problem such as "maximum parsimony" or "maximum likelihood". Here, algorithms play a secondary role as a means of obtaining or approximating a solution to the optimization problem.

Distance methods are usually *algorithmically* defined and have polynomial run time. Sequence methods usually involve solving an *optimization* problem by finding an optimal tree with respect to some criterion and are usually NP-hard.

Algorithmic methods combine the computation and definition of the preferred tree into a single statement.

Optimization methods involve formulating and solving two problems:

- computing the cost for a given tree T, and
- searching through all trees to find a tree that minimizes the cost.

11.18 Maximum parsimony

The maximum parsimony method is perhaps the most widely used sequence-based tree reconstruction method.

In science, the principle of *maximum parsimony* is well known: always use the simplest, *most parsimonious* explanation of an observation, until new observations force one to adopt a more complex theory.

In phylogenetic analysis, the *maximum parsimony problem* is to find a phylogenetic tree that explains a given set of aligned sequences using a minimum number of "evolutionary events".

11.18.1 The parsimony score of a tree

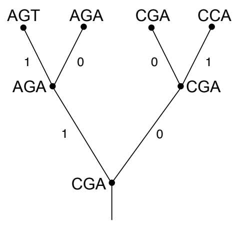
The *difference* between two sequences $x = (x_1, \dots, x_L)$ and $y = (y_1, \dots, y_L)$ is simply their non-normalized Hamming distance

$$\text{diff}(x, y) = |\{k \mid x_k \neq y_k\}|.$$

Given a multiple alignment of sequences $A = \{a_1, a_2, \dots, a_n\}$ and a corresponding phylogenetic tree T, leaf-labeled by A.

If we assign a hypothetical ancestor sequence $\lambda(v)$ to every internal node v in T, then we can obtain a score for T together with this assignment, by summing over all differences diff(x, y), where x and y are the sequences labeling two nodes that are joined by an edge in T.

Example:



$\Rightarrow$ Score = 3

The minimum value obtainable in this way is called the *parsimony score* $PS(T, A)$ of T and A.

Definition 11.18.1 (Parsimony score) *Given a multiple alignment of sequences A of length L and a corresponding phylogenetic tree T, leaf-labeled by A, where, in addition, all inner vertices are labeled by sequences of length L, its* parsimony score *can be defined as:*

$$PS(T, A) = \min_{\lambda} \sum_{\{u,v\} \in E} \text{diff}(u, v),$$

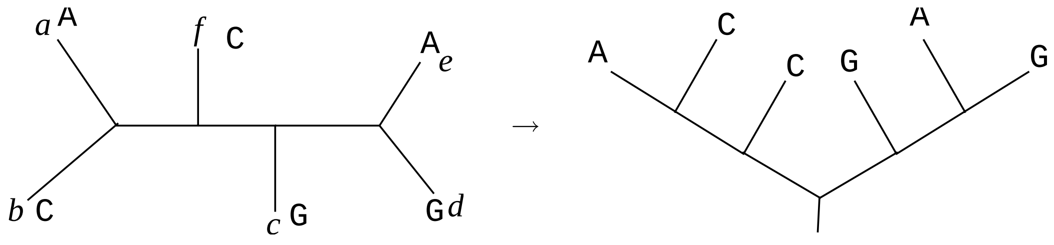
where the minimum is taken over all possible labelings λ of the internal nodes of T.

11.18.2 The Small Parsimony Problem

Problem 11.18.2 (Small Parsimony Problem) *The* Small Parsimony Problem *is to compute the parsimony score for a given tree T.*

Can it be solved efficiently?

As the parsimony score is obtained by summing over all columns, the columns are independent and so it suffices to discuss how to obtain an optimal assignment for one position:



unrooted tree rooted tree

11.18.3 The Fitch algorithm

In 1971, Walter Fitch[7] published a dynamic programming algorithm that solves the small parsimony problem efficiently.

Input: A phylogenetic tree T, with n leaves, and a single character c with a set Σ of k possible values. Denote the value of the character for node v by $c(v)$.

The Fitch algorithm consists of two steps/passes, the *forward pass* and the *backward pass.*

Let $z = 0$. We will assign to each node v a set $S(v) \subseteq \Sigma$ in the following fashion:

For each leaf v: $S(v) = \{c(v)\}$, $l(v) = 0$

For each inner node v with children u, w:

$$S(v) = \begin{cases} S(u) \cap S(w), & S(u) \cap S(w) \neq \emptyset \\ S(u) \cup S(w), & S(u) \cap S(w) = \emptyset \end{cases}$$

$$S(u) \cap S(w) \neq \emptyset$$
$$S(u) \cap S(w) = \emptyset \quad z := z + 1$$

To compute $S(v)$ and z we will traverse the tree in postorder - starting with the leaves and working our way down to the root. The parsimony score is then given by z.

