CSC 2427 - Algorithms in Molecular Biology

Lecture 8: February 3 2006

Lecturer: Michael Brudno
Scribe Notes by: Clement Chung

Background

Recall from previous lecture we can find the alignment between two DNA sequences by either performing global alignment using Needleman-Wunsch algorithm or local alignment using Smith-Waterman algorithm. The method the two algorithms used to handle gaps is of particular interest. Thus far the strategy is to add a fixed gap penalty when a gap occurs regardless what the alignment was for the previous nucleotide residue (see Figure 1). Below is the algorithm for calculating the alignment score based on linear gap penalty. This algorithm has both running time and space usage that is linear O(nm) .

$$\begin{aligned} M(i,j) = \max(&M(i-1,j-1)+s(A_i,B_j),\\ &M(i-1,j)+GAP,\\ &M(i,j-1)+GAP,\\ &0) \end{aligned}$$

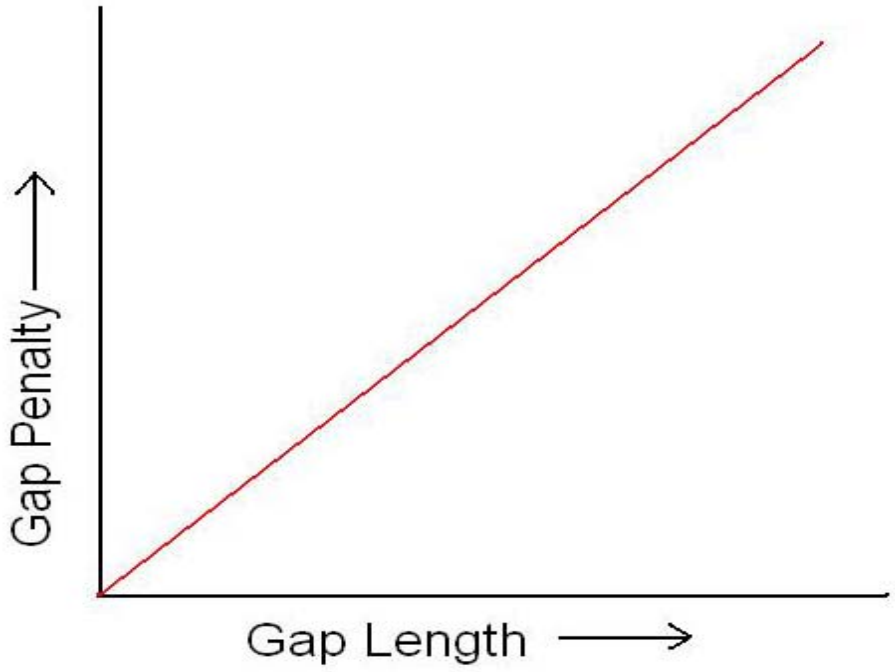

Figure 1: Linear gap penalty curve.

Ideal Gap Penalty

It is known that when gaps occur in an alignment they occur in bunches. This happens because insertion and deletions are *block* events, with a number of residues inserted/deleted at a time. Due to this fact it is likely that if a particular character is gapped, the probability of the next one being gapped is higher, and hence should be penalized less. To correctly model this phenomenon we will use a convex gap function where the gap penalty increases less and less as the gap extends further (see Figure 2), which when plotted is similar to a log curve. The modified algorithm is as follows, with GAP being any function:

$$\begin{aligned} M(i,j) = \max(&M(i-1,j-1)+s(A_i,B_j),\\ &\max_k(M(i-k,j)+GAP(k)),\\ &\max_k(M(i,j-k)+GAP(k)),\\ &0) \end{aligned}$$

