Chapter 6

598 AGB

Today’s material

- Maximum Parsimony
- Fixed tree versions (solvable in polynomial time using dynamic programming)
- Optimal tree search (NP-hard for both problems)
- Why MP is not statistically consistent under the CFN model
- Maximum compatibility (time permitting)

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*

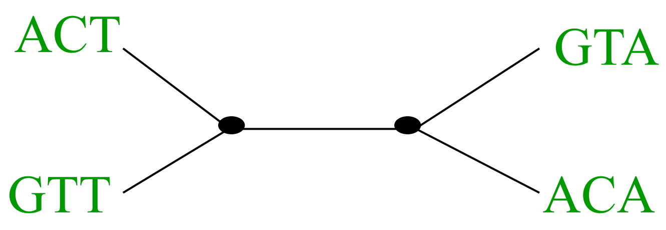
such that $$\sum_{(i,j)\in E(T)} H(i,j)$$

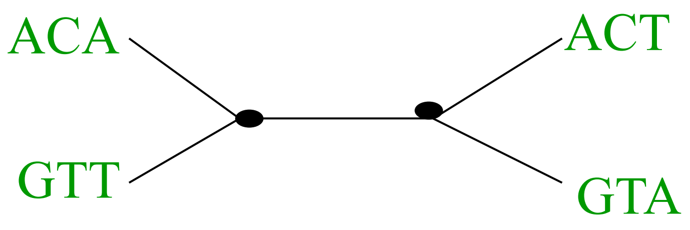
is minimized, where H(i,j) denotes the Hamming distance between sequences at nodes i and j

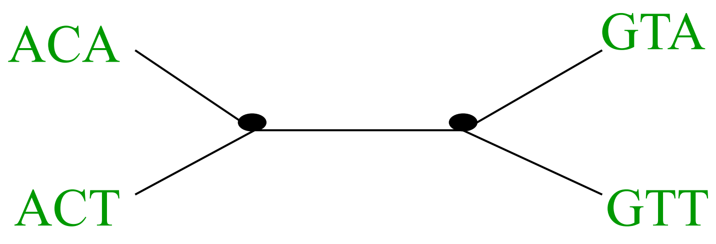
Maximum parsimony (example)

