Algorithmic Thinking
Luay Nakhleh

Dynamic Programming and Pairwise Sequence Alignment

- In this Module, we will apply algorithmic thinking to solving a central problem in evolutionary and molecular biology, namely pairwise sequence alignment. In particular, we will reason about the structure of the problem, turn it into an optimization problem, and investigate two attempts at solving it. We conclude with assessing the significance of alignments we compute. This Module illustrates the *dynamic programming* algorithmic technique for solving problems.

Having obtained two molecular sequences X and Y of a gene from two species, say human and mouse, a biologist next tries to assess their similarity in order to make inferences about their evolutionary relationship and, in some cases, about functional information that these two sequences may encode. However, the challenge is that most often X and Y are of different lengths, requiring the biologist to first *align* them. Aligning two sequences amounts to "reverse engineering" the evolutionary process that acted upon the two sequences and modified them so that their characters and their lengths differ. To illustrate the problem, the biologist gives us an example of two sequences $X = ACCT$ and $Y = TACGGT$, as well as their alignment:

$$\begin{array}{cccccccc} X' & = & - & A & C & - & C & T \\ Y' & = & T & A & C & G & G & T \end{array}$$

This alignment indicates (under a simplest possible explanation scenario) that the ancestral sequence of X and Y had 6 letters, and then during evolution, the following events took place: (1) first position in X got deleted; (2) fourth position in X got deleted; and (3) fifth position in Y mutated from C to G. After extensive back-and-forth with the biologist, we finally understand what an alignment is, so that we can formulate the *feasibility* condition, that is, what properties an instance must satisfy in order for it to be an alignment.

As you can imagine, given the same pair of sequences X and Y above, many more possible alignments exist. The biologist is not interested in all of them; instead, she is interested in a *good* alignment. The first question is, naturally: what is a good alignment? Put differently, if I show the biologist two alignments, how does she decide that one is better than the other? After a much more extended back-and-forth with the biologist, we finally understand how to measure the "goodness" of an alignment, and we have the *optimality* criterion, that is, what property makes a feasible instance an optimal one, and hence a desired solution by the biologist.

Having learned from the biologist how to assess the feasibility and determine the optimality of instances, we say goodbye to the biologist, head to Starbucks® (or the less expensive Rice Coffeehouse) with a pencil and paper, and start thinking about the next questions, for which little to no feedback is needed from the biologist: How hard is the problem of finding an optimal alignment? What algorithmic technique works for the problem? How do I assess the significance of the computed alignment? Answering these questions through algorithmic thinking is the goal of Module 4.

1 Strings, Sequences, Substrings, and Subsequences

In our context, and since we are dealing with words over an alphabet, we will use *string* and *sequence* to mean exactly the same thing[1].

Definition 1 *A* string *or a* sequence *over (finite) alphabet* Σ *is a finite ordered list of elements from* Σ.

[1] In general, the difference between the two is that strings are finite and sequences can be infinite. Here, we will not deal with any infinite sequences, and hence the identical definition.

For example, consider the alphabet $\Sigma = \{A, C, T, G\}$. Examples of strings and sequences over this alphabet include A, C, $AAAA$, and $CTATATCA$. Examples of words that are not strings or sequences over this alphabet include $\#$, 01011, $0ABC$, and $AAATCTC!$.

A special string or sequence that is defined over <u>any</u> alphabet, including the empty alphabet, is the *empty string*, often denoted by ε, which is the string that has no letters in it.

We define the *concatenation* of two strings x and y as appending y from the right to x, and denote that by $x + y$. For example, if $x = AC$ and $y = TATA$, then $x + y = ACTATA$. Also, $x + y + y + x = ACTATATATAAC$.

Notice that the order matters. For example, $y + x = TATAAC$ in this case, which is not equal to $x + y$. For any string x, we always have $x + \varepsilon = \varepsilon + x = x$. That is, concatenating the empty string with any string x gives the string x back.

We denote by $|x|$ the length of string x, which is the number of letters in x. For example, if $x = ACAC$, then $|x| = 4$. In particular, $|\varepsilon| = 0$. Further, we always have $|x + y| = |x| + |y|$.

Definition 2 *Consider a string x over alphabet Σ. We say that y is a <u>substring</u> of x, if there exist two strings u and v over alphabet Σ such that $x = u + y + v$. We say that z is a <u>subsequence</u> of x if z can be obtained by removing some letters from x.*

Let us illustrate with examples on string $x = ACGCT$. In this case, AC, ACG and T are all substrings of x (there are others as well). Further, AC, ACC, and AT are all subsequences of x. In general, every substring of x is also a subsequence of x; however, in general, not every subsequence is a substring. The empty string ε is a substring and a subsequence of every string.

