Bioinformatics Algorithms

(Fundamental Algorithms, module 2)

Zsuzsanna Lipták

Masters in Medical Bioinformatics
academic year 2017/18, spring term

$\boxed{\text{Phylogenetics II}}$

Character data

Character data

Now the input data consists of states of characters for the given objects, e.g.

- morphological data, e.g. number of toes, reproductive method, type of hip bone, . . . or
- molecular data, e.g. what is the nucleotoide in a certain position.

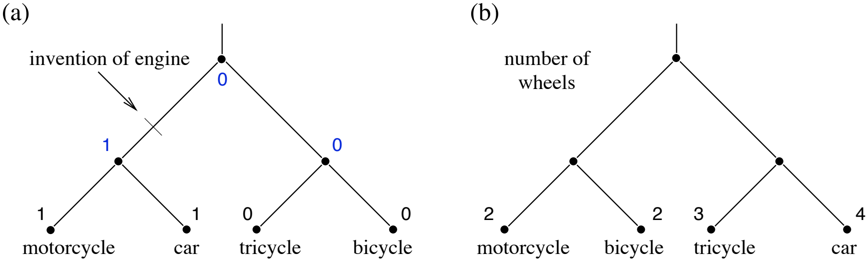
Character data

Example

	C_1 : # wheels	C_2 : existence of engine
bicycle	2	0
motorcycle	2	1
car	4	1
tricycle	3	0

- objects (species): Bicycle, motorcycle, tricycle, car
- characters: number of wheels; existence of an engine
- character states: $2, 3, 4$ for C_1;
  $0, 1$ for C_2 ($1 =$ YES, $0 =$ NO)
- This matrix M is called a character-state-matrix, of dimension $(n \times m)$, where for $1 \leq i \leq n, 1 \leq j \leq m$: $M_{ij} =$ state of character j for object i. (Here: $n = 4, m = 2$.)

Character data



Two different phylogenetic trees for the same set of objects.

Character data

We want to avoid

- parallel evolution ($=$ convergence)
- reversals

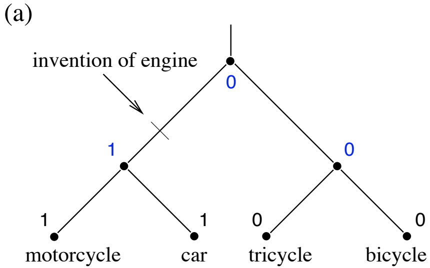
Together these two conditions are also called homoplasies.

Mathematical formulation: compatibility.

Compatibility

Definition

A character is compatible with a tree if all inner nodes of the tree can be labeled such that each character state induces one connected subtree.



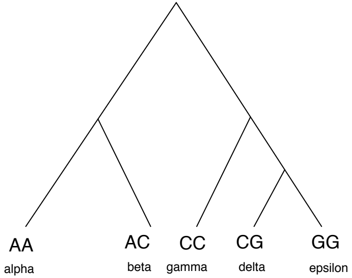
This tree is compatible with C_2, one possibility of labeling the inner nodes is shown.

Perfect Phylogeny

Definition

A tree T is called a perfect phylogeny (PP) for $\mathcal{C}$ if all characters $C \in \mathcal{C}$ are compatible with T.

Example



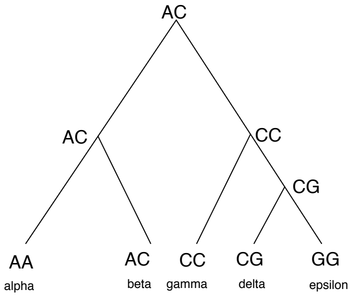
Why? We have to find a labeling of the inner nodes s.t. for both characters C_1 and C_2, each state induces a subtree.

Perfect Phylogeny

Definition

A tree T is called a perfect phylogeny (PP) if all characters are compatible with T.

Example



Note: Our first tree for the vehicles was also a PP. (Proof?)

Perfect Phylogeny

- Ideally, we would like to find a PP for our input data.

Perfect Phylogeny

- Ideally, we would like to find a PP for our input data.
- For binary characters, there is an efficient algorithm for deciding whether a PP exists. (Gusfield's algorithm, see lecture)

Perfect Phylogeny

- Ideally, we would like to find a PP for our input data.
- For binary characters, there is an efficient algorithm for deciding whether a PP exists. (Gusfield's algorithm, see lecture)
- Deciding in general whether a PP exists is NP-hard. (More precisely: For characters with number of states ≥ 4, the PP problem is NP-hard.)

