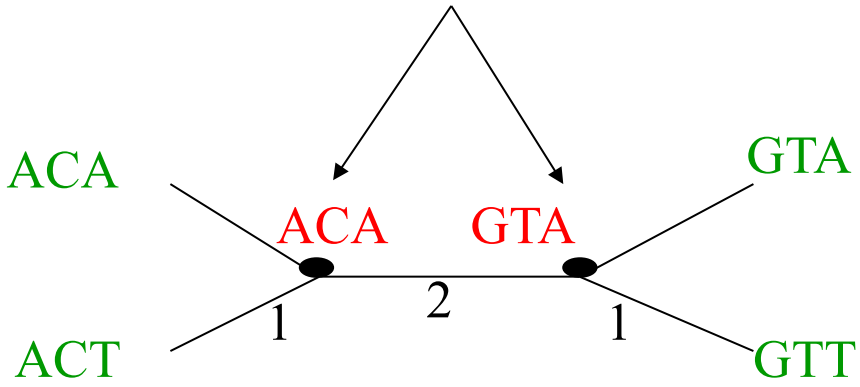
- A character c is compatible on a tree T if the states at the internal nodes of T can be set so that for every state, the nodes with that state form a connected subtree of T.
- Equivalently, c is compatible on T if the maximum parsimony score for c on T is k-1, where c has k states at the leaves of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

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**

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).

Two variants of MP

- Unweighted MP: all substitutions have the same cost
- Weighted MP: there is a substitution cost matrix that allows different substitutions to have different costs. For example: transversions and transitions can have different costs. Even if symmetric, this complicates the calculation – but not by much.

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.
- Let “Set(v)” denote the set of optimal nucleotides at node v (for an MP solution to the subtree rooted at v).

Solving 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)

Solving 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 element 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. However, here we have to remember that different substitutions have different costs.

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.

Setwise character compatibility

- Input: Matrix for a set S of taxa described by a set C of characters.
- Output: Tree T, if it exists, so that every character is compatible on T.

How hard is this problem?

First consider the case where all characters are binary.

Binary character compatibility

- To test binary character compatibility, turn the set of binary characters into a set of bipartitions, and test compatibility for the bipartitions.
- In other words, determining if a set of binary characters is compatible is solvable in polynomial time.

Lemmata

- Lemma 1: A set of binary characters is compatible if and only if all pairs of binary characters are compatible.
- Lemma 2: Two binary characters c,c' are compatible if and only if at least one of the four possible outcomes is missing:
  - (0,0), (0,1), (1,0), and (1,1)

Maximum Compatibility

- Given matrix M defining a set S of taxa and set C of characters, find a maximum size subset C' of C so that a perfect phylogeny exists for (S,C').
- Equivalently, find a tree with the largest MC score (# characters that are compatible)
- How hard is this problem? Consider the case of binary characters first.

Maximum Compatibility for Binary Characters

- 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.

Solving MC for binary characters

- 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.

Multi-state character compatibility

- When the characters are multi-state, the "setwise if and only if pairwise" compatibility lemma no longer holds.
- Testing if a set of multi-state characters is compatible is called the "Perfect Phylogeny Problem". This has many very pretty algorithms for special cases, but is generally NP-complete.

Multi-state character compatibility, aka "Perfect Phylogeny Problem"

- Input: Set of taxa described by a set of multi-state characters.
- Output: YES if the set of characters are compatible (equivalently, if there is a homoplasy-free tree for the input), and otherwise NO.

Not nearly as easy as binary character compatibility, and in fact NP-complete.

Triangulating colored graphs

- A triangulated graph (also known as a "chordal graph") is one that has no simple cycles of size four or larger
- Given a vertex-colored graph G=(V,E), we ask if we can add edges to G so that the graph is triangulated but also properly colored. (Decision problem – YES/NO).

PP and TCG are polytime equivalent

- Solving Perfect Phylogeny is the same as solving Triangulating Colored Graphs (polynomial time equivalent)
- # colors = # characters
- # vertices per color = # states per character
- Polynomial time algorithms for PP for all fixed parameter cases
  - Bounded number of states r
  - Bounded number of characters k
  - Bounded number of taxa

Perfect Phylogenies

- Useful for historical linguistics
- Less useful for biological data, but used to be popular there for analyzing morphological characters
- Some types of biological data seem to be "homoplasy resistant", so perfect phylogenies (or nearly perfect phylogenies) can be relevant even in biology

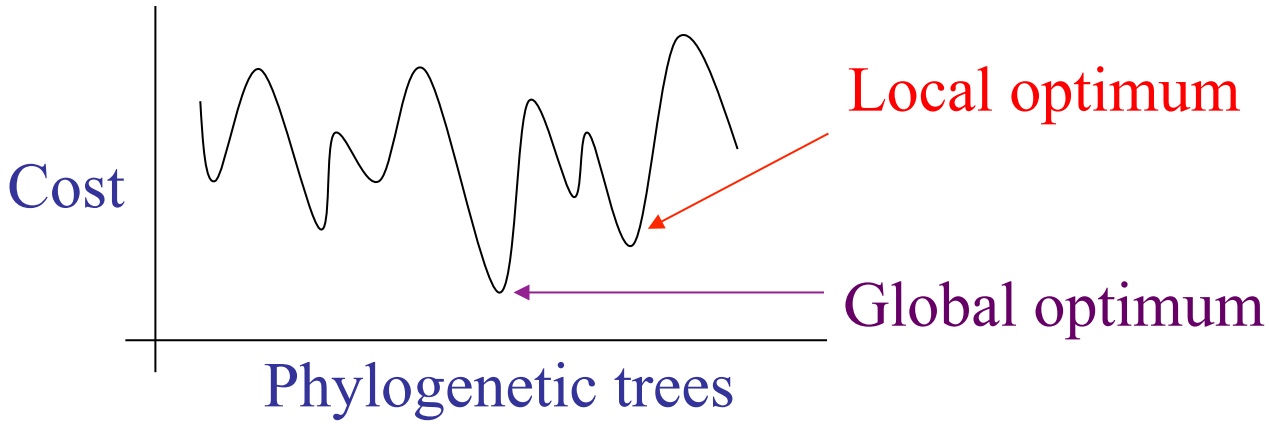
Solving NP-hard problems exactly is … unlikely

