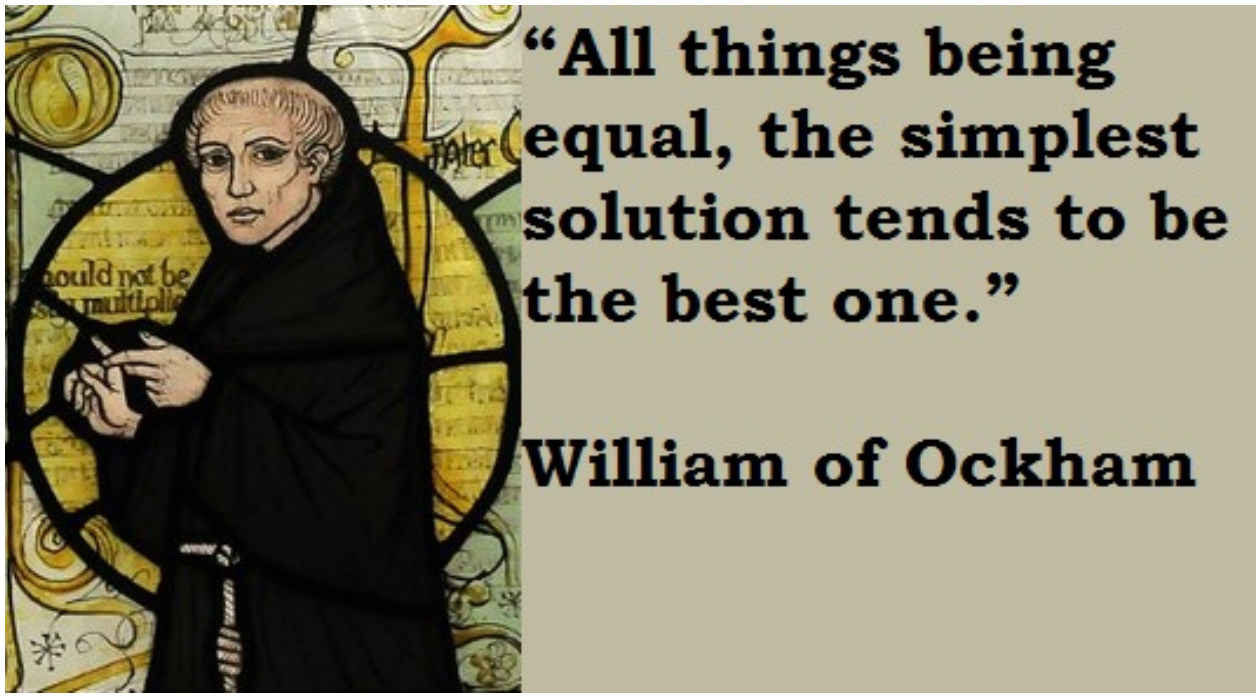
# Algorithmen der Bioinformatik I
# WS 2017/2018

Burkhard Morgenstern
Peter Meinicke

Dept. Bioinformatics
Institute of Microbiology and Genetics (IMG)
University of Göttingen

January 15, 2018

# Maximum Parsimony




Ockham’s razor: among competing hypotheses, the one with the fewest assumptions should be preferred.

# Maximum Parsimony

*Idea for phylogeny reconstruction:*

For sequences $S_1, \ldots, S_n$, find tree that minimizes number of evolutionary events that are necessary to explain $S_1, \ldots, S_n$.

For number of evolutionary events, branch lengths *not* relevant. Therefore: consider only *topology* of tree, *i.e.* branching scheme without branch lengths.

Topology can be represented by parenthesis, *e.g.*:

$$(((S_1, S_3)S_5)(S_2, S_4))$$

# Maximum Parsimony

## Example

| | | | | | | |
|---|---|---|---|---|---|---|
| $S_1$ | a | g | c | a | t | c |
| $S_2$ | a | g | c | a | t | c |
| $S_3$ | a | g | t | a | t | c |
| $S_4$ | a | g | t | a | t | c |

$n = 4$, three different nucleotides at position 3

- Question: How many mutations have to be assumed to explain evolution of sequences from common ancester along tree $T$?
- Answer: depends on tree $T$