Perfect Phylogeny

- Ideally, we would like to find a PP for our input data.
- For binary characters, there is an efficient algorithm for deciding whether a PP exists. (Gusfield's algorithm, see lecture)
- Deciding in general whether a PP exists is NP-hard. (More precisely: For characters with number of states ≥ 4, the PP problem is NP-hard.)
- Doesn't really matter, since most of the time, no PP exists anyway. Why: due to homoplasies; because our input data has errors; our evolutionary model may not be adequate; and, and, and ...

Perfect Phylogeny

- Ideally, we would like to find a PP for our input data.
- For binary characters, there is an efficient algorithm for deciding whether a PP exists. (Gusfield's algorithm, see lecture)
- Deciding in general whether a PP exists is NP-hard. (More precisely: For characters with number of states ≥ 4, the PP problem is NP-hard.)
- Doesn't really matter, since most of the time, no PP exists anyway. Why: due to homoplasies; because our input data has errors; our evolutionary model may not be adequate; and, and, and ...
- Therefore we usually want to find a best possible tree.

Compatibility and Parsimony

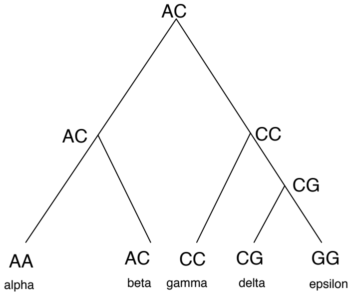
What is a best possible tree?

Two possibilities:

- Compatibility: what is the largest subset of the characters such that a PP exists? - This means ignoring part of the input data.
- Parsimony: If we want to keep our input data, then what is the smallest number of changes that have to be made along the edges?

Parsimony

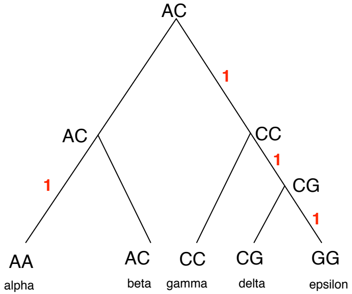
Parsimony: What is a best possible tree?



Why is this tree "perfect"?

Parsimony

What is a best possible tree?



Why is this tree "perfect"?

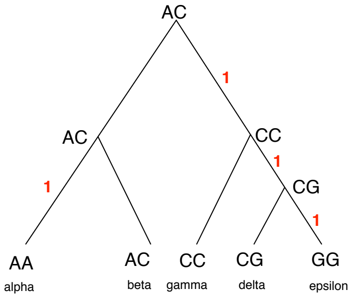
Because it has few changes of states!

In red, we marked the edges where there are state changes (an evolutionary event happened), and how many (in this case, always 1).

Parsimony

Definition

The parsimony cost of a phylogenetic tree with labeled inner nodes is the number of state changes along the edges (i.e. the sum of the edge costs, where the cost of an edge = number of characters whose state differs between child and parent).

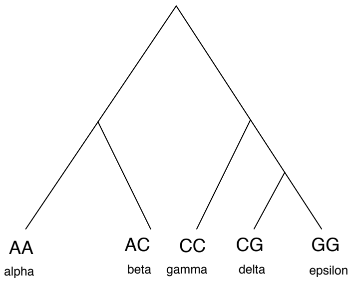


The parsimony cost of this labeled tree is 4.

Parsimony

Definition

The parsimony cost of a phylogenetic tree (without labels on the inner nodes) is the minimum of the parsimony cost over all possible labelings of the inner nodes.



The parsimony cost of this tree is 4, because the best labeling has cost 4.

Parsimony

Phylogenetic Reconstruction with Character Data

Given a character-state matrix M, our goal is to find a phylogenetic tree which minimizes the parsimony cost.

We split the problem into two sub-problems:

1. Small Parsimony: Given a phylogenetic tree, find its parsimony cost, i.e. find a most parsimonious labeling of the inner nodes. This problem can be solved efficiently.
2. Large Parsimony or Maximum Parsimony: Find a tree with minimum parsimony cost. This problem is NP-hard.

Small Parsimony

Small Parsimony Problem

Given: a phylogenetic tree T with character-states at the nodes.
Find: a labeling of the inner nodes with states with minimum parsimony cost.

Algorithm

This problem can be solved using Fitch' algorithm, which runs in time $O(nmr)$, where $n =$ number of species, $m =$ number of characters, and $r =$ maximum number of states over all characters.