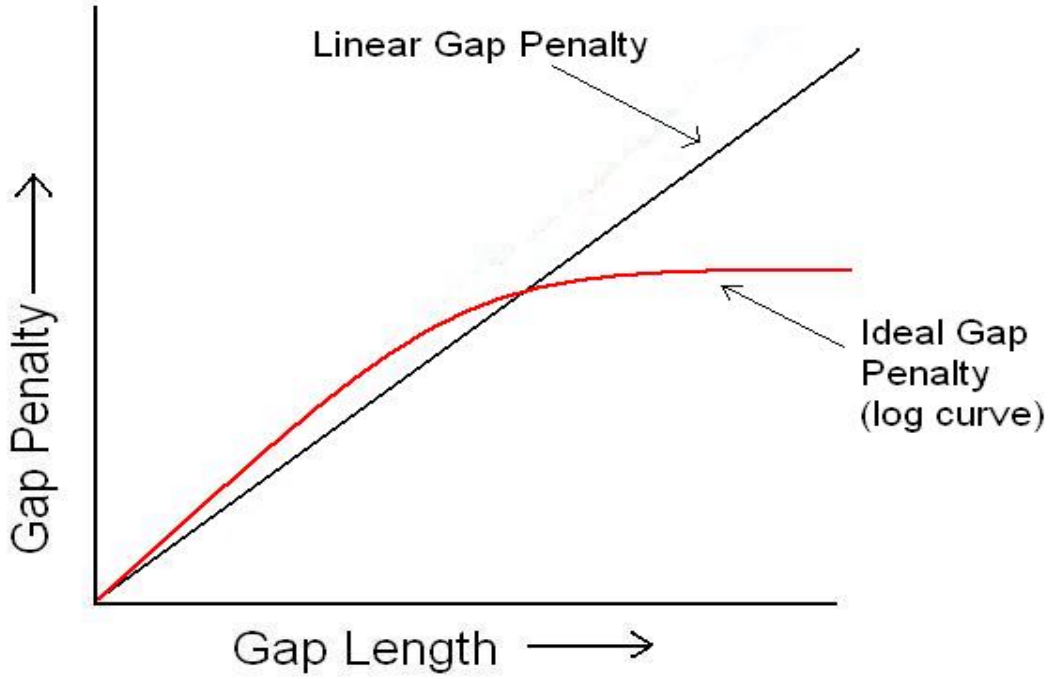

Figure 2: Ideal gap penalty curve. For large gap length the linear gap penalty is not a good representation of the ideal gap penalty.

Note that the algorithm above, which models the convex gap function, is very inefficient. When evaluating the gap penalty we need to loop through all previous nucleotides (k) to find the one that gives the maximum score. The running time for this algorithm is $O(n^2m)$, assuming n>m, and space O(nm).

Gotoh's Algorithm and Affine Gap Penalty

To make the algorithm more tractable we will use Gotoh's algorithm [1], which allows for affine gap penalty. The idea is to have a gap penalty for opening a gap and a different gap penalty for extending the gap, which is smaller then gap opening penalty (see Figure 3). This is a better fit to the ideal gap penalty model then the linear gap penalty. Also, it requires the algorithm to only know if the previous alignment was to a gap or not.

The implementation of dynamic programming for this algorithm has time complexity of O(nm), similar to the algorithms using linear gap penalty. However, now we need to keep track multiple values for each pair of residue coefficient (i.e. each cell in the original score matrix) instead of just the alignment score. Similar to the other algorithms we need to calculate the best score up to residues (i, j) given that x_i and y_j is aligned, $M(i, j)$. In addition, we need to keep track of the best score given that x_i is aligned to a gap, $N(i, j)$, and the best score given that y_j is in an insertion with respect to x, $O(i, j)$. The algorithm is presented below.

$$M(i,j) = \max\{M(i-1,j-1), N(i-1,j), O(i,j-1)\} + s(A_i, B_j),$$
$$N(i,j) = \max\{M(i-1,j-1) + GO, N(i-1,j) + GE\},$$
$$O(i,j) = \max\{M(i-1,j-1) + GO, O(i,j-1) + GE\}$$

where GO is the gap penalty for opening a gap and GE is the penalty for extending an opened gap.