We will use the x_i to index the letters in string x, where $0 \leq i < |x|$. For example, for string $x = ACCT$, we have $x_0 = A$ and $x_3 = T$.

2 Global Pairwise Alignment: Feasibility + Optimality

Let X and Y be two sequences over alphabet Σ (for DNA sequences, $\Sigma = \{A, C, T, G\}$ and for protein sequences, Σ is the set of all 20 amino acids). An alignment of X and Y is two sequences X' and Y' over the alphabet $\Sigma \cup \{-\}$, where X' is formed from X by adding only dashes to it, and Y' is formed from Y by adding only dashes to it, such that

1. $|X'| = |Y'|$,
2. there does not exist an i such that $X'[i] = Y'[i] = -$, and
3. X is a subsequence of X', and Y is a subsequence of Y'.

In the introduction above, the two sequences X' and Y' constitute a global pairwise alignment of X and Y. Another possible global alignment is

X'	=	-	-	A	C	-	C	T
Y'	=	T	A	-	C	G	G	T

However, the following is not a global alignment

X'	=	T	A	C	-	C	-
Y'	=	T	A	C	G	G	T

since it violates the third feasibility condition above (X is not a subsequence of X').

As you see above, a number of alignments exist for a given pair of sequences; therefore, we define a *scoring scheme* that is supposed to give higher scores to "better" alignments. Once the scoring scheme is defined, we seek the alignment with the highest score (among all feasible alignments). For DNA, a scoring scheme is given by a 5×5 matrix M, where for $\sigma_1, \sigma_2 \in \{A, C, T, G\}$, M_{σ_1,σ_2} specifies the score for aligning σ_1 in sequence X' with σ_2 in sequence Y', $M_{\sigma_1,-}$ denotes the penalty for aligning σ_1 in sequence X' with a dash in sequence Y', and M_{-,σ_2} denotes the penalty for aligning σ_2 in sequence Y' with a dash in sequence X'. Similarly for RNA or amino acid sequences, where the only difference is the alphabet. Assuming $|X'| = |Y'| = k$, the score of the alignment is

$$\sum_{i=1}^{k} M_{X'_i, Y'_i}. \tag{1}$$

For example, if we define $M_{\sigma_1,\sigma_1} = 5$, $M_{\sigma_1,\sigma_2} = 2$ (for $\sigma_1 \neq \sigma_2$), $M_{\sigma_1,-} = -2$ and $M_{-,\sigma_2} = -4$, the score of the alignment (given in the introduction)

$$\begin{array}{cccccccc} X' & = & - & A & C & - & C & T \\ Y' & = & T & A & C & G & G & T \end{array}$$

is

$$M_{-,T} + M_{A,A} + M_{C,C} + M_{-,G} + M_{C,G} + M_{T,T} = (-4) + 5 + 5 + (-4) + 2 + 5 = 9.$$

Now that we have defined the feasibility conditions of a global alignment, and how to score it, we are in position to define the GLOBAL PAIRWISE ALIGNMENT PROBLEM:

> **Input:** Strings X and Y over alphabet Σ and scoring matrix M of dimensions $(|\Sigma| + 1) \times (|\Sigma| + 1)$.
>
> **Output:** An alignment of X and Y whose score (as defined by M) is maximum among all possible alignments of X and Y.

A dynamic programming (DP) solution. As you will see in Homework 4, the number of possible alignments of two sequences grows very fast as the sequence lengths increase. Therefore, a brute-force algorithm that iterates explicitly over every global alignment, scores it, and then returns an optimal one, is infeasible except for very short sequences.

In Module 4 you will develop a DP algorithm for solving the GLOBAL PAIRWISE ALIGNMENT PROBLEM. To obtain an optimal global alignment of two sequences X and Y using a scoring matrix M, the algorithm builds an $(|X| + 1) \times (|Y| + 1)$ DP table S (which is a matrix whose height is $|X| + 1$ and whose width is $|Y| + 1$), such that entry $S[i, j]$, for $0 \leq i \leq |X|$ and $0 \leq j \leq |Y|$, contains the optimal score of globally aligning the first i letters of X with the first j letters of Y. Notice that entry $S[|X|, |Y|]$ would then be the score of an optimal global alignment of X and Y.

