8.2 Multiple alignment

A multiple sequence alignment (MSA) is a sequence alignment of three or more biological sequences (amino acid-, DNA-, or RNA-sequences). In many cases, the sequences under consideration are assumed to have an

Sequence	Species	Common name	Acronym
$S^1 = AGTAATGG$	*Solea vulgaris*	Common sole	SoV
$S^2 = TTTAATGA$	*Solea lascaris*	Sand sole	SoL
$S^3 = AAGAAATGG$	*Monochirus hispidus*	Whiskered sole	MoH
$S^4 = ATAAAATGG$	*Microchirus ocellatus*	Foureyed sole	MiO

Table 8.1: Subsequences from a mtDNA gene (coding for 16S rRNA) of four Mediterranean sole species.

evolutionary relationship by which they share a lineage and are descended from a common ancestor. From the resulting multiple sequence alignment, sequence homology can be inferred and phylogenetic analysis can be conducted to assess the sequences' shared evolutionary origins.

Again, we focus on global alignments.

Definition 8.2.1 A global *multiple alignment* of m strings $S^1, \ldots, S^m$ (on the alphabet Σ) is an $(m \times N)$ matrix A so that:

1. $A(i,j) \in \Sigma \cup \{-\}$, where $-$ is a special gap symbol not occurring in Σ.
2. After removal of all gap symbols the k-th row of A equals S^k.
3. No column of A consists solely of gap symbols.

This definition imposes restrictions on the length N of A. Condition (2) implies that $\max_{1 \le k \le m}\{n_k\} \le N$, while condition (3) has $N \le \sum_{k=1}^{m} n_k$ as a consequence.

As an example, we consider the following small part of a multiple alignment of the DNA sequences of the 16S rRNA genes of the mitochondrial genomes of four Mediterranean sole species; see Table 8.1 and Tinti et al. [310] for more details.

$$A^{sole} = \begin{pmatrix} A & - & G & T & A & A & T & G & G \\ T & T & - & T & A & A & T & G & A \\ A & A & G & A & A & A & T & G & G \\ A & T & A & A & A & A & T & G & G \end{pmatrix}$$

To assess the quality of a multiple alignment, we need to *score* the alignment. This score is usually based on pairwise alignment scoring schemes like Levenshtein costs (a dissimilarity scoring scheme used e.g. for DNA sequences) or PAM and BLOSUM substitution matrices (a similarity scoring scheme for amino acid sequences). Here we confine ourselves to the so-called sum-of-pairs score. In what follows, let $\pi_{i,j}(A)$ be the projection to the i-th and j-th row of the alignment A. For example,

$$\pi_{1,2}(A^{sole}) = \begin{pmatrix} A & - & G & T & A & A & T & G & G \\ T & T & - & T & A & A & T & G & A \end{pmatrix}$$

Definition 8.2.2 Let A be a multiple alignment of the strings $S^1, \ldots, S^m$. Given a pairwise alignment scoring scheme *score*, the *sum-of-pairs score* of A is defined by

$$score_{SP}(A) = \sum_{1 \le i < j \le m} score(\pi_{i,j}(A))$$

where $score(-,-) = 0$.

In our example, we use the Levenshtein cost function δ as a pairwise alignment scoring scheme and obtain $\delta(\pi_{1,2}(A^{sole})) = 4$, $\delta(\pi_{1,3}(A^{sole})) = 2$, $\delta(\pi_{1,4}(A^{sole})) = 3$, $\delta(\pi_{2,3}(A^{sole})) = 5$, $\delta(\pi_{2,4}(A^{sole})) = 4$, and $\delta(\pi_{3,4}(A^{sole})) = 2$. Thus, the sum-of-pairs score of A^{sole} is 20.

Exercise 8.2.3 Compute the sum-of-pairs score of the following multiple alignment of the amino acid sequences $S^1 = \texttt{NFLS}$, $S^2 = \texttt{NFS}$, $S^3 = \texttt{NKYLS}$, and $S^4 = \texttt{NYLS}$. Use the PAM250 matrix and score each indel with -8.

$$A^{prot} = \begin{pmatrix} \texttt{N} & \texttt{-} & \texttt{F} & \texttt{L} & \texttt{S} \\ \texttt{N} & \texttt{-} & \texttt{F} & \texttt{-} & \texttt{S} \\ \texttt{N} & \texttt{K} & \texttt{Y} & \texttt{L} & \texttt{S} \\ \texttt{N} & \texttt{-} & \texttt{Y} & \texttt{L} & \texttt{S} \end{pmatrix}$$

Definition 8.2.4 Given strings $S^1, \ldots, S^m$ and a cost function δ (similarity function σ, respectively), the global *multiple alignment problem* is to compute a global multiple alignment of $S^1, \ldots, S^m$ that has minimum (maximum, respectively) *sum-of-pairs score* for δ (σ, respectively).