# Maximum Parsimony

*Generally:*

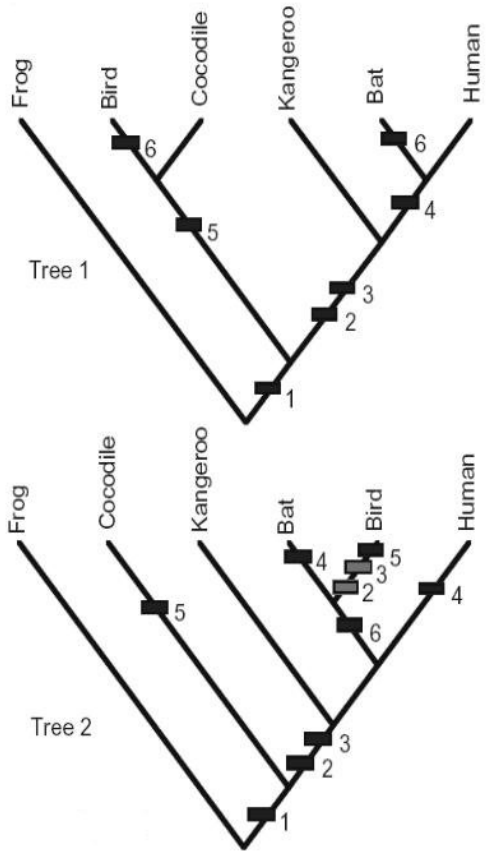
- Given *n* species and *k characters* present in all species.
- For characters, different *states* possible; input data represented as *matrix*
- Question: if species evolved from a common ancester, how many events (mutations) must be (at least) assumed, to explain different states of characters?

Number of events depends on tree (topology)

# Maximum Parsimony

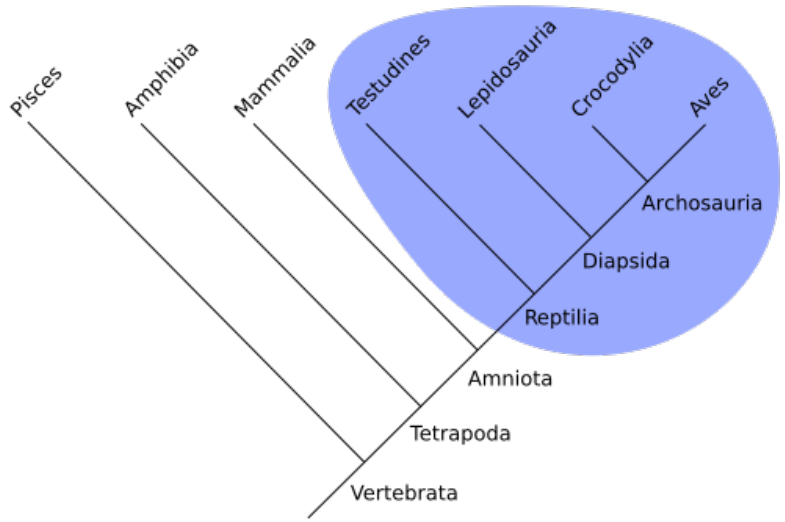
Exampel: evolution of vertebrates, 6 characters, each has two possible states ($+$ and $-$)

| | | CHARACTERS | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | 1 | 2 | 3 | 4 | 5 | 6 | |
| | | amnion | hair | lactation | placenta | antorbital fenestra | wings | |
| TAXA | Frog | - | - | - | - | - | - | |
| | Bird | + | - | - | - | + | + | |
| | Crocodile | + | - | - | - | + | - | |
| | Kangeroo | + | + | + | - | - | - | |
| | Bat | + | + | + | + | - | + | |
| | Human | + | + | + | + | - | - | TREE LENGTH |
| FIT | Tree 1 | 1 | 1 | 1 | 1 | 1 | 2 | 7 |
| | Tree 2 | 1 | 2 | 2 | 2 | 2 | 1 | 10 |



# Maximum Parsimony

Supporters of parsimony: *Cladists* (W. Hennig)
Goal: Systematic of organisms based on evolution. Find *monophyletic groups*




Source: Wikipedia
Example: reptils *not* monophyletic.

# Maximum Parsimony

For molecular sequence data:

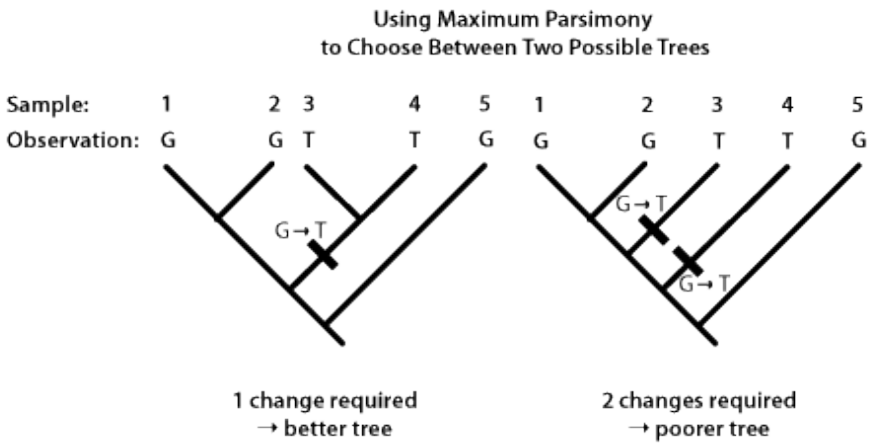
- Characters are positions in sequences
- States are bases or amino acids in sequences