Consider the two sequences $X = AC$ and $Y = TAG$, and the scoring matrix M given by: $M_{\sigma_1,\sigma_1} = 5$, $M_{\sigma_1,\sigma_2} = 2$ (for $\sigma_1 \neq \sigma_2$), $M_{\sigma_1,-} = -2$ and $M_{-,\sigma_2} = -4$. In this case, the DP table S is the following:

0	-4	-8	-12
-2	2	1	-3
-4	0	4	3

For example, $S[1, 2] = 1$, which means that the score of a global alignment of the A (the first letter from X) and TA (the first two letters from Y) is 1. Further, the score of the global alignment of X and Y is 3. Obtaining the optimal global alignment itself can be done via *traceback*, which basically means walking back in the matrix from entry $S[|X|, |Y|]$ to entry $S[0, 0]$ along the path that has score $S[|X|, |Y|]$, and while doing so, creating the alignment itself.

3 Local Pairwise Alignment

It has long been known in biology that certain regions of the sequences may be conserved during evolution (as indicated by high similarity of the subsequences in those regions) and other regions may be evolving too fast (as indicated by the lack, or very low degree, of similarity of the sequences in those regions). For example, a gene in the human genome has certain regions called *exons* and others that are called *introns*. The exons are the parts that get transcribed into RNA and eventually translated into amino acids (the protein). Given their important functional role, exons in general are much more conserved across specie than introns. In these cases it makes sense to seek an alignment of two sequences that identifies a local region within the original two sequences that align well and ignores the other parts of the sequences. Let us illustrate with sequences $X = ACC$ and $Y = TTTACACGG$ and the scoring matrix M given by:

- $M_{\sigma,\sigma} = 10$ for every $\sigma \in \Sigma$.
- $M_{\sigma_1,\sigma_2} = 2$ for every $\sigma_1, \sigma_2 \in \Sigma$ and $\sigma_1 \neq \sigma_2$.
- $M_{\sigma_1,-} = M_{-,\sigma_2} = -4$ for every $\sigma_1, \sigma_2 \in \Sigma$.

An optimal global alignment in this case is

$$\begin{array}{cccccccccc} X' & = & - & - & - & A & C & - & C & - & - \\ Y' & = & T & T & T & A & C & A & C & G & G \end{array}$$

and its score is 6. Notice that in this case, ACC from X aligns well with the substring $ACAC$ from Y, however, the other parts (the first three letters and the last two letters in the alignment) contribute negative scores that masks the high similarity that we would have found had we trimmed these two regions. Indeed, in this case, it would be more informative (for the biologist) to seek a solution to the LOCAL PAIRWISE ALIGNMENT PROBLEM:

> **Input:** Strings X and Y over alphabet Σ and scoring matrix M of dimensions $(|\Sigma| + 1) \times (|\Sigma| + 1)$.
>
> **Output:** Global pairwise alignment X' and Y' of substrings of of X and Y whose score (as defined by Equation (1) above) is maximum among all global alignments of all pairs of substrings from X and Y, respectively.

Let us illustrate this problem on the two strings $X = ACC$ and $Y = TTTACACGG$ in the example above. Based on the problem definition, we can try every substring u of X and v of Y, align (globally) u and v, and finally return one that has the maximum global alignment score. Here are some examples:

1. $u = \varepsilon$ and $v = TTT$. In this case, an optimal global alignment of u and v is $X' = ---$ and $Y' = TTT$, which has a score of -12.
2. $u = ACC$ and $v = TTTA$. In this case, an optimal global alignment of u and v is $X' = ACC-$ and $Y' = TTTA$, which has a score of 2.
3. $u = AC$ and $v = AC$. In this case, an optimal global alignment of u and v is $X' = AC$ and $Y' = AC$, which has a score of 20.
4. $u = ACC$ and $v = ACA$. In this case, an optimal global alignment of u and v is $X' = ACC$ and $Y' = ACA$, which has a score of 22.
5. $u = ACC$ and $v = ACAC$. In this case, an optimal global alignment of u and v is $X' = AC - C$ and $Y' = ACAC$, which has a score of 26.

Obviously, there are many more possibilities than the five we listed here. However, it turns out (and you can verify) that the fifth one listed ($u = ACC$ and $v = ACAC$) results in a solution to the LOCAL PAIRWISE ALIGNMENT PROBLEM. In this case, the solution is

$$\begin{array}{ccccc} X' & = & A & C & - & C \\ Y' & = & A & C & A & C \end{array}$$

with a score of 26.

Clearly, trying every substring of X and every substring of Y and aligning these globally is an infeasible solution for the LOCAL PAIRWISE ALIGNMENT PROBLEM except for cases with very short strings. In Module 4, you will see how a small modification to the dynamic programming algorithm for the GLOBAL PAIRWISE ALIGNMENT PROBLEM solves this problem.