In what follows, we only use cost functions as scoring schemes, and the notation δ_{SP} and δ (instead of $score_{SP}$ and *score*) will emphasize this.

It is possible to generalize the dynamic programming recurrences for pairwise alignments to multiple alignments. For $m = 3$ sequences and $i_1 \neq 0$, $i_2 \neq 0$, $i_3 \neq 0$, they have the following form:

$$D(i_1,i_2,i_3) = \min \begin{cases} D(i_1-1,i_2-1,i_3-1) + \delta_{SP}(S^1[i_1], S^2[i_2], S^3[i_3]) \\ \\ D(i_1-1,i_2-1,i_3) + \delta_{SP}(S^1[i_1], S^2[i_2], -) \\ D(i_1,i_2-1,i_3-1) + \delta_{SP}(-, S^2[i_2], S^3[i_3]) \\ D(i_1-1,i_2,i_3-1) + \delta_{SP}(S^1[i_1], -, S^3[i_3]) \\ \\ D(i_1-1,i_2,i_3) + \delta_{SP}(S^1[i_1], -, -) \\ D(i_1,i_2-1,i_3) + \delta_{SP}(-, S^2[i_2], -) \\ D(i_1,i_2,i_3-1) + \delta_{SP}(-, -, S^3[i_3]) \end{cases}$$

It is possible to simplify the notation by introducing Δ_k, which is 0 or 1, and defining

$$\Delta_k \circ c = \begin{cases} c & \text{if } \Delta_k = 1 \\ - & \text{if } \Delta_k = 0 \end{cases}$$

Then, $D(i_1,i_2,i_3)$ can be computed by

$$\min_{\Delta_1+\Delta_2+\Delta_3>0} D(i_1-\Delta_1, i_2-\Delta_2, i_3-\Delta_3) + \delta_{SP}(\Delta_1 \circ S^1[i_1], \Delta_2 \circ S^2[i_2], \Delta_3 \circ S^3[i_3])$$

For an arbitrary number m of strings, the recurrence relation is

$$D(i_1,\ldots,i_m) = \min_{\Delta_1+\cdots+\Delta_m>0} D(i_1-\Delta_1,\ldots,i_m-\Delta_m) + \delta_{SP}(\Delta_1 \circ S^1[i_1], \ldots, \Delta_m \circ S^m[i_m])$$

Although the multiple alignment problem can be solved exactly via dynamic programming and traceback, this solution is not practical. This can be seen by the following complexity analysis. The algorithm must compute an $n_1 \times n_2 \times \cdots \times n_m$ matrix, so the space complexity is $O(n_1 n_2 \cdots n_m)$, which is $O(n^m)$ if n denotes the maximum sequence length. Each of the $O(n^m)$ entries in the matrix is computed by taking a maximum of $2^m - 1$ values. Moreover, the computation of each of the $2^m - 1$ values requires $O(m^2)$ time (one has to compute the sum-of-pairs score of an m-dimensional vector). So the overall time complexity is $O(m^2 \cdot 2^m \cdot n^m)$.

Even worse, the multiple alignment problem has been proven to be NP-complete [171, 325]. There are three possible ways out of this trap:

1. Try to find methods that prune the search space without sacrificing an optimal solution. In this way, larger instances of the problem can be exactly solved in reasonable time.

2. Try to devise an efficient approximation algorithm that computes an approximate solution that is optimal up to a small constant factor.

3. Try to develop heuristics that find reasonably good solutions reasonably fast, without guaranteeing optimality.

In Sections 8.2.1, 8.2.2, and 8.2.3 we describe a representative of each of these methods. It will be convenient to use the following notation:

$$dist_\delta(S^1, S^2) = \min\{\delta(A) \mid A \text{ is an alignment of } S^1 \text{ and } S^2\}$$

That is, given a cost function δ, $dist_\delta(S^1, S^2)$ denotes the cost of an optimal alignment between S^1 and S^2 (the "distance" between S^1 and S^2 w.r.t. δ).