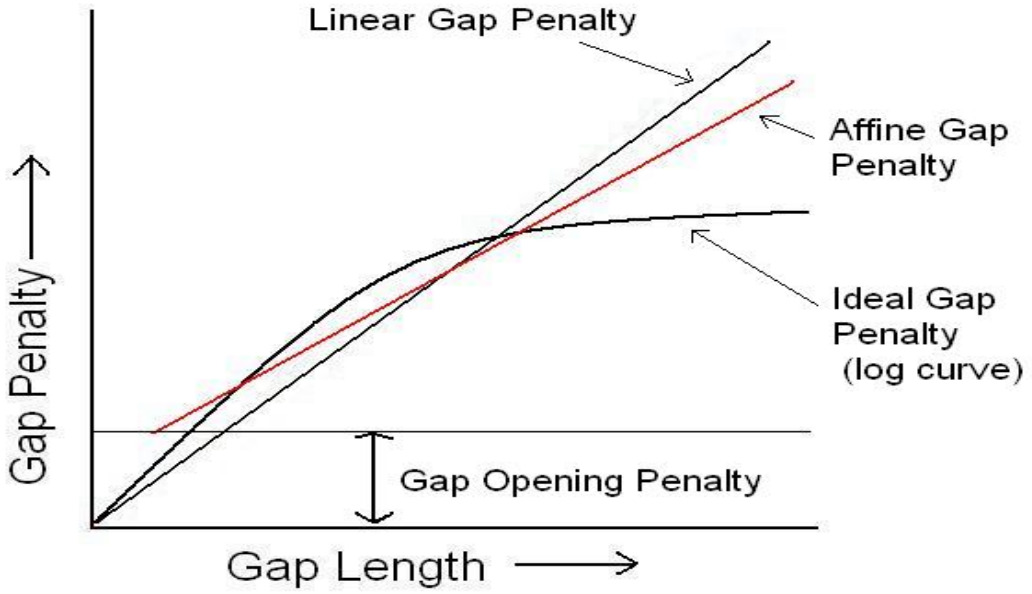

Figure 3: The affine gap penalty curve. At high gap length the affine gap penalty is a better model of the ideal curve then the linear gap penalty model.

Hidden Markov Model for Affine Gap Penalty

Gotoh's algorithm which uses a single affine gap penalty can be represented very elegantly using a Pair-Hidden Markov Model (Pair-HMM) diagram (see Figure 4). This diagram shows a state for each of the 3 matrix that we used in the algorithm, with transition arrows between states. Each transition arrow is associated with a score change, either an increase in the score which represents a match; or a decrease in the score which can be from a mismatch, gap opening, or gap extending. This is a Pair-HMM because it simultaneously outputs two sequences.

One of these HMM diagrams is for a single nucleotide alignment between the 2 sequences. Thus, the alignment between the 2 full sequences will be represented with a string of these HMM models. In relation to the scoring matrix that we have been using (Needleman-Wunsch and Smith-Waterman) we can view this model as having one such HMM model in each cell of the scoring matrix. For each cell we evaluate the alignment score base on the HMM model.

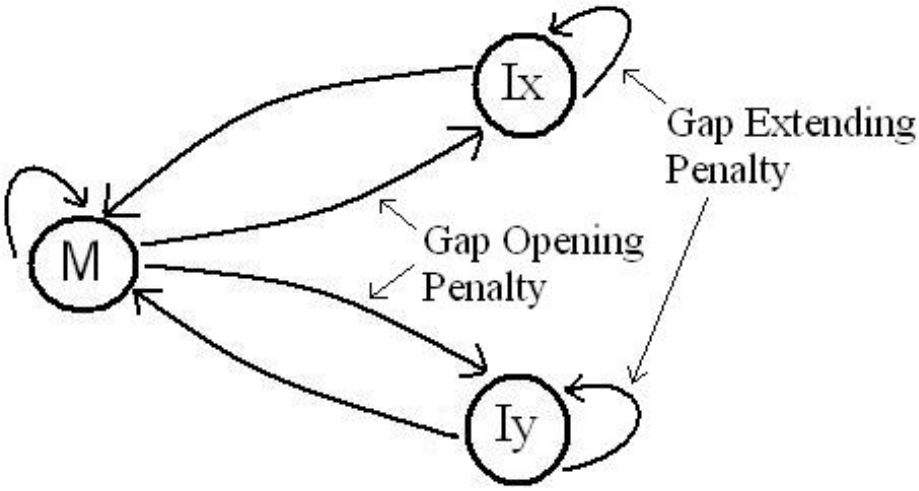

Figure 4: Hidden Markov Model (HMM) of a single affine gap model.

Multiple Affine Gap Model

Gotoh's algorithm, which uses one affine gap penalty, works better then the linear gap penalty model but it is still not a good fit to the idea gap penalty (log curve) at high gap length. One way to better model this is to use multiple affine gap penalties (see Figure 6), though the time complexity of the algorithm will increase rapidly as the number of affine gap increases.