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

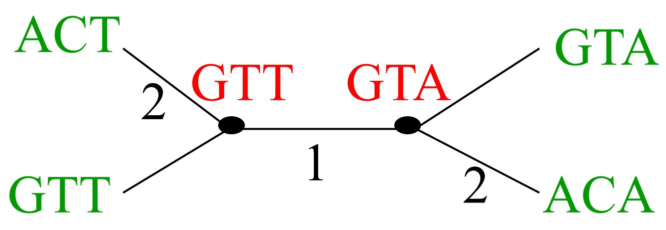
Maximum Parsimony



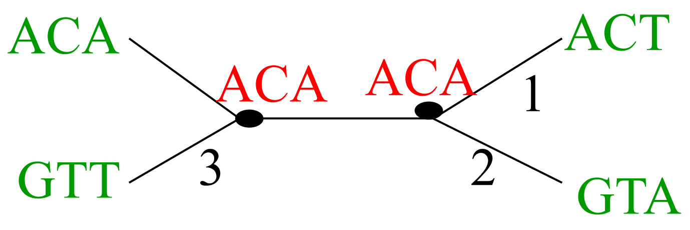




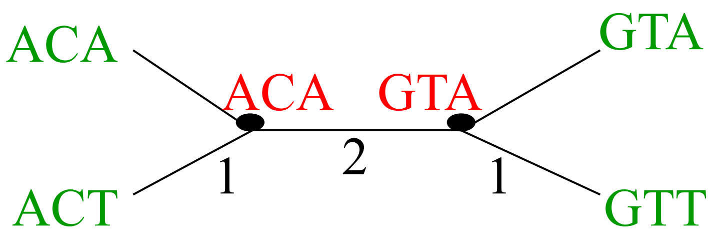
Maximum Parsimony



MP score = 5



MP score = 6

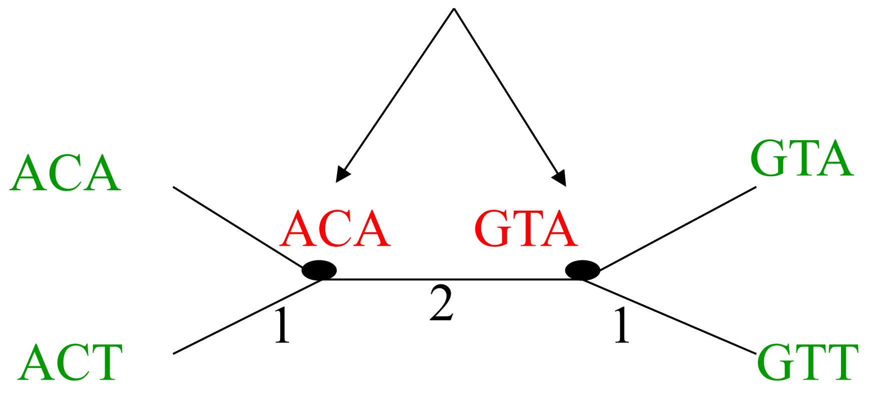


MP score = 4

Optimal MP tree

Maximum Parsimony: computational complexity

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



MP score = 4

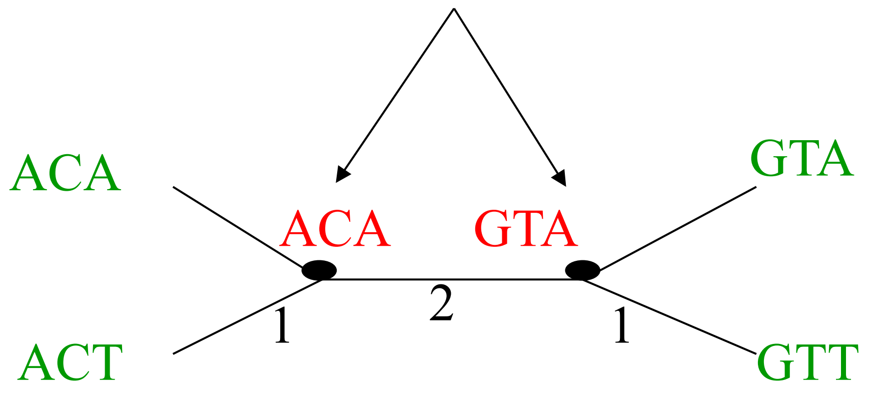
Finding the optimal MP tree is **NP-hard**

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

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)

$\text{Cost}(v,X) =$

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

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

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

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?

Solving NP-hard problems exactly is … unlikely

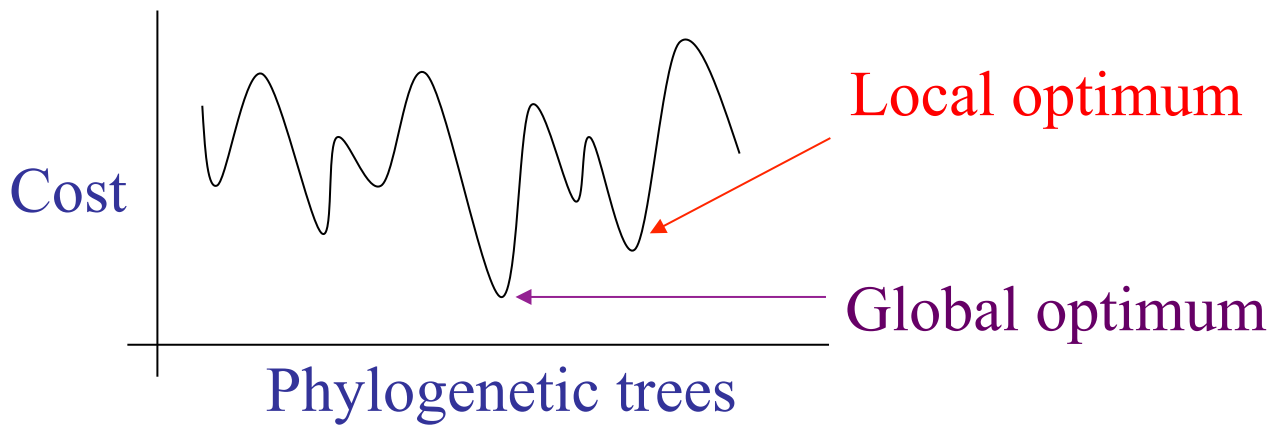
- Number of (unrooted) binary trees on *n* leaves is (2n-5)!!
- If each tree on **1000** taxa could be analyzed in **0.001** seconds, we would find the best tree in