In this case: data matrix = multiple alignment (ignore columns with gaps)

# Maximum Parsimony

For sequence data: multiple alignment as ‘data matrix’. How many substitutions in alignment column?

| | | | |
|---|---|---|---|
| 1 | . . . | *G* | . . . |
| 2 | . . . | *G* | . . . |
| 3 | . . . | *T* | . . . |
| 4 | . . . | *T* | . . . |
| 5 | . . . | *G* | . . . |




http://www.allanwilsoncentre.ac.nz/

Two questions to be solved for maximum parsimnoy:

1. How to calculate number of necessary substitutions for *given* tree (topology)?
   'Small parsimony problem'
2. How to find *best* topology, *i.e.* topology with minimum number of mutations?
   'Big parsimony problem'

# The ‘small parsimony problem’

- Consider number of mutations for *single* columns in alignment, take sum of necessary mutations over all columns.
- For given tree (topology) and given column in alignment: use *dynamic programming* to calculate for each node $k$ set $R_k$ of nucleotides that are possible at $k$ to minimize number of mutations in *subtree* below $k$.

Idea of *DP*: solve smaller sub-problems of a large problem *recursively*.

# The ‘small parsimony problem’

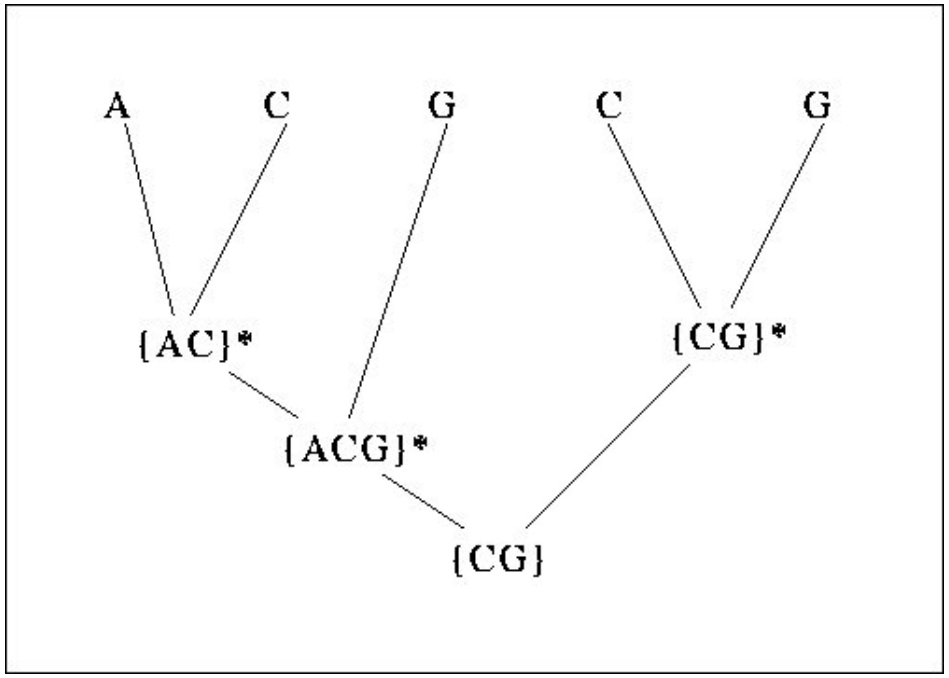



Figure: The sets $R_k$ of possible nucleotides at inner nodes of a tree for a given column in the alignment. Nucleotides at leaves (in alignment column): `A, C, G, C, G`

# The ‘small parsimony problem’

Algorithm:

- Traverse nodes of tree from leaves to root
- For each node $k$, calculate set $R_k$ of nucleotides that can be placed at $k$ to minimize number of mutations in sub-tree below $k$

  Recursion formula for node $k$ with daughter nodes $i$ and $j$:

$$R_k = \begin{cases} R_i \cap R_j & \text{if } R_i \cap R_j \neq \emptyset \quad (1) \\ R_i \cup R_j & \text{if } R_i \cap R_j = \emptyset \quad (2) \end{cases}$$

- Keep counter $C$ for number of necessary mutations
- If (2) applies, increase counter $C$ by 1.