In the HMM we can model this by adding 2 new states for a two affine gap penalty model (see Figure 7). To extend this to a n gap penalty model we will just add $2n$ states. As with the single affine gap penalty model the calculation for each cell will be just the evaluation of this new HMM.

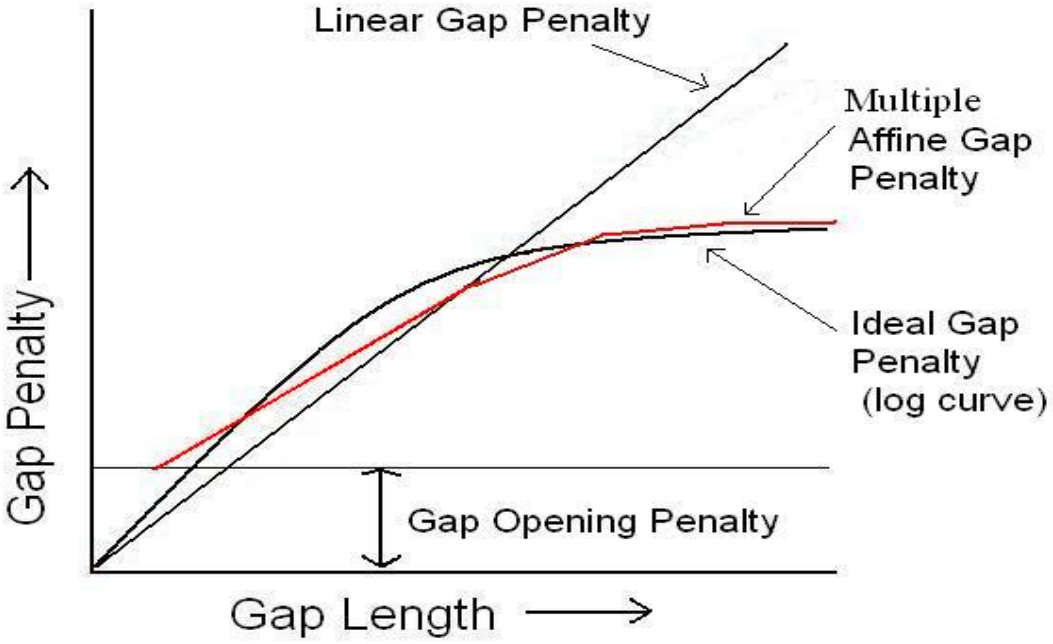

Figure 6: The multiple affine gap penalty curve. This curve is piecewise combination of multiple linear affine gap penalty curves. This curve looks more like the log curve (ideal gap penalty model) as the number of affine gap increases.

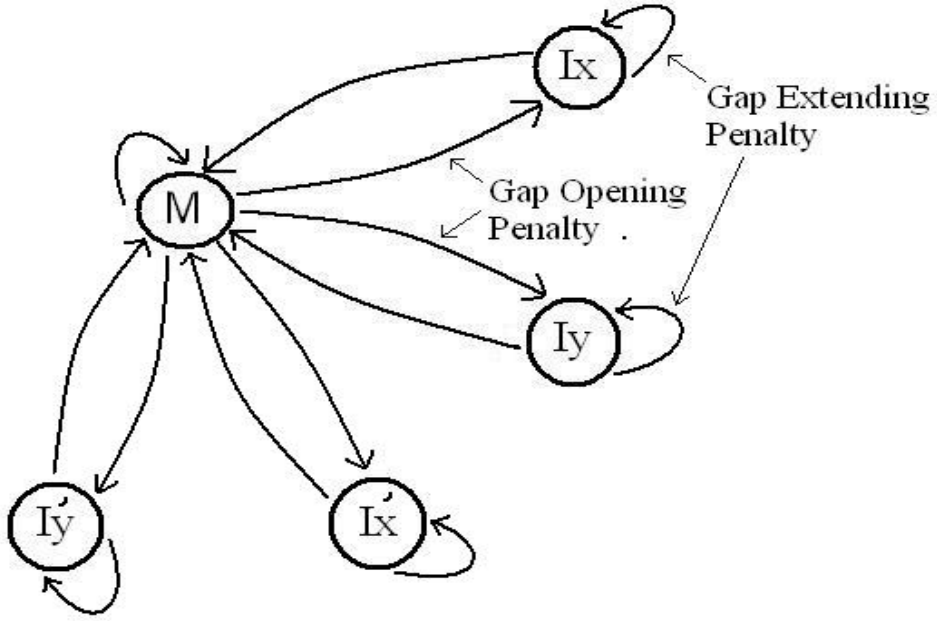

Figure 7: Hidden Markov Model of a two affine gap model.