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/ML

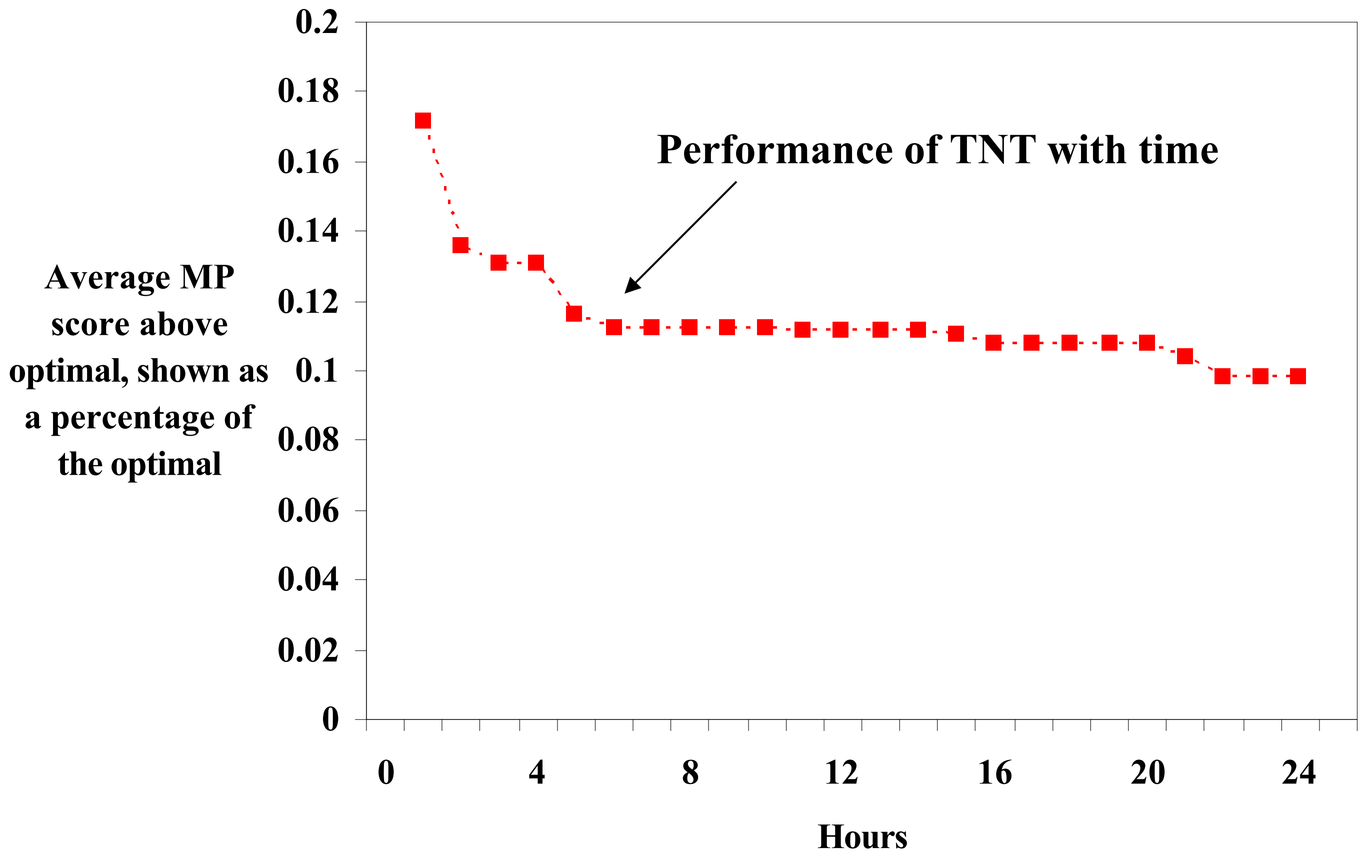
1. Hill-climbing heuristics (which can get stuck in local optima)
2. Randomized algorithms for getting out of local optima
3. Approximation algorithms for MP (based upon Steiner Tree approximation algorithms).



MP = maximum parsimony, 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%.



Summary (so far)

- Maximum Parsimony is an NP-hard optimization problem, but can be solved exactly (using dynamic programming) in polynomial time on a fixed tree.
- Heuristics for MP are reasonably fast, but apparent convergence can be misleading. And some datasets can take a long time.

Is Maximum Parsimony statistically consistent under CFN?