8.2.1 Pruning the search space

In the following, let A^{opt} be an optimal alignment of the strings $S^1, \ldots, S^m$, and let A^{heur} be an alignment obtained by a fast heuristic algorithm (for example by one of the alignment methods given in subsequent sections).

We have

$$\begin{aligned}\delta(A^{heur}) &\geq \delta(A^{opt}) \\ &= \sum_{k<l} \delta(\pi_{k,l}(A^{opt})) \\ &= \delta(\pi_{p,q}(A^{opt})) + \sum_{k<l,(k,l)\neq(p,q)} \delta(\pi_{k,l}(A^{opt})) \\ &\geq \delta(\pi_{p,q}(A^{opt})) + \sum_{k<l,(k,l)\neq(p,q)} dist_\delta(S^k, S^l)\end{aligned}$$

for every pair (p,q), $1 \leq p < q \leq m$.

Consequently,

$$U_{p,q} = \delta(A^{heur}) - \sum_{k<l,(k,l)\neq(p,q)} dist_\delta(S^k, S^l)$$

is an upper bound for $\delta(\pi_{p,q}(A^{opt}))$, i.e., $\delta(\pi_{p,q}(A^{opt})) \leq U_{p,q}$.

For every pair (p,q), define for $1 \leq i \leq n_p$ and $1 \leq j \leq n_q$

$$B_{p,q}(i,j) = dist_\delta(S^p[1..i], S^q[1..j]) + dist_\delta(S^p[i+1..n_p], S^q[j+1..n_q])$$

Thus, $B_{p,q}(i,j)$ is the minimum cost of all paths in the alignment graph of S^p and S^q that pass through the node (i,j). Note that $B_{p,q}(i,j)$ can be computed in $O(n_p n_q)$ time for all $1 \leq i \leq n_p$ and $1 \leq j \leq n_q$ by computing the dynamic programming matrix $D_{p,q}$, which computes the distance of S^p and S^q in the standard "forward" direction, and the dynamic programming matrix $D^{rev}_{p,q}$, which computes the distance of S^p and S^q in the "backward" direction; see Section 8.1.2.

The method of Carillo and Lipman [51] uses the notions defined above to restrict the search space as follows: The m-dimensional dynamic programming algorithm computes values only for those nodes $(i_1, i_2, \ldots, i_m)$ in the m-dimensional alignment graph of $S^1, \ldots, S^m$ that satisfy $B_{p,q}(i_p, i_q) \leq U_{p,q}$ for all pairs (p,q); the corresponding entries in the m-dimensional dynamic programming matrix are marked in Figure 8.12.

Why can the other nodes safely be skipped? To understand this, suppose that the path in the alignment graph that corresponds to a multiple alignment A of $S^1, \ldots, S^m$ passes through a node $(i_1, i_2, \ldots, i_m)$ for which there is a pair (k,l) so that $B_{k,l}(i_k, i_l) > U_{k,l}$. It then follows from $\delta(\pi_{k,l}(A)) \geq B_{k,l}(i_k, i_l)$ that $\delta(\pi_{k,l}(A)) > U_{k,l}$. According to the preceding discussion, this means that the alignment A cannot be optimal.

One still has to find a path of minimum cost from node $(0, \ldots, 0)$ to node $(n_1, \ldots, n_m)$ in the reduced m-dimensional alignment graph. This can e.g. be done by using Dijkstra's algorithm [61, 80] (a standard shortest path algorithm) or the so-called A^*-algorithm [200].

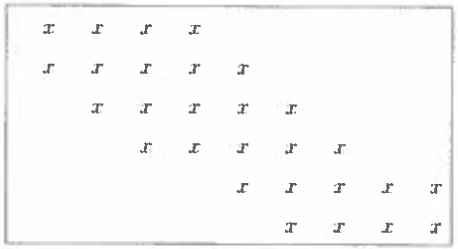

Figure 8.12: A 2-dimensional projection of the restricted m-dimensional dynamic programming matrix.

Carillo and Lipman's method is implemented in the program MSA. According to [205], MSA "is able to align in reasonable time as many as eight sequences the length of an average protein." Another implementation by Gupta et al. [137] uses a heuristic to further reduce the search space. They added an additional parameter $\epsilon_{p,q}$ to the program, and a node in the alignment graph is considered irrelevant when $B_{p,q}(i_p, i_q) \leq U_{p,q} - \epsilon_{p,q}$. Note that this heuristic does not guarantee an optimal alignment.

