Algorithmic Bioinformatics DD2450, spring 2010, Lecture 10

Lecturer Jens Lagergren
Several current and previous students
will be acknowledged in a separate document.

May 22, 2010

Phylogenetics is the study of how species are related through evolution. The relationships between different species are represented in a phylogenetic tree, where proximity in the tree is correlated with similarity. One procedure for building a phylogenetic tree is to do the following.

1. Choose a gene family.
2. Pick one gene from each species of interest.
3. Make a multiple sequence alignment of the genes.
4. Discard columns that contain blanks.
5. Construct the tree using the similarity between the genes.

A problem with this procedure is that some gene families have an evolutionary history that deviates from the species'; the so called gene tree differs from the species tree. Gene duplications can for instance have this effect. For this reason we must take care when selecting the gene family. We will however ignore this problem from here on, and instead study the following two phylogenetic problems.

- **Big problem**: Find the phylogenetic tree that best describes a set of sequences.
- **Small problem**: Given a set of sequences and a tree, score the tree.

1 Parsimony

Notation 1. *Let T be a binary rooted tree.*

- *$E(T)$ is the set of edges in T.*
- *$V(T)$ is the set of vertices in T.*

- *$L(T)$ is the set of leaves in T.*
- *$R(T)$ is the root vertex of T.*
- *T_v is the subtree of T that has v as root.*

Definition 1. *Let T be a binary rooted tree and Σ be an alphabet.*

- *A labeled tree is a pair $\langle T, l\rangle$ where $l : V(T) \to \Sigma^m$.*
- *A leaf-labeled tree is a pair $\langle T, e\rangle$ where $e : L(T) \to \Sigma^m$.*

We will be working with aligned genes, i.e. strings in Σ^m where $\Sigma = \{A, C, G, T\}$. We will use the Hamming distance as distance measure. The Hamming distance of two strings $s, s' \in \Sigma^m$ is

$$h(s, s') = \text{the number of positions where } s \text{ and } s' \text{ are different}$$

We use the Hamming distance to produce a cost function for labeled trees.

$$H(T, l) = \sum_{(u,v)\in E(T)} h(l(u), l(v))$$

Our goal is to find the labeled tree of minimum cost that, when restricted to the leaves, has labels equal to a given leaf-labeling. We write the latter condition as $l|_{L(T)} = e$. The cost of a leaf-labeled tree $\langle T, e\rangle$ is given by

$$\mu(T, e) = \min_{l|_{L(T)}=e} H(T, l)$$

1.1 Big problem

Input: $s_1, \ldots, s_n \in \Sigma^m$

Output: Leaf-labeled tree $\langle T, e\rangle$ such that

1. e is a bijection between $L(T)$ and $\{s_1, \ldots, s_n\}$.
2. $\mu(T, e)$ is minimal with respect to (1).

This problem is NP-complete. There exist approximation algorithms that can approximate the solution within a factor around 1.57. Those algorithms are not covered in this course, instead we will use heuristic methods.

1.2 Small problem

Input: Leaf-labeled tree $\langle T, e\rangle$.

Output: A labeling l such that

1. $l|_{L(T)} = e$
2. $\mu(T, e) = H(T, l)$

The cost can be computed one position at a time. I.e. we can solve a problem instance with labels in Σ^m by solving m problem instances with labels in Σ. Therefore we will focus on solving the problem for $m = 1$.
This problem can be solved using dynamic programming. The subproblem is to, given a tree with a labeled root, calculate the cost $\mu(T, e)$. Let $\mu(T, e, N)$ be the minimum cost of the leaf-labeled tree $\langle T, e \rangle$ when the root is labeled $N \in \Sigma$, i.e.

$$\mu(T, e, N) = \min_{\substack{l : V(T) \to \Sigma \\ l|_{L(T)} = e \\ l(R(T)) = N}} H(T, l)$$

Let $c(v, N) = \mu(T_v, e|_{L(T_v)}, N)$. This gives the following base cases for $v \in L(T)$

$$c(v, N) = \begin{cases} 0 & \text{if } N = e(v) \\ \infty & \text{otherwise} \end{cases}$$

and the following recursion for $u \in V(T) \backslash L(T)$ with children v and w.

$$c(u, N) = \min_{N_v, N_w \in \Sigma} h(N_v, N) + c(v, N_v) + h(N_w, N) + c(w, N_w)$$

We iterate over the instances in such a way that the children of a vertex are always visited before the vertex itself. The solution is constructed by starting from the optimal root-labeling and then following back-pointers stored during the recursion. This dynamic programming algorithm has a time complexity of $O(n|\Sigma|^3)$ where $n = |V(T)|$. We run the algorithm m times, so in total we get a time complexity of $O(nm|\Sigma|^3)$.
The complexity of the algorithm can be improved by individually minimizing the cost resulting from each subtree, since they are independent during minimization. This gives the following change in the recursion, which reduces the complexity by a factor $|\Sigma|$.

$$c(u, N) = (\min_{N_v \in \Sigma} h(N_v, N) + c(v, N_v)) + (\min_{N_w \in \Sigma} h(N_w, N) + c(w, N_w))$$

We can also note that, for each of the two minimizations, either N or the optimum at the subtree's root will minimize the expression. Hence we only have to test a constant number of alternatives, allowing us to compute the cost in constant time. The complexity hence lands at $O(nm|\Sigma|)$.

2 Branch swap

A branch swap is a tree operation. The picture below shows a branch swap applied to the center edge of the tree to the left, producing the tree on the right.

- See figure 1

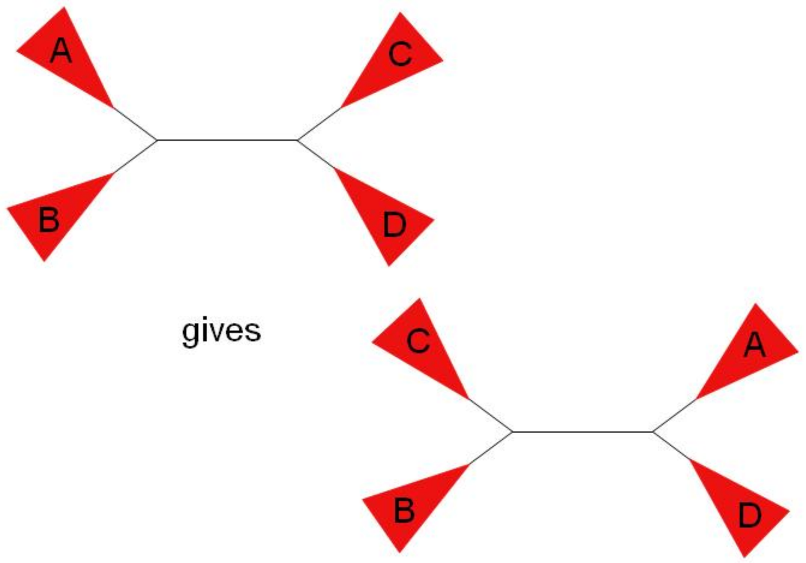

Figure 1: Branch Swapping

The operation can be combined with hill climbing to search for a good tree.