# The ‘small parsimony problem’

Initial values:

- For leave node $l$ with nucleotide $x$, $R_l = \{x\}$
- $C = 0$

To find optimal assignments of nucleotides to nodes:
Go back from root $r$ to leaves (‘trace back’)

- Assign arbitrary nucleotide from $R_r$ to root $r$
- If nucleotide $x$ has been chosen for node $k$, select nucleotide for daughter node as follows:
  - If $x \in R_i$, chose $x$ for node $i$
  - Else, chose arbitrary nucleotide from $R_i$

# The ‘small parsimony problem’

Generalisation: *weighted parsimony*

- ‘cost’ $S(a, b)$ for mutation $a \to b$ or $b \to a$.
- Wanted: tree that minimizes sum of costs of necessary mutations.
- For given alignment column and topology, apply dynamic programming.

# The ‘small parsimony problem’

Calculate for each node $k$ and nucleotide $a$ minimal costs

$$S_k(a)$$

for subtree $T_k$ below $k$, if $a$ is at $k$

Initialise for leave $k$:

$$S_k(a) = \begin{cases} 0 \text{ if base } a \text{ at leave } k \\ \infty \text{ else} \end{cases}$$

# The ‘small parsimony problem’

Recursion for inner node $k$ with children $i$ and $j$:

$$S_k(a) = \min_b[S_i(b) + S(a,b)] + \min_b[S_j(b) + S(a,b)]$$

As above: calculate $S_k(a)$ from leaves to root

If root $w$ reached:

Total cost for tree = $\min_a S_w(a)$

# The ‘big parsimony problem’

# The ‘big parsimony problem’

*Big parsimony problem:*

Find optimal tree topology, i.e. topology that requires minimal number of substitutions.

Problem: number of possible tree topologies huge, no efficient algorithm known to find best topology.

# The ‘big parsimony problem’

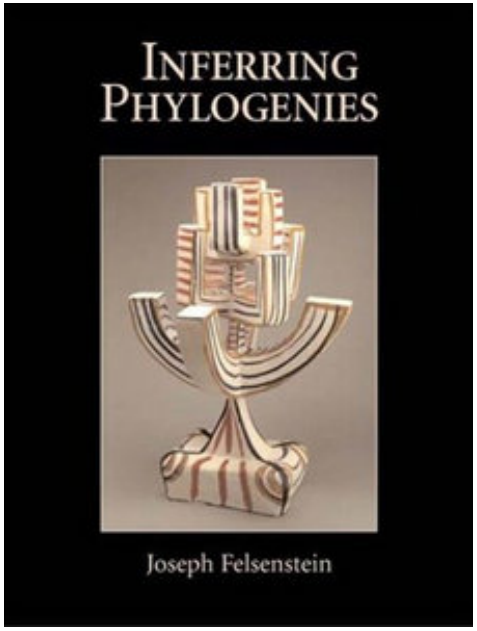



Figure: Figures from Joe Felsenstein, *Inferring Phylogenies*, 2004

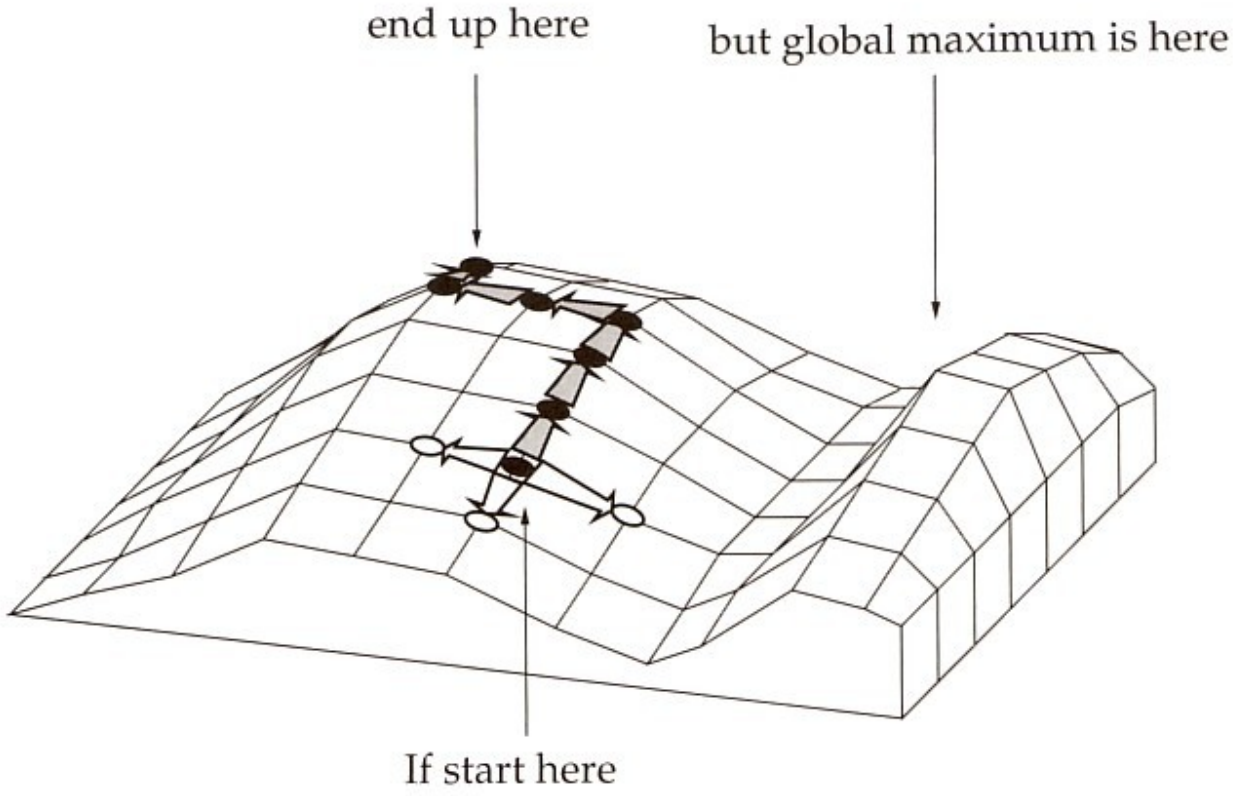
# The ‘big parsimony problem’