8.2.2 A 2-approximation algorithm

If the cost function δ is a distance (i.e., it satisfies the metric axioms), then the so-called *center star method* [138, 139] yields a simple 2-approximation algorithm for the multiple alignment problem as we shall now see. Given strings $S^1, \ldots, S^m$, a *center string* S^c is one of the strings $S^1, \ldots, S^m$ that minimizes

$$\sum_{i=1}^{m} dist_\delta(S^c, S^i)$$

That is to say, S^c is a string whose distance to all the rest is minimum; see Figure 8.13. Then, the center star alignment A^c is constructed as a combination of the pairwise optimal alignments of the center string with the other strings.

To determine the center string S^c in our *sole* example, we compute all pairwise edit distances (i.e., we use Levenshtein costs).

$dist_\delta$	S^1	S^2	S^3	S^4	$\sum$
S^1	0	3	2	3	8
S^2	3	0	5	4	12
S^3	2	5	0	2	9
S^4	3	4	2	0	9

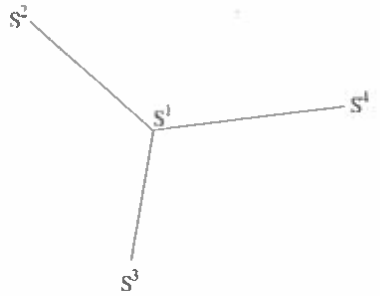

Figure 8.13: S^1 is the center string of $S^1, \ldots, S^4$.

Taking the minimum of the sums of each row, we infer that $S^c = S^1$ is the center string. Optimal pairwise alignments of S^1 with the other sequences are:

$$A^{1,2} = \begin{pmatrix} A & G & T & A & A & T & G & G \\ T & T & T & A & A & T & G & A \end{pmatrix}$$

$$A^{1,3} = \begin{pmatrix} - & A & G & T & A & A & T & G & G \\ A & A & G & A & A & A & T & G & G \end{pmatrix}$$

$$A^{1,4} = \begin{pmatrix} A & - & G & T & A & A & T & G & G \\ A & T & A & A & A & A & T & G & G \end{pmatrix}$$

Then, the center star alignment A^c is constructed as a combination of the pairwise optimal alignments of the center string with the other strings. First, by combining $A^{1,2}$ and $A^{1,3}$, we obtain the multiple alignment

$$A^{1,2,3} = \begin{pmatrix} - & A & G & T & A & A & T & G & G \\ - & T & T & T & A & A & T & G & A \\ A & A & G & A & A & A & T & G & G \end{pmatrix}$$

Second, the combination of the multiple alignment $A^{1,2,3}$ with the pairwise alignment $A^{1,4}$ yields the center star alignment A^c. Note that entire columns must be shifted to incorporate the gap of the alignment $A^{1,4}$.

$$A^c = \begin{pmatrix} - & A & - & G & T & A & A & T & G & G \\ - & T & - & T & T & A & A & T & G & A \\ A & A & - & G & A & A & A & T & G & G \\ - & A & T & A & A & A & A & T & G & G \end{pmatrix}$$

The sum-of-pairs score of the center star alignment is $\delta(A^c) = 21$. This is not optimal because the sum-of-pairs score of the alignment A^{sole} is $\delta(A^{sole}) = 20$.

This example shows that the center star method does—in general—not yield an optimal alignment. However, Gusfield [138, 139] showed that

$$\delta_{SP}(A^c) \leq \frac{2(m-1)}{m}\, \delta_{SP}(A^{opt})$$

That is, the center star method is a $(2 - \frac{2}{m})$-approximation algorithm for the multiple sequence alignment problem. In particular, even for large m, the sum-of-pairs score of the center star alignment A^c is at most twice the sum-of-pairs score of an optimal alignment.

In order to derive this guaranteed error bound, we use the next lemma; cf. [299].