Maximum Parsimony

Maximum Parsimony Problem

The maximum parsimony problem is, given a character-state matrix, find a phylogenetic tree with lowest parsimony cost (= a "most parsimonious tree").

- When a PP exists, then it is also the most parsimonious tree.
- In general, this problem is NP-hard.

Algorithms for Maximum Parsimony

- Since problem NP-hard, we cannot hope to find an algorithm that solves it efficiently.
- We have seen the following algorithms for this problem:
  1. Greedy Sequential Addition Algorithm - heuristic algorithm: guaranteed polynomial running time but no guarantee on the quality of the solution (may or may not be correct, i.e. may or may not output the best tree)
  2. Branch-and-Bound for Parsimony - running time heuristic: guarantee on exact solution, but no guarantee on the running time (may or may not be fast)
  3. Nearest Neighbor Interchange - a local optimization algorithm (also a heuristic algorithm, but guarantees to output a local optimum)

Summary for character data

- When the input is a character-state matrix, then we would like to find a tree which is compatible with each character.

Summary for character data

- When the input is a character-state matrix, then we would like to find a tree which is compatible with each character.
- Such a tree is called a perfect phylogeny (PP).

Summary for character data

- When the input is a character-state matrix, then we would like to find a tree which is compatible with each character.
- Such a tree is called a perfect phylogeny (PP).
- PPP can be solved efficiently for binary characters, is NP-hard for number of states ≥ 4.
- Usually, no PP exists, therefore in general ...

Summary for character data

- When the input is a character-state matrix, then we would like to find a tree which is compatible with each character.
- Such a tree is called a perfect phylogeny (PP).
- PPP can be solved efficiently for binary characters, is NP-hard for number of states ≥ 4.
- Usually, no PP exists, therefore in general ...
- We are looking for a most parsimonious tree (a tree with lowest parsimony cost).

Summary for character data

- When the input is a character-state matrix, then we would like to find a tree which is compatible with each character.
- Such a tree is called a perfect phylogeny (PP).
- PPP can be solved efficiently for binary characters, is NP-hard for number of states ≥ 4.
- Usually, no PP exists, therefore in general ...
- We are looking for a most parsimonious tree (a tree with lowest parsimony cost).
- The parsimony cost is defined as the minimum number of the state changes on the edges over all possible labelings of the inner nodes.

Summary for character data

- When the input is a character-state matrix, then we would like to find a tree which is compatible with each character.
- Such a tree is called a perfect phylogeny (PP).
- PPP can be solved efficiently for binary characters, is NP-hard for number of states ≥ 4.
- Usually, no PP exists, therefore in general ...
- We are looking for a most parsimonious tree (a tree with lowest parsimony cost).
- The parsimony cost is defined as the minimum number of the state changes on the edges over all possible labelings of the inner nodes.
- Recall: There are super-exponentially many trees on n taxa (both rooted and unrooted), so we cannot try them all.

Summary for character data (cont'ed)

- recall: We are looking for a a most parsimonious tree (a tree with lowest parsimony cost).

Summary for character data (cont'ed)

- recall: We are looking for a a most parsimonious tree (a tree with lowest parsimony cost).
- Problem is split into Small Parsimony and Maximum Parsimony.

Summary for character data (cont'ed)

- recall: We are looking for a a most parsimonious tree (a tree with lowest parsimony cost).
- Problem is split into Small Parsimony and Maximum Parsimony.
- Small Parsimony can be solved efficienly, e.g. by Fitch' algorithm.

Summary for character data (cont'ed)

- recall: We are looking for a a most parsimonious tree (a tree with lowest parsimony cost).
- Problem is split into Small Parsimony and Maximum Parsimony.
- Small Parsimony can be solved efficiently, e.g. by Fitch' algorithm.
- Maximum Parsimony is NP-hard, so probably no efficient algorithms exist.

Summary for character data (cont'ed)

- recall: We are looking for a a most parsimonious tree (a tree with lowest parsimony cost).
- Problem is split into Small Parsimony and Maximum Parsimony.
- Small Parsimony can be solved efficienly, e.g. by Fitch' algorithm.
- Maximum Parsimony is NP-hard, so probably no efficient algorithms exist.
- We saw three algorithms for Maximum Parsimony: one heuristic (Greedy Seq. Addition Algo.) and one exact algorithm which is a running time heuristic (Branch-and-Bound for Parsimony), and a local optimization algorithm (Nearest Neighbor Interchange).