$\Rightarrow$ ‘Heuristic’ algorithms necessary, *i.e.* algorithms that find *approximate* solution quickly.

Popular method: *hill climbing*. Start with arbitrary topology $T$, consider ‘neighbours’ of $T$, *i.e.* similar topologies, go to best neighbour topology, *etc.* until no further improvement can be achieved.

Disadvantage: may end up in local maxima / minima.

# The ‘big parsimony problem’




Searching best solution for optimization problem by *hill climbing*.
Height in landscape represents quality of solution.

# The ‘big parsimony problem’

To find optimal topology:

- Start with initial topology
- Gradually improve topology by moving to ‘neighbouring’ topologies.

Different possibility to define ‘neighbouring’ topology.

General assumption: trees without root

# The ‘big parsimony problem’

How to define ‘neighbours’ of tree topologies?

*1. Possibility‘: Nearest-neighbor interchanges:*

- Consider *internal* edges in topology $\mathcal{T}$, *i.e.* edges *not* directly adjacent to a leave.
- For each internal edge $e$ consider four sub-trees $S$, $T$, $U$, $V$ connected by $e$
- There are three possibilities to connect $S$, $T$, $U$, $V$, *i.e.* two additional possibilities not realized in $\mathcal{T}$.
- Consider these topologies as *neighbours* of $\mathcal{T}$.

# The ‘big parsimony problem’

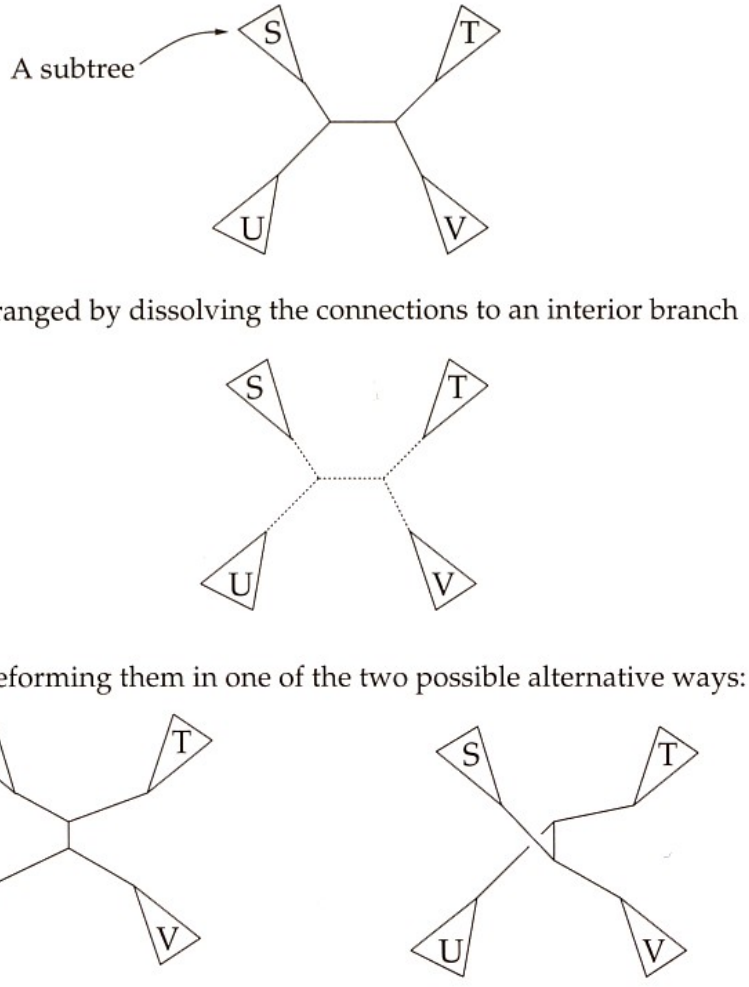



Figure: Nearest-neighbour interchanges

# The ‘big parsimony problem’

For $n$ leaves, there are $2n - 3$ edges in an unrooted tree.