- 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

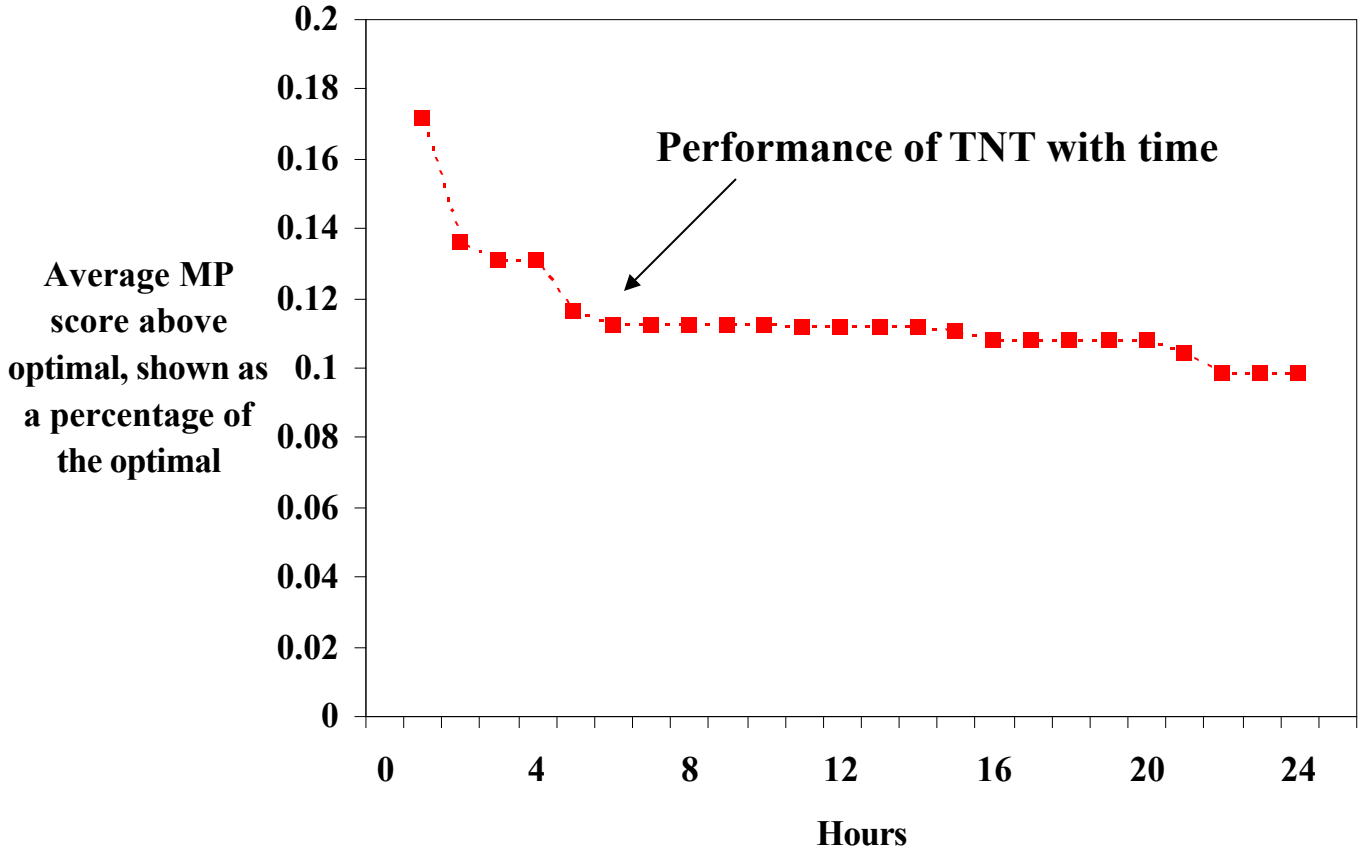
1. Hill-climbing heuristics (which can get stuck in local optima)
2. Randomized algorithms for getting out of local optima
3. Approximation algorithms for MP (based upon Steiner Tree approximation algorithms).



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

Problems with heuristics for MP (OLD EXPERIMENT)

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 about MP/MC/ML

- Large datasets may need months (or years) of analysis to reach reasonably good solutions.
- Even optimal solutions to MP, ML, or MC may not be that close to the true tree. (Probably better to solve ML than the other methods, because of statistical consistency, but the point is nevertheless valid.)
- 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".

Consensus Methods

- Strict Consensus – containing all the splits that all trees share (unique)
- Majority Consensus – containing all the splits that >50% of the trees share (unique)
- Greedy Consensus – order the splits by their frequency, then put them into a tree in that order… adding each split if possible (not unique)

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?

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)

But quartet trees will have error…

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.)

Homework (due Feb 17)

- Find 1 paper related to 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.
- "Weighted Quartets Phylogenetics", by Avani, Cohen, and Snir. Systematic Biology, advance access, November 2014.