1 Brief Review:

In the previous lecture we talked about parsimony trees. we saw that using the parsimony method given a tree with specific structure and leaves, we could calculate what is the minimum number of changes needed on all the possible inner nodes to represent a given group of sequences, but this parsimony method uses a key assumption that there is a given tree structure, to work with. the question remains, how do we find a tree that will minimize the number of changes on the inner nodes?
it’s obvious we can’t go over all possible trees and check them all, cause there is an exponential number of trees, and this would be a very hard task.

2 Most Parsimonious Tree:

so the problem in front of us:

In: given MSA
Out: most parsimonious tree T

This problem is NP-hard, thus we don’t have an efficient algorithm that will work in all cases. we will need to compromise on a more simple problem to solve that will help us estimate the most parsimonious tree. Lets look on two ways people have suggested to solve this problem:

2.1 Branch & Bound:

assume we have some n-dimensional world in which we would like to find the maximum or the minimum. as shown in Figure 1 say we mark the world as

\[ \Omega \]

in this world we would like to find a minimum to a given function

\[ f() \]

we could divide the world in sections

\[ A, B \ldots \]

so that for each of them

\[ A \subseteq \Omega \]

and then we will look for the lower bound in each section

\[ l(A) \leq \min_{x \in A} f(x) \]
now how does this simplifies the problem?
well the intuition for this lies in the fact that for a smaller section the problem will be easier to calculate and given the lower bound in some sections we could cast upon some of the others and disregard them if they have a higher lower bound. for example:

Example 2.1 if in section $B$ there is a point $y \in B$ so that $f(y) \leq l(A)$

$y$ could be a global minimum or a local minimum maybe even local in section $B$ but anyhow this would imply that the global minimum is for sure not in $A$ and we could from now on disregard section $A$.

To make it simple, lets assume each devision into sections is binary. in other words two sections in each devision.
this devision defines a binary tree, in which the inner nodes are sub-sections of the world and in the leaves are points in the world.

now back to our problem, we would like to find the leaf in this tree with the lowest value. but this tree is enormous and traversing all of it will be exponentially big, and this is not surprising since we started with an NP-hard problem. so we will need to find an algorithm that will find what we desire but will spare us from entering non-relevant sub-trees on the way.

such algorithm will use DFS to traverse the tree, and before it enters a sub-tree it will check against the lowest candidate in the sub-tree and by it decide if it should enter that sub-tree or not. so actually for each inner node $A$ the algorithm will calculate $l(A)$.

and when in a node, given it’s two children it will decide with which sub-tree under which child to proceed with using this $l(\cdot)$ lower bound.

Note: take caution, in our original problem all the points in the world are actually trees themselves - evolutionary letter trees.

Example 2.2 say we have $n$ different species -

\begin{align*}
1 & \ldots n
\end{align*}

we would like a tree with $n$ nodes for example lets look at an example tree as in Figure 2 and reduce it from 6 to 4 leaves as in Figure 3 is consistent with the original tree but the tree in Figure 4 is not consistent
now how do we create all the consistent trees from an original tree?
we start with an initial consistent small tree with three nodes (e.g. Figure 5) for three species any 1 . . . n tree are consistent. now we would like to divide our space by the possibilities of adding the fourth node for three subgroups of consistent trees as in Figure 6
now each subgroup can be divided into subgroups by a fifth species. now we will define a lower bound by it we could disregard the non-relevant subgroups.

but this division is exponential.
so we should think of other methods - heuristics. they might not offer the best or optimal solution but they will run in a reasonable time.

2.2 Some Heuristics:

there are a few of heuristics we could use on this problem.

• one possible heuristic method would be to use the DFS method from before with disregarding subtrees and after some reasonable time with no result stop and start over.

• an other heuristics would be to go over all the species and in each step to add the species in the place that results in the highest score.

• another one can be to traverse the space of the trees, we would start with a specific tree and change it just a bit in every step. and we will try to preserve good characteristics from step to step.

the most common method to do this is - grafting:
you take off a subtree and graft it in an other valid location, now there something like $n^2$ possible options the number of branches (edges) is

$$\frac{1}{2}(n + 3(n - 2)) = 2n - 3$$

and so for each branch we take off there are $O(n)$ positions where we could graft it and there are $O(n)$ branches we can take off so on total we have $O(n^2)$ grafting possibilities.

we have talked about gradient ascent before in continuous spaces using a greedy search this is similar but discrete. we assume that for a good tree, it’s neighbour trees are also good. since the number of neighbours is big we can sample just part of the neighbours, and sometimes go backwards so we wont get stuck in a local minimum. we can spare on calculations since when we change only a part of the tree there is still a part that is already calculated.
3 Summery:

so we are left with the questions, are we really interested in all these trees? is this good? what is it good for? does the resulting tree really give the right evolutionary course? if we look only on trees for four species, people have made these changes in the trees branches, and saw areas with specific relations, but still the resulting tree was inferior. articles were published on the subject, but all of them with had the conclusion that only specific trees are good.

most parsimonious tree methods give a good measure on what really happened in the evolution. but the parsimony methods also make mistakes on this kind of data, when people checked the performance they found that the most parsimonious tree is good only in a third of the cases.

so to solve our problem, we need to move to methods that count changes and calculate the probability of the change over evolution. after the semester vacation we will advance on the subject.
4 Summery Of The Semester:

what did we learn in the first semester:

• Sequence Alignment
  – Dynamic Programing
  – Distance Matrix
  – E-value, P-value
  – Multiple Sequence

• Information Theory

• Bayesian Estimation

• HMMs - Hidden Markov Models
  – EM
  – Dynamic Programing

• Motif Finding

• Phylogeny

and what will be in the following semester?

• Gene Expression

• Structure
  – RNA
  – Protein

• Networks
  – Regulatory - $TF \rightarrow Target$
  – Interaction / Complex

• Competitive Genomics