Thus, there are $n - 3$ internal edges.

*Rightarrow* For a toplolgy with $n$ leaves, there are $2n - 6$ neighbouring topologies.

# The ‘big parsimony problem’

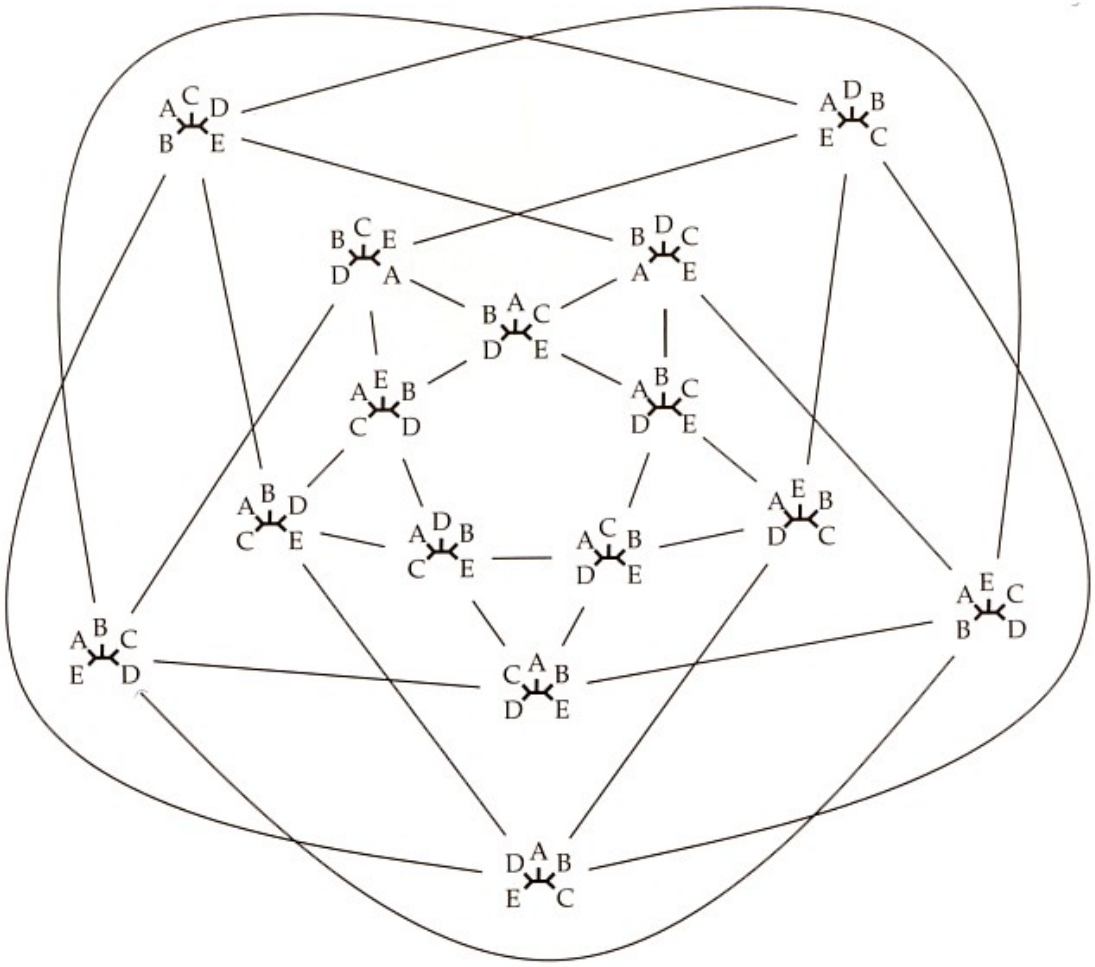


Figure: ‘Landscape’ of possible solutions (topologies) for *hill climbing* with *nearest-neighbour interchanges*

# The ‘big parsimony problem’

| | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|
| A | 1 | 0 | 0 | 1 | 1 | 0 |
| B | 0 | 0 | 1 | 0 | 0 | 0 |
| C | 1 | 1 | 0 | 0 | 0 | 0 |
| D | 1 | 1 | 0 | 1 | 1 | 1 |
| E | 0 | 0 | 1 | 1 | 1 | 0 |