Multiple Alignment (details in following lectures)

Multiple alignments refer to aligning more than 2 sequences at once. Multiple alignments differ significantly from pairwise alignment in both the scoring method and the alignment approach. Sum of Pairs (SP) scores is one approach to score the alignments and progressive alignment is one approach for building the alignments.

Progressive alignment works by constructing a succession of pairwise alignments. First, the two closest sequences are chosen and aligned using standard pairwise alignment algorithm. Then, a third sequence is chosen and aligned to the resulting alignment of the first two sequences. When doing this we can use sum-of-pairs scoring: the SP scoring function is defined as the sum of all pairwise scores between all pairs of letters in the columns of the multiple alignments:

$$S(a) = \sum_{k<l} S(a^{kl}),$$

where scores s(a,b) comes from a substitution scoring matrix such as a PAM or BLOSUM matrix. This is repeated until all sequences are aligned. A phylogenetic tree similar to Figure 8 can be used as a guide for choosing the order of the pairwise alignment.

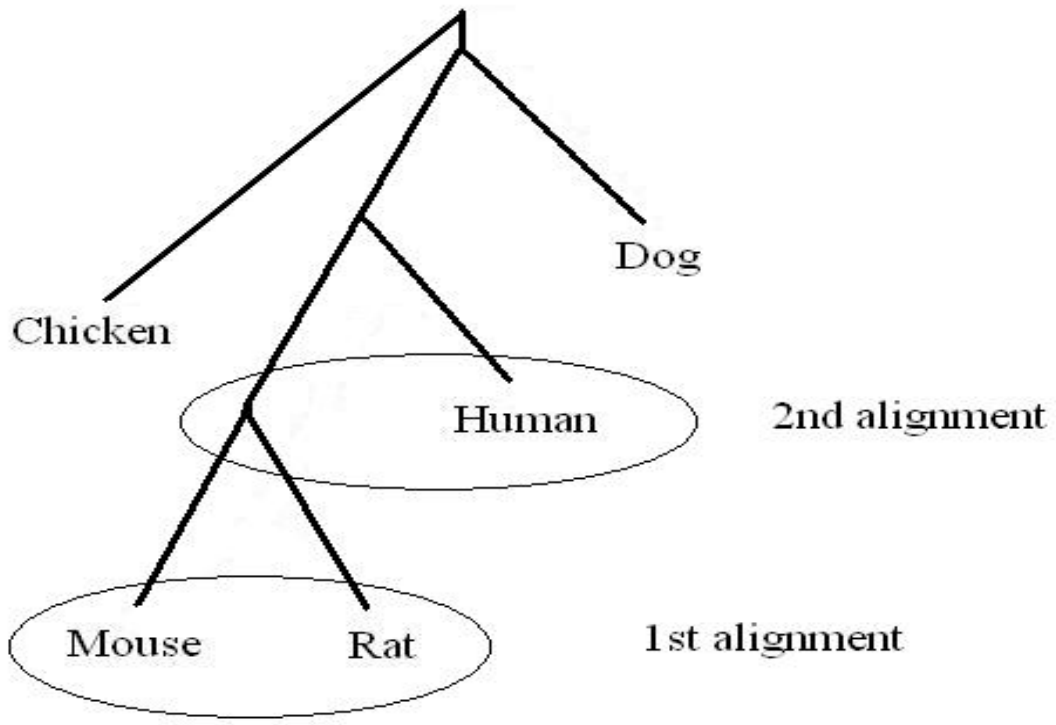

Figure 8: Phylogenetic tree. It can be used to provide progressive alignment algorithms the order of the pairwise alignment.

Reference

1. Gotoh, O. An improved algorithm for matching biological sequences. Journal of Molecular Biology, 162 (3), 705-708