Lemma 8.2.5 *Let S^c be a center string and A^{opt} be an optimal alignment of the strings $S^1, \ldots, S^m$. Then*

$$\frac{m}{2} \sum_{i=1}^{m} dist_\delta(S^i, S^c) \leq \delta_{SP}(A^{opt})$$

Proof

$$\begin{aligned}
\frac{m}{2} \sum_{i=1}^{m} dist_\delta(S^i, S^c) &= \frac{1}{2} \underbrace{\left(\sum_{i=1}^{m} dist_\delta(S^1, S^c) + \cdots + \sum_{i=1}^{m} dist_\delta(S^i, S^c) \right)}_{m\text{-times}} \\
&\leq \frac{1}{2} \left(\sum_{i=1}^{m} dist_\delta(S^i, S^1) + \cdots + \sum_{i=1}^{m} dist_\delta(S^i, S^m) \right) \\
&= \frac{1}{2} \sum_{j=1}^{m} \sum_{i=1}^{m} dist_\delta(S^i, S^j) \\
&= \frac{1}{2} \sum_{i=1}^{m} \sum_{j=1}^{m} dist_\delta(S^i, S^j) \\
&\leq \frac{1}{2} \sum_{i=1}^{m} \sum_{j \neq i} \delta(\pi_{i,j}(A^{opt})) \\
&= \sum_{1 \leq i < j \leq m} \delta(\pi_{i,j}(A^{opt})) \\
&= \delta_{SP}(A^{opt})
\end{aligned}$$

□

Now we are in a position to prove the error bound of the center star method.

Theorem 8.2.6 *Let A^{opt} be an optimal alignment of $S^1, \ldots, S^m$. Furthermore, let S^c be a center string and A^c the corresponding alignment. Then*

$$\delta_{SP}(A^c) \leq \frac{2(m-1)}{m}\,\delta_{SP}(A^{opt})$$

Proof

$$\begin{aligned}
\delta_{SP}(A^c) &= \sum_{1\leq i<j\leq m} \delta(\pi_{i,j}(A^c)) \\
&= \frac{1}{2}\sum_{i=1}^{m}\sum_{j\neq i} \delta(\pi_{i,j}(A^c)) \\
&\leq \frac{1}{2}\sum_{i=1}^{m}\sum_{j\neq i} (\delta(\pi_{i,c}(A^c)) + \delta(\pi_{c,j}(A^c))) && \text{(triangle inequality)} \\
&= \frac{1}{2}\sum_{i=1}^{m}\sum_{j\neq i} (\delta(\pi_{i,c}(A^c)) + \delta(\pi_{j,c}(A^c))) && \text{(symmetry)} \\
&= \frac{1}{2}\sum_{i=1}^{m}\sum_{j\neq i} \delta(\pi_{i,c}(A^c)) + \frac{1}{2}\sum_{i=1}^{m}\sum_{j\neq i} \delta(\pi_{j,c}(A^c)) \\
&= \frac{1}{2}\sum_{i=1}^{m}\sum_{j\neq i} \delta(\pi_{i,c}(A^c)) + \frac{1}{2}\sum_{j=1}^{m}\sum_{i\neq j} \delta(\pi_{j,c}(A^c)) \\
&= \frac{1}{2}(m-1)\sum_{i=1}^{m} \delta(\pi_{i,c}(A^c)) + \frac{1}{2}(m-1)\sum_{j=1}^{m} \delta(\pi_{j,c}(A^c)) \\
&= (m-1)\sum_{i=1}^{m} \delta(\pi_{i,c}(A^c)) \\
&= (m-1)\sum_{i=1}^{m} dist_\delta(S^i, S^c) && \text{(center star align.)} \\
&\leq \frac{2(m-1)}{m}\,\delta_{SP}(A^{opt}) && \text{(Lemma 8.2.5)}
\end{aligned}$$

□

In the complexity analysis, we assume for ease of presentation that the sequences are roughly of the same length n. The computation of all $\binom{m}{2}$ pairwise distances takes $O(m^2n^2)$ time and $O(m^2+n)$ space (provided all $O(m^2)$ pairwise distances are stored). Once the center string S^c is identified, the $m-1$ pairwise alignments of S^c with the other strings can be computed in $O(mn^2)$ time and $O(mn)$ space by Hirschberg's algorithm (cf. Section 8.1.2) from the $m-1$ pairwise distances of S^c with the other strings. The combination of these pairwise alignments into a multiple alignment can be done in time proportional to the size of the multiple alignment,

which is $O(mn)$. To sum up, the overall time complexity is $O(m^2n^2)$, while the overall space complexity is $O(mn)$.