Figure: Example (from Felsenstein): data matrix for 5 ‘species’ and 6 ‘characters’

# The ‘big parsimony problem’

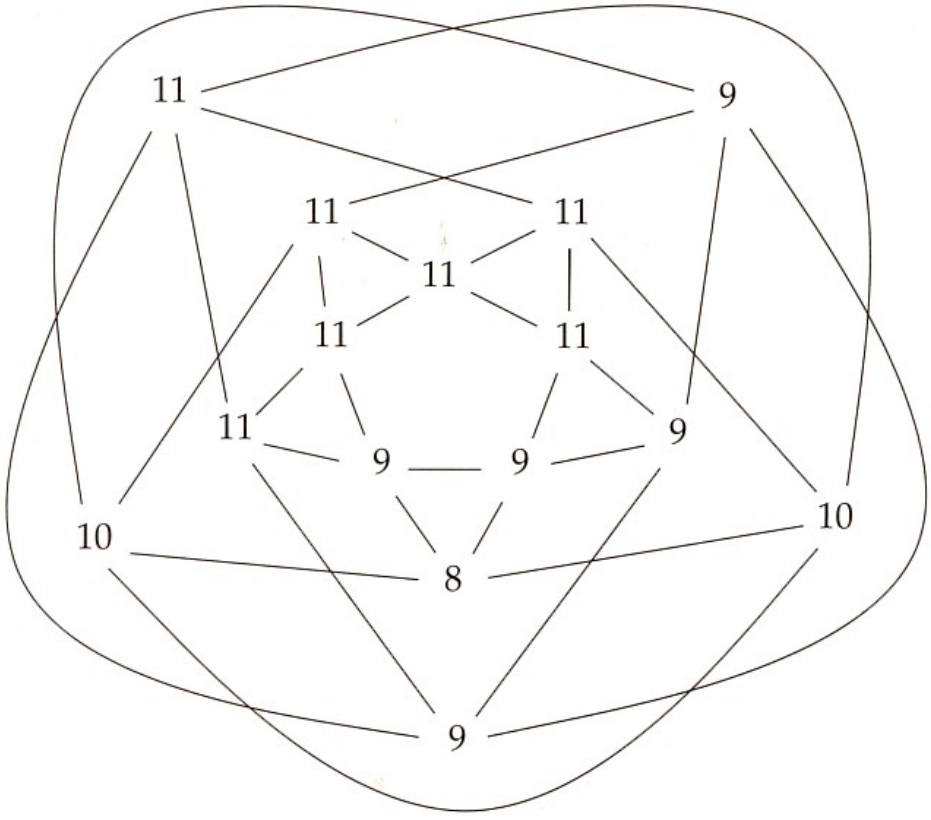


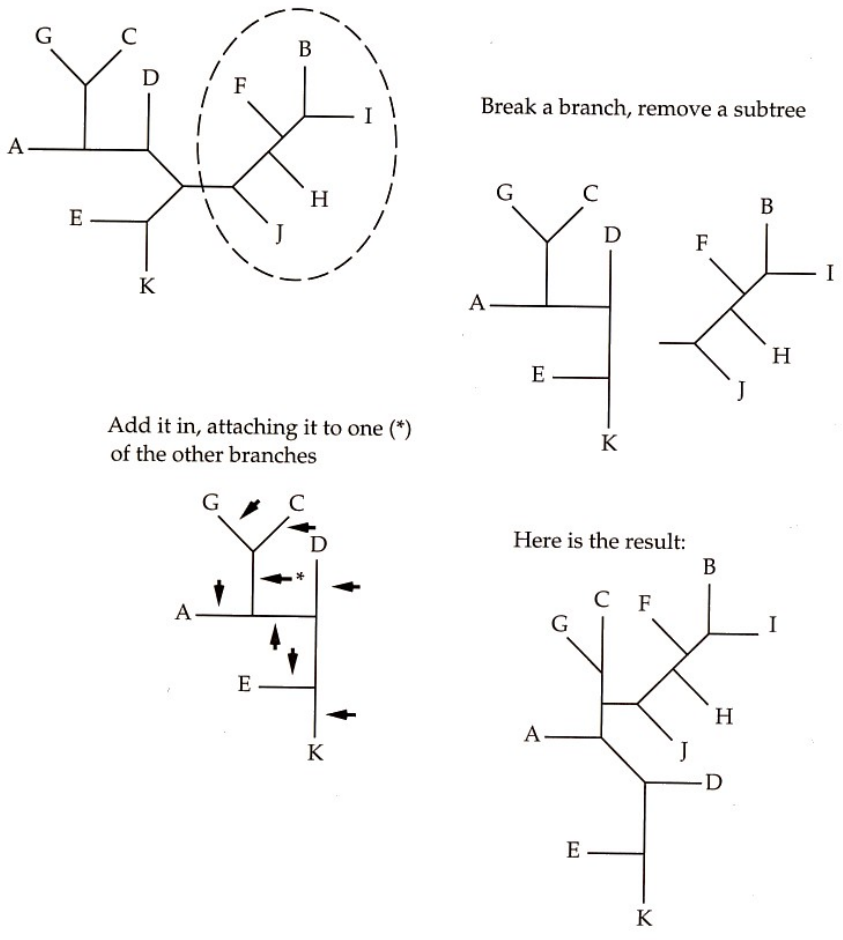

Figure: ‘Scores’ for all possible solutions (= topologies), *i.e.* number of required substitutions. In *this* example, optimal solution found by *hill climbing* (independently of starting point).

# The ‘big parsimony problem’

*2. Subtree-pruning and regrafting:*

Cut off sub-tree $T'$ from $T$, plant $T'$ into arbitrary edge in remaining sub-tree of $T$.