In total, the algorithm requires $O(nL)$ steps.

The following algorithm computes the parsimony score for T and a fixed column c in the sequence alignment. Initially it is called with $e = null$ and v the root node and $PS(T, c) = 0$.

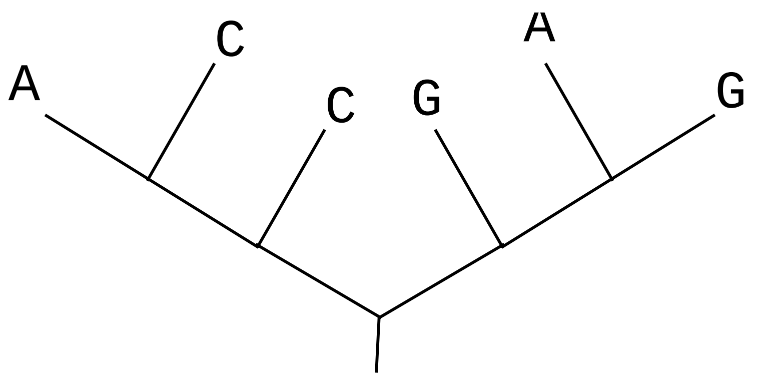
Algorithm 11.18.3 (ParsimonyScore(e, v), Walter Fitch, 1971)

```
Input: A binary phylogenetic tree $T$ and a character $c(v)$ for each leaf $v$
Output: The parsimony score $PS(T, c)$ for $T$ and $c$
if $v$ is a leaf node then
    set $S(v) = \{c(v)\}$
else
    for the two edges $f_1, f_2 \neq e$ adjacent to $v$ do
        Let $w_i$ be the opposite node of $f_i$
        Call ParsimonyScore$(f_i, w_i)$ // to compute $S(w_i)$
    if $S(w_1) \cap S(w_2) \neq \emptyset$ then
```

[7]Fitch, W. (1971) Toward defining the course of evolution: minimum change of specified tree topology. Syst. Zoology 20:406-416

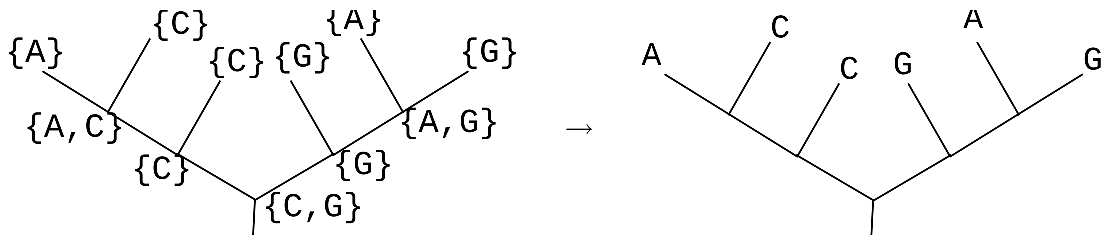
```
        Set $S(v) = S(w_1) \cap S(w_2)$
    else set $S(v) = S(w_1) \cup S(w_2)$ and $PS(T, c) = PS(T, c) + 1$
```

Example: we use the Fitch algorithm to compute the parsimony score for the following labeled tree:



The forward pass algorithm computes the parsimony score. An optimal labeling of the internal nodes is obtained via traceback:

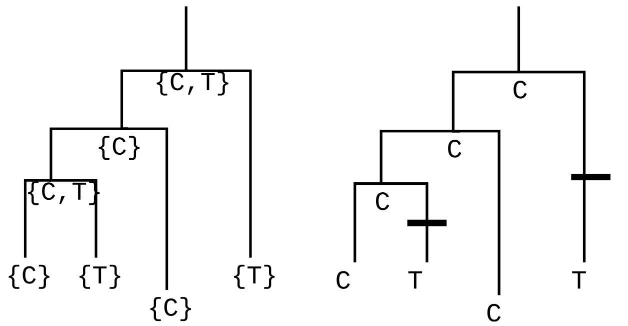
Starting at the root node r, we label r using any character in $S(r)$. Then, for each child w, we use the same letter, if it is contained in $S(w)$, otherwise we use any letter in $S(w)$ as label for w. We then visit the children von w etc:



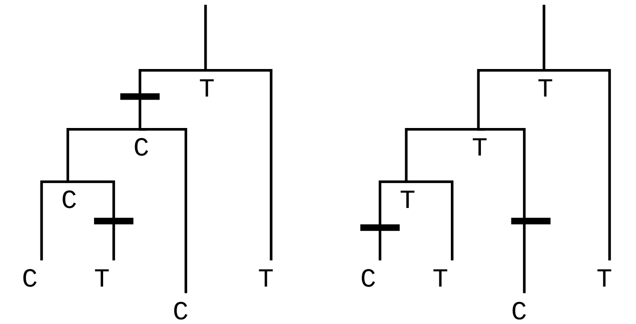
This algorithm also requires $O(nL)$ steps, in total.

11.18.4 Traceback

Example:



Fitch labeling one traceback result



another traceback result not obtainable by traceback

The following algorithm computes the assignments of the internal nodes for T and a fixed column c in the sequence alignment. Initially it is called with $e = null$ and r the root node.

Algorithm 11.18.4 (Fitch algorithm, traceback)

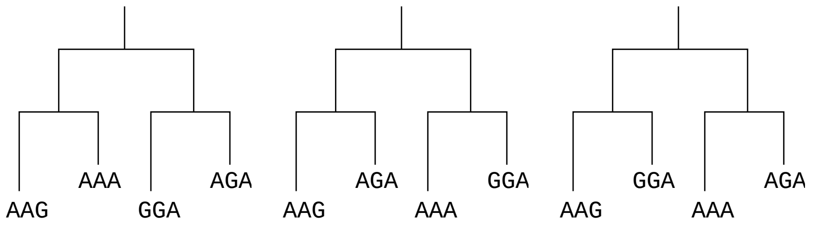
```
Input: A phylogenetic tree $T$ and the set $S(v)$ of each node $v$ in $T$
Output: The fully labeled tree $T$
for each node $v$, in breath-first order, starting at the root do
    if $v$ is the root node then
        Set $c(v) = t$ with $t \in S(v)$
    else
        Let $w$ be the parent node of $v$
        if $c(w) \in S(v)$ then set $c(v) = c(w)$
        else set $c(v) = t$ for some $t \in S(v)$
end
```

11.18.5 A simple example

Assume we are given the following four aligned sequences:

```
Seq 1:  A  A  G
Seq 2:  A  A  A
Seq 3:  G  G  A
Seq 4:  A  G  A
```

There are three possible binary (non-rooted) topologies on four taxa:



In each tree, label all internal nodes with sequences so as to minimize the score obtained by summing all mismatches along edges. Which tree minimizes this score?

11.18.6 The Fitch algorithm on unrooted trees

Above, we formulated the Fitch algorithm for rooted trees. However, the minimum parsimony cost is independent of where the root is located in the tree T:

To see this, consider a root node ρ connected to two nodes v and w, and let $c(\cdot)$ denote the character assigned to a node in the Fitch algorithm:

1. If $c(v) = c(w)$, then $c(\rho) = c(v) = c(w)$, because any other choice would add 1 to the score.
2. If $c(v) \neq c(w)$, then either $c(\rho) = c(v)$ or $c(\rho) = c(w)$, and both choices contribute 1 to the score.

In both cases, we could replace ρ by a new edge e that connects v and w without changing the parsimony score of T.

11.18.7 The large parsimony problem

Definition 11.18.5 (Parsimony score of an alignment) *Given a multiple alignment* $A = \{a_1, \ldots, a_n\}$, *its* parsimony score *is defined as*

$$PS(A) = \min\{PS(T, A) \mid T \textit{ is a phylogenetic tree on } A\}.$$

Problem 11.18.6 (Large Parsimony Problem) *The* Large Parsimony Problem *is to compute* $PS(A)$.

Potentially, we need to consider all $(2n-5)!!$ possible trees. Unfortunately, in general this can't be avoided and the maximum parsimony problem is known to be NP-hard.

Exhaustive enumeration of all possible tree topologies will only work for $n \leq 10$, say.

Thus, we need more efficient strategies that either solve the problem exactly, such as the "branch and bound" technique, or return good approximations, such as "heuristic searches".