- Recall the CFN model of binary sequence evolution: iid site evolution, and each site changes with probability p(e) on edge e, with $0 < p(e) < 0.5$.
- Is MP statistically consistent under this model?

Statistical consistency under CFN

- We will say that a method M is statistically consistent under the CFN model if:
  - For all CFN model trees (T,Θ) (where Θ denotes the set of substitution probabilities on each of the branches of the tree T), as the number L of sites goes to infinity, the probability that M(S)=T converges to 1, where S is a set of sequences of length L.

Is MP statistically consistent?

- We will start with 4-leaf CFN trees, so the input to MP is a set of four sequences, A, B, C, D.
- Note that there are only three possible unrooted trees that MP can return:
  - ((A,B),(C,D))
  - ((A,C),(B,D))
  - ((A,D),(B,C))

Analyzing what MP does on four leaves

- MP has to pick the tree that has the least number of changes among the three possible trees.
- Consider a single site (i.e., all the sequences have length one).
- Suppose the site is A=B=C=D=0. Can we distinguish between the three trees?

Analyzing what MP does on four leaves

- Suppose the site is A=B=C=D=0.
- Suppose the site is A=B=C=D=1
- Suppose the site is A=B=C=0, D=1
- Suppose the site is A=B=C=1, D=0
- Suppose the site is A=B=D=0, C=1
- Suppose the site is A=C=D=0, B=1
- Suppose the site is B=C=D=0, A=1

Uninformative Site Patterns

Uninformative site patterns are ones that fit every tree equally well. Note that any site that is constant (same value for A,B,C,D) or splits 3/1 is parsimony uninformative.

On the other hand, all sites that split 2/2 are parsimony informative!

Parsimony Informative Sites

- [A=B=0, C=D=1] or [A=B=1, C=D=0]
  - These sites support ((A,B),(C,D))
- [A=C=1, B=D=0] or [A=C=0, B=D=1]
  - These sites support ((A,C),(B,D))
- [A=D=0,B=C=1] or [A=D=1, B=C=0]
  - These sites support ((A,D),(B,C))

Calculating which tree MP picks

- When the input has only four sequences, calculating what MP does is easy!

1. Remove the parsimony uninformative sites
2. Let I be the number of sites that support ((A,B),(C,D))
3. Let J be the number of sites that support ((A,C),(B,D))
4. Let K be the number of sites that support ((A,D),(B,C))
5. Whichever tree is supported by the largest number of sites, return that tree. (For example, if I >max{J,K}, then return ((A,B),(C,D).)
6. If there is a tie, return all trees supported by the largest number of sites.

MP on 4 leaves

- Consider a four-leaf tree CFN model tree ((A,B),(C,D)) with a very high probability of change (close to ½) on the internal edge (separating AB from CD) and very small probabilities of change (close to 0) on the four external edges.
- What parsimony informative sites have the highest probability? What tree will MP return with probability increasing to 1, as the number of sites increases?

MP on 4 leaves

- Consider a four-leaf tree CFN model tree ((A,B),(C,D)) with a very high probability of change (close to ½) on the two edges incident with A and B, and very small probabilities of change (close to 0) on all other edges.
- What parsimony informative sites have the highest probability? What tree will MP return with probability increasing to 1, as the number of sites increases?

MP on 4 leaves

- Consider a four-leaf tree CFN model tree ((A,B),(C,D)) with a very high probability of change (close to ½) on the two edges incident with A and C, and very small probabilities of change (close to 0) on all other edges.
- What parsimony informative sites have the highest probability? What tree will MP return with probability increasing to 1, as the number of sites increases?

Summary (updated)

- Maximum Parsimony (MP) is statistically consistent on some CFN model trees.
- However, there are some other CFN model trees in which MP is not statistically consistent. Worse, MP is positively misleading on some CFN model trees. This phenomenon is called “long branch attraction”, and the trees for which MP is not consistent are referred to as “Felsenstein Zone trees” (after the paper by Felsenstein).
- The problem is not limited to 4-leaf trees…

Overall summary

- Maximum Parsimony is not statistically consistent under standard sequence evolution models, but it can be consistent on some model trees.
- Maximum parsimony is NP-hard and computationally intensive in practice. The best heuristics for MP used to be faster than the best heuristics for ML (maximum likelihood), but this may no longer be true.
- Even so, MP remains one of the popular techniques for phylogeny estimation.

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.

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.

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.