# The ‘big parsimony problem’



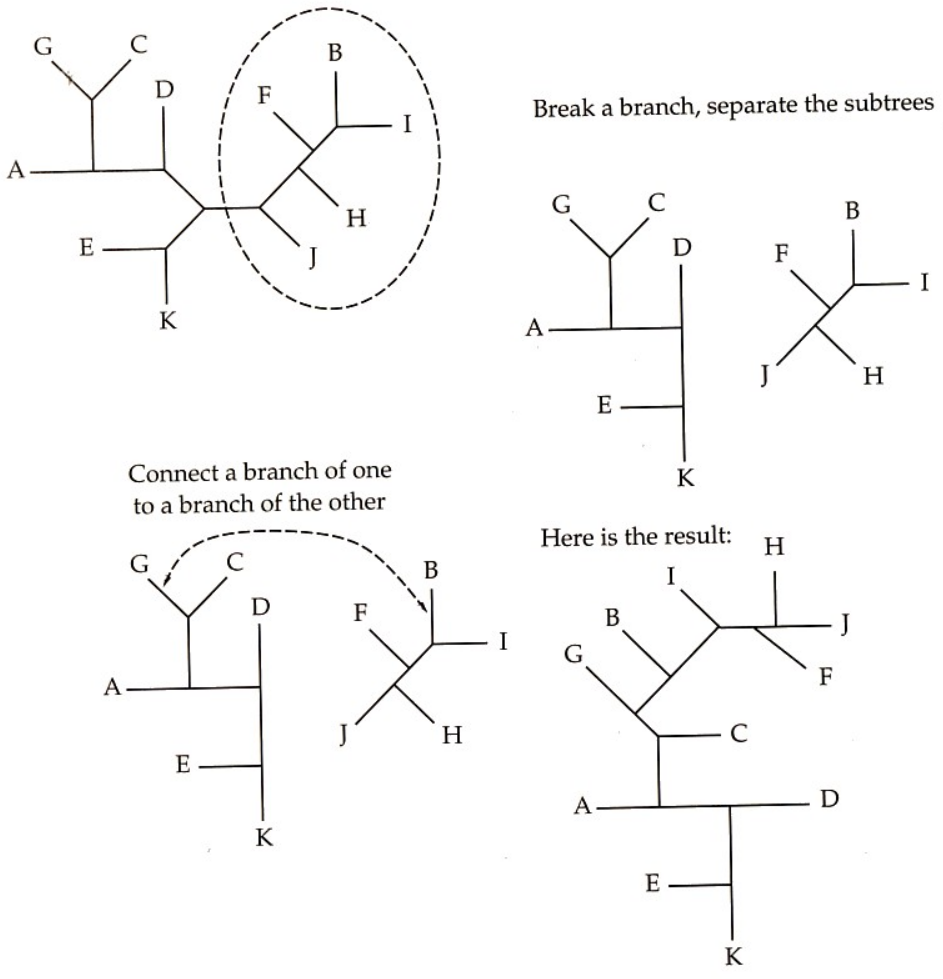

Neighbouring trees with ‘subtree pruning and regrafting’

# The ‘big parsimony problem’

*3. Subtree bisection and reconnection*

Variant of *subtree pruning*:
Split $T$ into sub-trees $T_1$ and $T_2$; glue $T_1$ and $T_2$ together arbigrarily.

# The ‘big parsimony problem’




Neighbouring trees with ‘subtree bisection and reconnection’

# The ‘big parsimony problem’

- All three possibilities allow to transform every possible tree topology into every other topology.
- Important: Start *hill climbing* with good initial tree $T_0$