11.19 Tree Searching Methods

- Exhaustive search (exact)
- Branch and bound search (exact)
- Heuristic search methods (approximate)
  - Stepwise addition
  - Star decomposition
  - Branch swapping

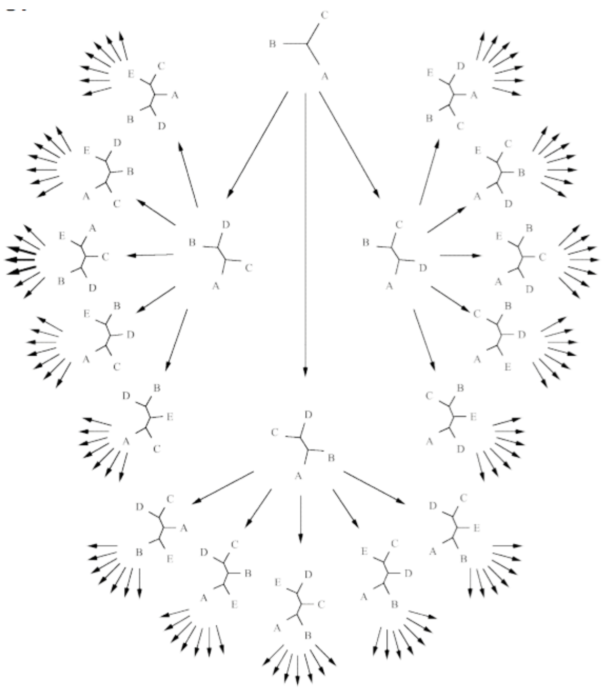
11.19.1 Branch and bound

Recall how we obtained an expression for the number $U(n)$ of unrooted phylogenetic tree topologies on n taxa:

For $n = 3$ there are three ways of adding an extra edge with a new leaf to obtain an unrooted tree on 4 leaves. This new tree has $(2n-3) = 5$ edges and there are 5 ways to obtain a new tree with 5 leaves etc.

To be precise, one can obtain any tree T_{i+1} on $\{a_1, \ldots, a_i, a_{i+1}\}$ by adding an extra edge with a leaf labeled a_{i+1} to some (unique) tree T_i on $\{a_1, \ldots, a_i\}$.

In other words, we can produce the set of *all* possible trees on n taxa by adding one leaf at a time in all possible ways, thus systematically generating a complete *enumeration tree.*

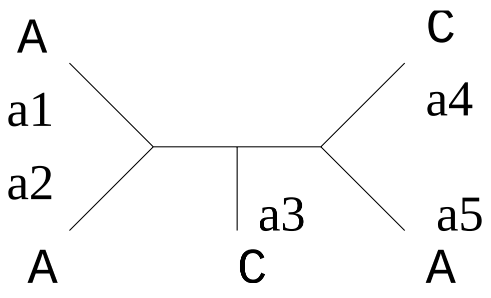


A simple, but crucial observation is that adding a new sequence a_{i+1} to a tree T_i to obtain a new tree T_{i+1} cannot lead to a smaller parsimony score.

This gives rise to the following bound criterion when generating the enumeration tree:

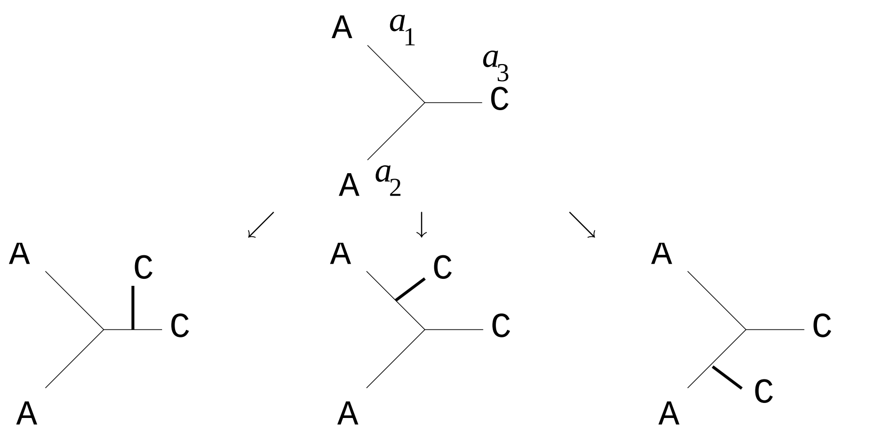
> If the *local* parsimony score of the current incomplete tree T' is larger or equal to the best *global* score for any complete tree seen so far, then we do not generate or search the enumeration subtree below T'.

Example:
Assume we are given an msa $A = \{a_1, a_2, \ldots, a_5\}$ and at a given position i the characters are: $a_{1i} =$ `A`, $a_{2i} =$ `A`, $a_{3i} =$ `C`, $a_{4i} =$ `C` and $a_{5i} =$ `A`. Assume that the Neighbor-Joining tree on A looks like this:



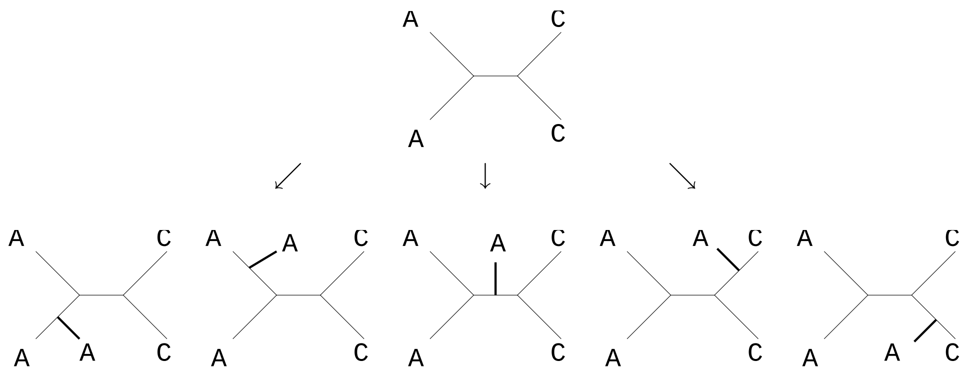
The PS is =2, thus we initialize *global* bound with 2 for position i.

The first step in generating the enumeration tree is this:



local $= 1 <$ *global* continue | *local* $= 2 \not< $ *global* don't continue | *local* $= 2 \not<$ *global* don't continue

Only the first of the three trees fulfills the bound criterion and we do not pursue the other two trees. The second step in generating the enumeration tree looks like this:



local $= 1$ *local* $= 1$ *local* $= 1$ *local* $= 2$ *local* $= 2$

The first three trees are optimal. Note that the bound criterion in the first step reduced the number of full trees to be considered by two thirds.

Application of branch-and-bound to evolutionary trees was first suggested by Mike Hendy and Dave Penny (1982)[8].

In practice, using branch and bound one can obtain exact solutions for data sets of twenty or more sequences, depending on the sequence length and the "messiness" of the data.

A good starting strategy is to first compute a tree T_0 for the data, e.g. using Neighbor-Joining, and then to initialize the *global* bound to the parsimony score of T_0.

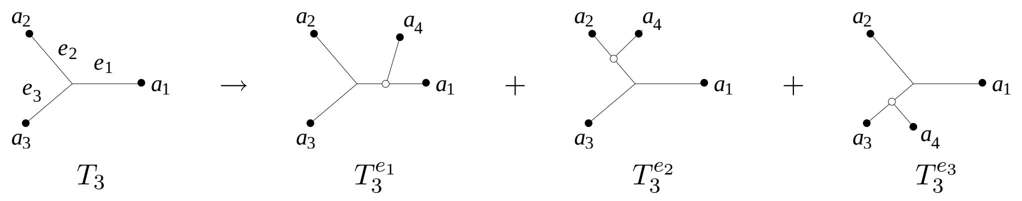
11.19.2 The stepwise-additional heuristic

Now we discuss a simple greedy heuristic (Felsenstein 1981) for approximating the optimal tree or score. We build the tree T by adding one leaf after the other, in each step choosing the optimal position for the new leaf-edge:

Given a multiple sequence alignment $A = \{a_1, a_2, \ldots, a_n\}$, start with a tree T_2 consisting of two leaves labeled a_1 and a_2.

Given T_i, for each edge e in T_i, obtain a new tree T_i^e as follows: Insert a new node v in e and join it via a new edge f to a new leaf w with label a_{i+1}. Set $T_{i+1} = \arg\min\{P(T_i^e, A)\}$.

[8]Hendy, M. D. and Penny, D. (1982). Branch and bound algorithms to determine minimal evolutionary trees. Mathematical Biosciences 59: 277-290



Obviously, this approach does not guarantee an optimal result. Moreover, the result obtained will depend on the order in which the sequences are processed.

11.19.3 The star-decomposition heuristic

This employs a similar strategy to Neighbor-Joining. We start with a star tree on all n taxa. At each step, the optimality criterion is evaluated for every possible joining of a pair of lineages incident to the central node. The best tree found at one step is then used as the basis of the next step: