3 Multiple Sequence Alignment (script by Huson & Nieselt)

A *multiple sequence alignment* (MSA) is simply an alignment of more than two sequences, like this:

A small fragment of a multiple sequence alignment of hemoglobin protein sequences and homologues. The MSA shows conserved residues, conserved regions and more sophisticated patterns. Multiple alignments are helpful for protein structure prediction.

### 3.1 Why multiple sequence alignments?

MSAs are of interest because they can be used to:

- detect homologous residues,
- infer the evolutionary history of the sequences
- determine the members of protein families,
- compute profiles,
- predict the secondary structures of proteins,
- compute motifs (e.g. transcription-factor binding sites),
- and more...

### 3.2 Definition of an MSA

Suppose we are given \( r \) sequences \( A_i, i = 1, \ldots, r \) over an alphabet \( \Sigma \):

\[
A := \begin{cases}
A_1 = (a_{11}, a_{12}, \ldots, a_{1n_1}) \\
A_2 = (a_{21}, a_{22}, \ldots, a_{2n_2}) \\
\vdots \\
A_r = (a_{r1}, a_{r2}, \ldots, a_{rn_r})
\end{cases}
\]

A *multiple sequence alignment* (MSA) of \( A \) is obtained by inserting gaps (’-’) into the original sequences such that all resulting sequences \( A_i^* \) have equal length \( L \geq \max\{n_i | i = 1, \ldots, r\} \), we can get back the sequence \( A_i \) by removing all gaps from \( A_i^* \), and no column consists of gaps only:

\[
A^* := \begin{cases}
A_1^* = (a_{11}^*, a_{12}^*, \ldots, a_{1L}^*) \\
A_2^* = (a_{21}^*, a_{22}^*, \ldots, a_{2L}^*) \\
\vdots \\
A_r^* = (a_{r1}^*, a_{r2}^*, \ldots, a_{rL}^*)
\end{cases}
\]
3.3 The alignment graph

In the following, we will consider a novel approach to computing an MSA based on “integer linear programming”.

Suppose we are given two sequences $a_1 = \text{AGCT}$ and $a_2 = \text{AGT}$.

The complete alignment graph is the following bipartite graph $G = (V, E)$, with node set $V$ and edge set $E$:

```
A   G   C   T
\---\--\--\--\--
A   G   T   \--\--
\---\--\--\--\--
```

Each edge $e = (u, v)$ has a weight $\omega(e) = s(u, v)$, the score for placing $v$ under $u$.

An alignment graph is any subgraph of the complete alignment graph.

3.3.1 The trace of an alignment

Given an alignment such as:

```
A   G   C   T
A   G   -   T
```

we say that an edge in the alignment graph is realized, if the corresponding positions are aligned:

```
A   G   C   T
\---\--\--\--\--
A   G   T   \--\--
\---\--\--\--\--
```

The set of realized edges is called the trace of the alignment. An arbitrary subset $T \subseteq E$ of edges is called a trace, if there exists some alignment that realizes precisely the edges in $E$.

Similarly, we define the (complete) alignment graph and trace for multiple alignments. For $r$ sequences, the resulting graph will be $r$-partite.

3.3.2 Maximum-weight trace problem

**Problem 3.3.1 (Maximum-Weight Trace Problem)** Given sequences $A$ and a corresponding alignment graph $G = (V, E)$ with edge weights $\omega$. The maximum-weight trace problem is to find a trace $T \subseteq E$ of maximum weight.

3.3.3 Characterization of traces

We have seen that an alignment can be described by a trace in the complete alignment graph $G = (V, E)$.

**Question**: Is every subset $T \subseteq E$ the trace of some alignment?

The answer is clearly **no**.
Our goal is to characterize all legal traces.

Here are two examples:

- **trace**
  - $\begin{array}{cccc}
  A & G & C & T \\
  A & G & T & \\
  A & C & T & \\
  \end{array}$
  - $\begin{array}{cccc}
  A & G & C & T \\
  A & G & T & \\
  A & C & T & \\
  \end{array}$

  \[ \rightarrow \]

- **alignment**
  - $\begin{array}{cccc}
  A & G & C & T \\
  A & - & C & T \\
  \end{array}$

  \[ \rightarrow \]

- $\begin{array}{cccc}
  A & G & C & T \\
  A & - & G & T \\
  A & C & T & - \\
  \end{array}$

\[ \rightarrow \]

**not ok.**

### 3.3.4 Partial orders

A binary relation $\leq$ is a **partial order**, if it is

1. reflexive, i.e., $a \leq a$,
2. antisymmetric, i.e., $a \leq b$ and $b \leq a$ implies $a = b$, and
3. transitive, i.e., $a \leq b$ and $b \leq c$ implies $a \leq c$.

A binary relation $<$ is a **strict partial order**, if it is

1. irreflexive, i.e., $a \not< a$, and
2. transitive, i.e., $a < b$ and $b < c$ implies $a < c$.

Given a binary relation $<$, the **transitive closure** of $<$ is a binary relation $\lt^*$ such that $x \lt^* x'$ if there exists a sequence of elements $x = x_1, x_2, \ldots, x_k = x'$ with $x_1 \lt x_2 \lt \ldots x_k$.

### 3.3.5 The extended alignment graph

We define a partial order $\prec$ on the cells of the matrix $A = \{a_{ij}\}$ by writing $a_{ij} \prec a_{ij'}$, if $j' = j + 1$, and indicate the pairs $(a_{ij}, a_{i,j+1})$ by a set $H$ of directed edges in the alignment graph:
Let \( \prec^* \) denote the transitive closure of \( \prec \), i.e. we write \( a_{ij} \prec^* a_{ij'} \), if \( j < j' \).

Consider two sets of nodes \( X \subseteq V \) and \( Y \subseteq V \). We define

\[
X \triangleleft Y,
\]

if and only if

\[
\exists x \in X, \exists y \in Y : x \prec y.
\]

We define \( \triangleleft^* \) to be the transitive closure of \( \triangleleft \), that is, we write \( X \triangleleft^* Y \), if for one of the sequences \( a_p \in A \) we have that \( X \) contains a node representing a position \( a_{pj} \) in \( a_p \) and \( Y \) contains a node representing another position \( a_{pk} \) in \( a_p \), with \( j < k \).

In other words, we write

\[
X \triangleleft^* Y,
\]

if and only if

\[
\exists x \in X, \exists y \in Y : x \triangleleft^* y.
\]

Consider the two examples again and define sets \( X_1, X_2, \ldots \) via the two given traces:

In (a), \( X_1 \triangleleft^* X_2 \triangleleft^* X_3 \triangleleft^* X_4 \) and we have a strict partial order.

In (b), we have \( X_1 \triangleleft^* X_2, X_3, X_4; X_2 \triangleleft^* X_3, X_4; \) and \( X_3 \triangleleft^* X_2, X_4 \). In this case, the condition for a partial or for a strict partial order is not fulfilled.

### 3.3.6 Characterization of traces

**Theorem 3.3.2 (Trace characterization)** Suppose we are given a set of sequences \( A \). Let \( G = (V, E, H) \) be an extended alignment graph for \( A \). A subset \( T \subseteq E \) of edges is a trace, if and only if \( \triangleleft^* \) is a strict partial order the set of all connected components of \( G' = (V, T) \).
(Recall that a connected component of a graph is a maximal set of nodes \( U \subseteq V \) such that any two nodes \( v, u \in U \) are connected by a path of edges in the graph.)

### 3.3.7 Proof

We first prove \( \Rightarrow \): Assume that \( T \) is the trace of an alignment \( A^* \) of \( A \), with \( L \) columns. We know that \((a_{pi}, a_{qj}) \in T\) only if the \( i^{th} \) position of sequence \( a_p \) and the \( j^{th} \) position of sequence \( a_q \) are aligned in the same column. So, all nodes contained in the same connected component of \( G' = (V, T) \) correspond to symbols aligned in the same column and we have the following partition of the nodes:

\[
V = X_1 \cup X_2 \cup \cdots \cup X_L,
\]

where \( X_i \) contains all nodes labeled \( a^*_p \) for \( p = 1, 2, \ldots, r \).

Thus, if follows that \( \triangleleft^* \) is a strict partial order on \( X_1, \ldots, X_L \).

We now prove \( \Leftarrow \): Assume \( T \subseteq E \) such that \( \triangleleft^* \) is a strict partial order on the connected components of \( G' = (V, T) \). Extend \( \triangleleft^* \) to a total order \( \triangleleft^*_\text{tot} \), always possible.

Let \( X_1 \cup X_2 \cup \cdots \cup X_m = V \) be the connected components of \( G = (E, T) \), ordered such that \( i < j \) implies \( X_i \triangleleft^*_\text{tot} X_j \).

For each sequence \( a_p \), we define

\[
a^*_p = \begin{cases} 
a_{pi}, & \text{if } a_{pi} \in X_j, \\
- & \text{else,}
\end{cases}
\]

obtaining \( a^*_p = a^*_p, \ldots, a^*_pm \).

We have to show that \( A^* = (a^*_1, \ldots, a^*_m) \) is an alignment.

1. Note that \( a^*_p \) contains all positions of \( a_p \), because each \( a_{pi} \) is contained in some component \( X_j \) and no such component can contain two different positions from the same sequence, due to the strictness of \( \triangleleft^* \).

2. Note that the order of the symbols in \( a_p \) is preserved in \( a^*_p \): Assume \( a^*_p \) and \( a^*_pj \) are two symbols whose order have been reversed in \( a^*_p \), with \( i < j \). By the strictness of \( \triangleleft^* \), there exist two distinct components \( X_s \) and \( X_t \), with \( X_s \triangleleft^* X_t \), such that \( a_{pi} \in X_s \) and \( a_{pj} \in X_t \). On the other hand, the reversal of the order of \( a^*_p \) implies \( X_t \triangleleft^* X_s \), a contradiction to the assumed strictness.

\( \square \)

### 3.3.8 Simple mixed cycles

A mixed cycle \( Z \) is a cycle in the extended alignment graph \( G = (V, E, H) \) that contains both undirected and directed edges, from \( E \) and \( H \), respectively, the latter all in the same direction:

A mixed cycle \( Z \) is called simple, if all nodes in \( Z \cap a_p \) occur consecutively in \( Z \) for every sequence \( a_p \in A \). In other words, a simple mixed cycle enters and leaves a sequence in \( A \) at most once.

The following result says that we can restrict our attention to those mixed cycles that are simple:

**Lemma 3.3.3 (Simple cycles suffice)** The graph \( G' = (V, H \cup T) \) contain a simple mixed cycle if and only if it contains a mixed cycle.
We now obtain a nice result for determining whether a proposed trace is truly the trace of an alignment:

**Theorem 3.3.4 (Trace characterization)** A subset $T \subseteq E$ is a trace, if and only if $G' = (V, H \cup T)$ does not contain a simple mixed cycle.

### 3.3.9 Block partition and the GMT problem

Suppose we are given a set of sequences $A = \{a_1, a_2, \ldots, a_r\}$. The complete alignment graph is usually too big to be useful.

Often, we are given a set of block matches between pairs of the sequences $a_p$ and $a_q$, where a match relates a substring of $a_p$ and a substring of $a_q$ via a run of non-crossing edges (called a block), as shown here for two blocks $D$ and $D'$:

```
    A -> U -> G -> C -> U -> G -> C
   |    |    |    |
   D'   D   D'   D

   C -> U -> G -> A -> U -> G -> A
```

Suppose we are given such a partition $D$ of the edges of $G = (V, E)$ obtained from a set of matches. For a trace we require that:

- for any given block $D \in D$, either all edges in $D$ are realized, or none.

Each block $D$ is assigned a positive weight $\omega(D)$, reflecting the number and weight of the edges that it contains.

**Problem 3.3.5 (GMT problem)** Suppose we are given an extended alignment graph $G = (V, E, H)$ and a partition $D$ of $E$ into blocks with weights $\omega(D)$. The Generalized Maximum Trace (GMT) problem is to determine a set $M \subseteq D$ of maximum total weight such that the edges in $\bigcup_{D \in M} D$ do not induce a mixed cycle on $G$.

Although blocks play an important role in practice, to simplify the following discussion, we will not use them explicitly. However, everything that follows is easily adjusted to the case that a set of blocks is given.

### 3.4 Linear programming

A linear program (LP) consists of a set of linear inequalities,

\[
\begin{align*}
a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \leq b_1 \\
a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \leq b_2 \\
\vdots & \\
a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \leq b_m,
\end{align*}
\]

together with an objective function

\[
c_1x_1 + c_2x_2 + \cdots + c_nx_n
\]

to be optimized, i.e. minimized or maximized.
Linear programs can be efficiently solved using the *simplex method*, developed by George Dantzig in 1947.

There exist powerful computer programs for solving LPs, even when huge numbers of variables and inequalities are involved.

CPLEX is a very powerful commercial LP solver. Moderate size problems can be solved using *lp_solve*, which is free for academic purposes.

The inequalities describe a convex *polyhedron*, which is called a *polytope*, if it is bounded.

For example, the inequalities

\[
\begin{align*}
-1x_1 - 1x_2 & \leq 5 \\
-2x_1 + 1x_2 & \leq -1 \\
1x_1 + 3x_2 & \leq 18
\end{align*}
\]

describe the following hyperplanes and polytope:

For example, the objective function $2x_1 - 3x_2$ takes on a maximum of 6, for $x_1 = 6$ and $x_2 = 2$, and a minimum of $-9$, for $x_1 = 3$ and $x_2 = 5$.

### 3.4.1 Integer linear program

An *integer linear program (ILP)* is a linear program with the additional constraint that the variables $x_i$ are only allowed to take on integer values.

Solving ILPs has been shown to be NP-hard. (See the book by Garey and Johnson 1979, for this and many other NP-completeness results.)

There exist a number of different strategies for approximating or solving such an ILP. These strategies usually first attempt to solve *relaxations* of the original problem, which are obtained by dropping some of the inequalities. They usually also rely on the *LP-relaxation* of the ILP, which is the LP obtained by dropping the integer condition.

### 3.4.2 Integer LP for the GMT problem

How to encode the GMT problem as an integer LP?

Assume we are given an extended alignment graph $G = (V, E, H)$, with $E = \{e_1, e_2, \ldots, e_n\}$.

Each edge $e_i \in E$ is represented by a variable $x_i$, that will take on value 1, if $e_i$ belongs to the best scoring trace, and 0, if not.

Hence, our variables are $x_1, x_2, \ldots, x_n$.

To ensure that the variables are *binary*, we add constraints $x_i \leq 1$ and $-x_i \leq 0$.

Additional inequalities must be added to prevent mixed cycles.

(This and the following is from: Knut Reinert, A Polyhedral Approach to Sequence Alignment Problems, Dissertation, Saarbrücken 1999.)

For example, consider:
There are three possible simple mixed cycles in the graph, one using $e_1$ and $e_3$, one using $e_2$ and $e_3$, and one using $e_2$ and $e_4$. We add the constraints
\[
x_1 + x_3 \leq 1, \\
x_2 + x_3 \leq 1, \\
x_2 + x_4 \leq 1.
\]
to ensure that none of the simple mixed cycles is realized.

For example, consider:

with three edges $e_1$, $e_2$ and $e_3$ that all participate in a simple mixed cycle. The constraint
\[
x_i + x_j + x_k \leq 2
\]
prevents them from being realized simultaneously.

In summary, given an extended alignment graph $G = (V, E, H)$ with $E = \{e_1, e_2, \ldots, e_n\}$, and a score $\omega_i$ defined for every edge $e_i \in E$.

We can obtain a solution to the GMT problem by solving the following ILP:

Maximize
\[
\sum_{e_i \in E} \omega_i x_i,
\]
subject to
\[
\sum_{e_i \in C \cap E} x_i \leq |C \cap E| - 1, \quad \text{for all simple mixed cycles } C, \quad \text{and}
\]
\[
x_i \in \{0, 1\} \quad \text{for all variables } i = 1, \ldots, n.
\]

### 3.4.3 Solving the ILP using branch-and-cut

Solving IPs and ILPs is a main topic in combinatorial optimization. We will take a brief look at the *branch-and-cut* approach.

**Branch-and-cut:** This makes use of two techniques:

- Cutting: to solve an ILP, one considers the LP-relaxation of the problem and repeatedly cuts away parts of the polytope (by adding new constraints) in the hope of obtaining an integer solution.
- Branch-and-bound: an enumeration tree of all possible choices of parameters is partially traversed, computing local upper- and global lower-bounds, which are used to avoid parts of the tree that cannot produce the optimal value.

First note that the number of mixed cycles grows exponentially with the size of the graph. So, initially, we select a polynomial number of constraints. That is, we consider a relaxation of the original problem.

We further relax the problem by solving the LP-relaxation.

If the solution \( \hat{x} \) is not an integer, or is not feasible, then we add an unused constraint to the LP to cut away a part of the polyhedron that contains \( \hat{x} \). (This is a non-trivial operation that we won’t discuss).

This is repeated until an integer solution is found that fulfills all constraints, or until we get stuck.

If no appropriate cut plane can be found, then we branch. That is, we choose a variable \( x_i \) and solve two sub-cases, namely the case \( x_i = 0 \) and the case \( x_i = 1 \). Repeated application produces an enumeration tree of possible cases.

We call an upper bound for the original ILP local, if it is obtained from considering such a subproblem in the enumeration tree.

If the solution found for a subproblem is feasible for the original problem and has a higher score than any solution found so far, then it is recorded and its value becomes the new global lower bound for the original objective function.

Subsequently, we only pursue subproblems whose local upper bound is greater or equal to the global lower bound.

General strategy:

As the details are quite involved, we will skip them.

Example:
3.4.4 An ILP for pairwise alignment

We discuss how to formulate the ILP for the problem of aligning two sequences.

Suppose we are given two sequences \( a = (a_1, \ldots, a_n) \) and \( b = (b_1, \ldots, b_m) \). Let \( s(f, g) \) denote the score for aligning symbols \( f \) and \( g \).

The objective function that we would like to maximize is:

\[
\sum_{1 \leq i \leq n, 1 \leq j \leq m} s(a_i, b_j)x_{ij},
\]

where \( x_{ij} \) is a variable that will indicate whether the edge from node \( a_i \) to node \( b_j \) belongs to the trace, or not.

To ensure that every variable \( x_{ij} \) is binary, we use the inequalities

\[
x_{ij} \leq 1 \quad \text{and} \quad -x_{ij} \leq 0,
\]

for all \( i, j \) with \( 1 \leq i \leq n \) and \( 1 \leq j \leq m \).

In the case of two sequences, every simple mixed cycle is given by an ordered pair of positions \((i, j)\) in sequence \( a \) and an ordered pair of positions \((k, l)\) in sequence \( b \):

\[
\begin{align*}
& a_1 \rightarrow \ldots \rightarrow a_i \rightarrow \ldots \rightarrow a_j \rightarrow \ldots \rightarrow a_n \\
& b_1 \rightarrow \ldots \rightarrow b_k \rightarrow \ldots \rightarrow b_l \rightarrow \ldots \rightarrow b_m
\end{align*}
\]

This gives rise to the following set of inequalities:

\[
x_{il} + x_{jk} \leq 1,
\]

for all \( i, j, k, l \) with \( 1 \leq i \leq j \leq n, 1 \leq k \leq l \leq m \) and, additionally, \( i \neq j \) or \( k \neq l \).

For example, given sequences \( a = \text{GCT} \) and \( b = \text{GT} \), and assume the match- and mismatch scores are 1 and \(-1\), respectively.

In the format used by the program \texttt{lp.solver}, the ILP has the following formulation:

\[
\begin{align*}
\text{max:} & \quad +1 \ast x_{1001} - 1 \ast x_{1002} - 1 \ast x_{2001} - 1 \ast x_{2002} - 1 \ast x_{3001} + 1 \ast x_{3002} \\
& \quad \text{} x_{1001} < 1; \\
& \quad \text{} x_{1002} < 1;
\end{align*}
\]
Here, we use the variable $x_{1000i+j}$ to represent the edge from $a_i$ to $b_j$ for all $i,j$. The program `lp_solve` interprets “<” as “≤” and assigns only non-negative values to variables.

### 3.4.5 The gapped extended alignment graph

The extended alignment graph $G = (V, E, H)$ does not explicitly model gaps. To allow the scoring of gaps, we add a new set $B$ of edges to the graph, joining any two consecutive nodes in a sequence, as indicated here:

For two sequences $a = \text{CGTU}$ and $b = \text{AGGTC}$ we see (a) the gapped extended alignment graph, (b) an alignment and (c) the *gapped trace* that realizes the gapped alignment.

Suppose we are given a gapped extended alignment graph $G = (V, E, H, B)$. When modeling gaps in a trace we require for any pair of sequences that a node must

- either be incident to an alignment edge between the two sequences, or
- it must be incident to or enclosed by exactly one gap edge.

Additionally, we require that a consecutive run of gap characters is regarded as one gap. Without going into details, this gives rise to the definition of a *gapped trace* $(T, C)$, with $T \subseteq E$ and $C \subseteq B$.

Given weights $\omega$ for all edges in $E$ and $B$, a gapped trace can be scored as follows:

$$\alpha((T, C)) = \sum_{e \in T} \omega(e) - \sum_{g \in C} \omega(g).$$

In the gapped trace formulation, it is trivial to encode, linear, affine, or any other reasonable gap cost function.
3.5 Secondary structure

RNA is a single stranded molecule that folds intra-molecularly to form hydrogen-bonded base pairs, mostly C-G and A-U. This configuration is called the secondary structure of RNA.

This can also be depicted as:

(We will discuss RNA secondary structure in detail in a later Chapter.)

3.5.1 Structured sequences

Suppose we are given a set of sequences $A = \{a_1, \ldots, a_r\}$. How can we use information about their secondary structures to obtain biologically better alignments?

Let $a_t$ be a sequence of length $n$. An interaction is a pair $(i, j)$ of positions in $a_t$, with $a_{ti} \neq \cdot$, $a_{tj} \neq \cdot$, and $1 \leq i < j \leq n$. Two interactions $(i, j)$ and $(k, l)$ are in conflict, if $\{i, j\} \cap \{k, l\} \neq \emptyset$.

A set of non-conflicting interactions for $a_t$ is called a (secondary) structure for $a_t$, and we call $\langle a_t, P \rangle$ a structured sequence.

The sequence and interactions depicted in the previous example form a structured sequence.

3.5.2 Structural alignment

Suppose we are given a set of structured sequences

$$A = \{(a_1, P_1), (a_2, P_2), \ldots, (a_r, P_r)\}.$$ 

A structural multiple sequence alignment of $A$ is a set

$$\hat{A} = \{\hat{a}_1, P_1, \ldots, \hat{a}_r, P_r\}$$

of structured sequences such that:

1. $\{\hat{a}_1, \hat{a}_2, \ldots, \hat{a}_r\}$ is a MSA for $\{a_1, a_2, \ldots, a_r\}$, and
2. for all $t \in \{1, 2, \ldots, r\}$ and all $(i, j) \in P_t$, we have

$$(i - \# \text{gaps}(\hat{a}_{t1}, \ldots, \hat{a}_{ti}), j - \# \text{gaps}(\hat{a}_{t1}, \ldots, \hat{a}_{tj})) \in P_t.$$ 

The second condition ensures that the structural MSA only contains interactions that are present in the input.
3.5.3 Scoring a structural alignment

**Goal:** Define a scoring scheme that reflects the biological relatedness of the sequences under consideration, taking their secondary structure into account.

In general terms, a scoring function simply assigns a real number to every possible structural alignment. In practice, scoring functions usually consist of a weighted sum of two parts, one that evaluates the alignment of the characters in the sequences and one that evaluates the alignment of the interactions in the secondary structure.

In the pairwise case, for example, one could use a pairwise sequence alignment score $\text{sim}$ to score the sequence alignment and then define an interaction score $\text{isim} : \Sigma^4 \to \mathbb{R}$ that assigns a score to two pairs of aligned characters that are defined by two matching interactions (where $\Sigma$ denotes the alphabet of characters, excluding 'X').

The following structural alignment score function $\text{rna}(\hat{A})$ can be used to identify pairwise structural alignments of RNA sequences that have both high sequence similarity and high structure conservation:

$$
\text{rna}(\hat{A}) = \sum_{i=1}^{L} \text{sim}(\hat{a}_{1i}, \hat{a}_{2i}) + \sum_{(j,l) \in \hat{P}_1, (q,r) \in \hat{P}_2} \text{isim}(\hat{a}_{1j}, \hat{a}_{1l}, \hat{a}_{2q}, \hat{a}_{2r}).
$$

**Example:**

![Diagram showing pairwise structural alignment with high sequence score and interaction score.]}

3.5.4 Structural alignment graph

Suppose we are given a set of structured sequences $A = \{(a_1, P_1), \ldots, (a_r, P_r)\}$. Let $G = (V, E, H)$ denote a extended alignment graph associated with $\{a_1, \ldots, a_r\}$.

A *structural (extended) alignment graph* $G' = (V, E, H, I)$ is obtained by introducing a set of new edges $I$ that represent all given interactions.

**Example:**

![Diagram of a structural alignment graph.]
3.5.5 Structural traces

Let $G = (V, E, H, I)$ be a structural alignment graph for a set of structured sequences $A = \{(a_1, P_1), \ldots, (a_r, P_r)\}$. A structural trace of $G$ is a pair $(T, B)$ with $T \subseteq E$ and $B \subseteq I$ such that

- the induced subgraph $(V, T \cup H)$ does not contain a simple mixed cycle, and
- there are no two conflicting edges in $B$.

Here are two possible structural traces:

![Structural traces diagram]

**Lemma 3.5.1** Structural traces correspond to structural alignments.

3.5.6 Scoring a structural trace

Let $i_p$ and $i_q$ denote two interaction edges in a structural trace that are contained in the secondary structures of two difference sequences. We say that $i_p$ matches $i_q$, if the two alignment edges $e_l$ and $e_r$ joining the two left respectively two right nodes of $i_p$ and $i_q$ are realized, i.e. if $e_l, e_r \in T$.

![Scoring a structural trace diagram]

We call any such set $\{i_p, i_q, e_l, e_r\}$ an interaction match.

Let $(T, B)$ be a structural trace. As for conventional traces, each edge $e \in T$ is given a weight $\omega(e)$ which is simply the score for aligning the two corresponding symbols.

Any interaction match $\{i_p, i_q, e_l, e_r\}$ is completely specified by the two edges $e_l$ and $e_r$ and so we can denote it by $m_{tr}$ and assign a weight $\omega(l, r)$ to it. Let $M$ denote the set of all interaction matches:

$$M = \{m_{tr} \mid m_{tr} = \{i_p, i_q, e_l, e_r\} \text{ is an interaction match in } G\}.$$

We define the score of a structural trace as:

$$S((T, B)) = \sum_{e \in T} \omega(e) + \sum_{i_p, i_q \in B, e_l, e_r \in T} \omega(l, r).$$

We can find the maximal scoring trace by solving an ILP.
3.5.7 Maximal scoring structural trace

Given a structural alignment graph $G = (V, E, H, I)$, the score $\omega_i$ for realizing an edge $e_i \in E$ and the score $\omega_{lr}$ for realizing an interaction match $m_{lr}$.

The problem of maximizing the score of a structural trace can be formulated as the following ILP:

Maximize \[ \sum_{e_i \in E} \omega_i x_i + \sum_{m_{ij} \in M} \omega_{ij} x_{ij}, \]
subject to \[ \sum_{e_i \in C \cap E} x_i \leq |C \cap E| - 1, \quad \text{for all simple mixed cycles } C, \]
\[ \sum_{j} x_{ij} \leq x_i \quad \text{and} \quad \sum_{i} x_{ij} \leq x_j, \quad \text{for all } i, j, \text{ and} \]
\[ x_i, x_{ij} \in \{0, 1\} \quad \text{for all variables.} \]

Lemma 3.5.2 Any solution to this ILP corresponds to a structural trace.

Proof The set of all variables $x_i$ with $x_i = 1$ define a set of edges $T \subseteq E$ and the simple mixed cycle inequalities ensure that $T$ is a trace.

The set of all variables $x_{ij}$ with $x_{ij} = 1$ define a set of matched interactions $B \subseteq I$. To establish that $B$ does indeed give rise to a structural trace, note that the second set of constraints in the ILP ensure that

- an interaction match $m_{lr}$ can only be realized if both $e_l$ and $e_r$ are realized, and
- only one interaction match can “use” any specific alignment edge $e_i$ as its left- or right-connecting edge, and so no conflicts can arise.

\[ \square \]

3.6 Scoring an MSA

As for pairwise alignments we assume independence of the different columns of an MSA and use an additive function for the alignment score:

score $\alpha(A^*)$ of an MSA $A^*$ is the sum of column scores:

\[ \alpha(A^*) := \sum_{i=1}^{L} s(a_{1i}^*, a_{2i}^*, \ldots, a_{ri}^*). \]

Assume that $s(a_{1i}^*, a_{2i}^*, \ldots, a_{ri}^*)$ is a function that returns a score for every combination of $r$ symbols (including the gap symbol).

$r = 2$

\[
\begin{array}{c c c c c}
 a_{1i} & a_{1i} \\
 a_{2j} & a_{2j} & - \\
\end{array}
\]

$r = 3$:

\[
\begin{array}{c c c c c c}
 a_{1i} & a_{1i} & a_{1i} & - & - & a_{1i} \\
 a_{2j} & a_{2j} & a_{2j} & a_{2j} & - & - \\
 a_{3k} & a_{3k} & a_{3k} & a_{3k} & - & - \\
\end{array}
\]
For $r$ sequences, the number of different column types is

$$\sum_{i=1}^{r-1} \binom{r}{i} = 2^r - 1$$

where $i$ is the number of gaps.

Considerations for the scoring function $s$:

- Scoring matrices such as BLOSUM or PAM defined for comparing two symbols
- Generating scoring matrices for more than two sequences is unfeasible because of the multitude of combinations of column types
- Independence of order in column: e.g. $s(A, -, A, L) = s(L, A, A, -)$
- $s$ should reward many matching or similar characters, and penalize gaps and dissimilar characters.

A multiple sequence alignment is simply an alignment of more than two sequences, like this:

A small fragment of a multiple alignment of the hemoglobin protein sequences and homologues. The MSA shows conserved residues, conserved regions and more sophisticated patterns. Multiple alignments are helpful for protein structure prediction.

### 3.6.1 The sum-of-pairs (SP) score

Consider two sequences $A_p^*$ and $A_q^*$ in the alignment. For two aligned symbols $u$ and $v$ we define:

$$s(u, v) := \begin{cases} 
\text{match score for } u \text{ and } v, & \text{if } u \text{ and } v \text{ are residues,} \\
-d & \text{if either } u \text{ or } v \text{ is a gap, or} \\
0 & \text{if both } u \text{ and } v \text{ are gaps.}
\end{cases}$$

(Note that $u = -$ and $v = -$ can occur simultaneously in a multiple alignment.)

Consider pairwise alignment of sequences $A_p$ and $A_q$ imposed by a multiple alignment $A^*$ of $r$ sequences.

Denote the score of this induced pairwise alignment as

$$s(A_p^*, A_q^*) = \sum_{i=1}^{L} s(a_{pi}^*, s_{qi}^*)$$

Sum up the pairwise scores for a multiple alignment:

$$S(A_1^*, \ldots, A_r^*) = \sum_{1 \leq p < q \leq r} s(A_p^*, A_q^*)$$
Definition 3.6.1 The sum-of-pairs (SP) score of an alignment is defined as\footnote{introduced by Carillo and Lipman in “The multiple sequence alignment problem in biology”, SIAM J. Appl. Math. 48:1073-1082, 1988.}

\[ \alpha_{SP}(A^*) := \sum_{1 \leq p < q \leq r} s(A^*_p, A^*_q) = \sum_{i=1}^L s_{SP}(a^*_1, a^*_2, \ldots, a^*_r), \]

with

\[ s(A^*_p, A^*_q) := \sum_{i=1}^L s(a^*_p, a^*_q) \]

and

\[ s_{SP}(a^*_1, \ldots, a^*_r) := \sum_{1 \leq p < q \leq r} s(a^*_p, a^*_q). \]

Thus, we obtain a score for a multiple alignment based on a pairwise-scoring matrix.

3.6.2 Example

\[
\begin{align*}
\text{Multiple alignment:} & \quad \begin{array}{cccc}
\text{Seq. 1} & \ldots & N & \ldots & N & \ldots & N & \ldots \\
\text{Seq. 2} & \ldots & N & \ldots & N & \ldots & N & \ldots \\
\text{Seq. 3} & \ldots & N & \ldots & N & \ldots & N & \ldots \\
\text{Seq. 4} & \ldots & N & \ldots & N & \ldots & C & \ldots \\
\text{Seq. 5} & \ldots & N & \ldots & C & \ldots & C & \ldots \\
\end{array} \\
\text{Comparisons:} & \quad \begin{array}{ccc}
\text{(1)} & \text{(2)} & \text{(3)} \\
\end{array} \\
\text{# comparisons} & \quad \begin{array}{ccc}
10 & 10 & 10 \\
\binom{5}{2} & \binom{5}{2} & \binom{5}{2} \\
\text{N-N pairs:} & 10 & 6 & 3 \\
\text{N-C pairs:} & 0 & 4 & 6 \\
\text{C-C pairs:} & 0 & 0 & 1 \\
\text{BLOSUM62:} & 60 & 24 & 9 \\
\end{array}
\end{align*}
\]

(BLOSUM62 scores: N-N: 6, N-C: -3, C-C: 9)

3.7 Optimal MSA

Optimal multiple alignment problem: Given \( r \) sequences \( A_1, A_2, \ldots, A_r \) and an alignment score function \( s \), find an alignment \( A^* \) of \( A_1, A_2, \ldots, A_r \) such that \( s(A^*) \) is maximal among all possible alignments of \( A_1, A_2, \ldots, A_r \). Such an alignment \( A^* \) is called an optimal alignment, and

\[ s(A_1, A_2, \ldots, A_r) := s(A^*) \]

is the optimal alignment score of \( A_1, A_2, \ldots, A_r \).

3.8 The dynamic program for a global MSA

Dynamic programs developed for pairwise alignment can be modified to multiple alignments. We discuss how to compute a global MSA for three sequences, in the case of a linear gap penalty. Suppose we are given:

\[
A = \begin{cases}
A_1 = (a_{11}, a_{12}, \ldots, a_{1n_1}) \\
A_2 = (a_{21}, a_{22}, \ldots, a_{2n_2}) \\
A_3 = (a_{31}, a_{32}, \ldots, a_{3n_3}).
\end{cases}
\]

We proceed by computing the entries of an \((n_1 + 1) \times (n_2 + 1) \times (n_3 + 1)\)-matrix \(F(i, j, k)\) recursively. After the computation, \(F(n_1, n_2, n_3)\) will contain the best score \(\alpha\) for a global alignment \(A^*\). As in the pairwise case, we can use traceback to recover an optimal alignment.

The main recursion is (remember there are \(2^r - 1 = 8 - 1 = 7\) types of columns in this case):

\[
F(i, j, k) = \max \begin{cases} 
F(i - 1, j - 1, k - 1) + s(a_{1i}, a_{2j}, a_{3k}), \\
F(i - 1, j - 1, k) + s(a_{1i}, a_{2j}, -), \\
F(i - 1, j, k - 1) + s(a_{1i}, -, a_{3k}), \\
F(i, j - 1, k - 1) + s(-, a_{2j}, a_{3k}), \\
F(i - 1, j, k) + s(a_{1i}, -, -), \\
F(i, j - 1, k) + s(-, a_{2j}, -), \\
F(i, j, k - 1) + s(-, -, a_{3k}), \\
\end{cases}
\]

for \(1 \leq i \leq n_1, 1 \leq j \leq n_2, 1 \leq k \leq n_3\),

where \(s(a, b, c)\) returns a score for a given column of symbols \(a, b, c\); for example, \(s = s_{SP}\), the sum-of-pairs score.

Example:

\[
A = \begin{cases} 
A_1 = \text{ABDE} \\
A_2 = \text{ACBE} \\
A_3 = \text{ADCEE} \\
\end{cases} \implies A^* = \begin{cases} 
A_1^* = \text{A - B - D - E - } \\
A_2^* = \text{A C B - - E - } \\
A_3^* = \text{A - - D C E E } \\
\end{cases}
\]

3.8.1 Complexity of dynamic program for an MSA

DP complexity for MSA of \(r\) sequences of length \(n\) using SP-score:

Space complexity: \(O(n^r)\)

Time complexity: \(O(r^2 \cdot n^r \cdot 2^r)\).

\textbf{Theorem 3.8.1} Computing an MSA with optimal SP-score is NP-complete \(^2\).

\(^2\)L. Wang and T. Jiang, J Comp Biol 1994
3.9 The Carillo-Lipman MSA approach

Algorithm by Carrillo and Lipman\(^3\): reduces search space and time needed to compute MSA.

Basic idea: for two similar sequences, optimal path for alignment is close to main diagonal of DP matrix.

Generalize: for three or more similar sequences, optimal alignment path near the center of the hypercube.

Carillo and Lipman approach: compute bounds that approximate the center of a multi-dimensional hypercube (similar to the \(k\)-banded algorithm).

3.9.1 Projections

Every multiple alignment induces a set of pairwise alignments.

\[
A^* = \begin{cases}
A_1^* = A - B \ D - E \\
A_2^* = A \ C \ B - - E - \\
A_3^* = A - - D \ C \ E \ E
\end{cases}
\]

This example induces three pairwise alignments:

\[
A - B \ D \ E \quad A - B \ E \quad A - - - D \ C \ E \ E
\]

Projecting 3-D multiple alignment path on to a 2-D face of the cube.

Problem: projected pairwise alignments are not optimal.

Still: projected pairwise alignments cannot be bad alignment (if all pairwise projections were bad, the whole multiple alignment would also have a bad sum-of-pairs score).

Carillo-Lipman’s approach: restricts the search space to those vertices that are part of close-to-optimal pairwise alignments for each pairwise projection.

Example: three-dimensional lattice with optimal MSA path

Projection of the alignment onto 2 sequences = light source mapping path onto opposing face

---

If the sequences are rather similar, and the cost for introducing gaps is not too low, then we can expect a priori that the optimal alignment path is contained in a “polyhedron” close to the main diagonal.

General idea to find polyhedron: via branch and bound

Assume: MSA computed by heuristic alignment with path X units from optimal MSA path

Polyhedron with radius X around heuristic MSA path contains optimal path.

We will calculate a “polyhedron” consisting of those nodes in the lattice that are traversed by at least one path which has, in all of its projections, a cost below a specific bound. We will calculate one bound for each possible projection (i.e. each possible pair of sequences). For 3 sequences, we will calculate 3, and for 4 sequences, we will calculate 6 bounds. We will show that these bounds are obeyed by the projections of the optimal multiple alignment.

To obtain the bounds, we will look at the pairwise projections of a heuristic alignment, and of the hypothetical optimal multiple alignment.

\[ A = \{A_1, \ldots, A_r\} \]

\[ \alpha_{SP}(A^*) = \sum_{1 \leq p < q \leq r} s(A_p^*, A_q^*) \]

\[ A^o = \text{optimal SP-alignment:} \]

\[ \alpha_{SP}(A^o) = \max_{A^*} \alpha_{SP}(A^*) . \]

Projection of \( A^* \) onto \( A_p \) and \( A_q \): \( A^*|p,q \)

Optimal pairwise alignment score = \( s(\hat{A}_p, \hat{A}_q) \)

Score of the projected optimal alignment: \( s(A^o|p,q) \)

Lower bound for \( \alpha_{SP}(A^o) \):

\[ B(A) \leq \alpha_{SP}(A^o) \]

\[ = \sum_{i<j} s(A^o|i,j) \]

\[ = s(A^o|p,q) + \sum_{i<j,(i,j)\neq(p,q)} s(A^o|i,j) \]

\[ \leq s(A^o|p,q) + \sum_{i<j,(i,j)\neq(p,q)} s(\hat{A}_i, \hat{A}_j) \]

\[ = (s(A^o|p,q) - s(\hat{A}_p, \hat{A}_q)) + \sum_{i<j} s(\hat{A}_i, \hat{A}_j) \]
\[
\Rightarrow B(A) - \left( \sum_{i<j} s(\hat{A}_i, \hat{A}_j) - s(\hat{A}_p, \hat{A}_q) \right) \leq s(A^o|p, q)
\]

\[
B_{pq} := B(A) - \left( \sum_{i<j,(i,j)} s(\hat{A}_i, \hat{A}_j) - s(\hat{A}_p, \hat{A}_q) \right) \leq s(A^o|p, q)
\]

This equation provides a lower bound, called the Carillo-Lipman bound, on the pairwise score \(s(A^o|p, q)\), for any pair \((p, q)\) we may have selected.

Note that in order to compute the lower bound, only pairwise optimal alignments and a heuristic multiple alignment need to be calculated.

### 3.9.2 The Carrillo-Lipman path

**Question:** how to compute \(B(A)\)?

**Answer:** use algorithm to construct multiple alignments from pairwise alignments such as the star alignment algorithm or one of the heuristics introduced below.

Let \(X_{p,q}\) be the set of paths which have, in projection, \((p, q)\) scores larger than or equal to \(B_{pq}\). These paths consist of coordinate pairs \((i_p, i_q)\), such that the optimal alignment of \(A_p\) and \(A_q\) through \((i_p, i_q)\) scores more than \(B_{pq}\).

Exploiting the bounds, pair by pair, we want to consider only paths through the hyperlattice that obey the Carrillo-Lipman bound in all their projections. These paths are described by the set

\[
X = \bigcap_{(p,q), p<q} X_{p,q}
\]

The algorithm is restricted to evaluate only the cells \((i_1, i_2, \ldots, i_r)\) for which \((i_p, i_q)\) is in \(X_{pq}\). Thus \(X\) describes the polyhedron in which we will find the optimal multiple alignment.

(Figure from Durbin et al., p. 144)

### 3.9.3 The MSA program

The above algorithm was implemented in the multiple sequence alignment program MSA\(^4\). MSA can optimally align five to seven sequences of length 200-300 residues. The drawback with using MSA is that it requires an enormous amount of both computer time and memory to align more than a few distantly related sequences. The size of the problems solved by MSA are directly related to the sequence lengths, the number of sequences, and the amount of sequence diversity.

A newer implementation is QAlign: http://gi.cebitec.uni-bielefeld.de/QAlign.

3.10 Heuristics for MSA

Since we appreciate that computing optimal MSA are computationally too demanding, we turn now to heuristics. The majority of the programs build the MSA by gradually adding one sequence after the other. At each step basically we are performing a pairwise alignment. We distinguish progressive and iterative alignment methods.

Other methods do not use pairwise alignments. The program DIALIGN starts from ungapped local alignments, while for example SAGA uses genetic programming to compute an MSA.

3.11 Progressive alignment

Principle idea of progressive alignment:

Implementations differ

1. in the order in which the sequences are aligned (ie. depend on the tree reconstruction algorithm)
2. whether during the alignment process a single multiple alignment is generated or several ones, following a tree structure,
3. which parameters are used (such as scoring function, gap penalties, weight of individual sequences)
3.11.1 Progressive alignment along a tree

![Progressive Alignment Diagram]

3.11.2 Order of alignment matters

Example:

\[ A_1 = \text{ALVK}, A_2 = \text{APFK}, A_3 = \text{ALFVK}, A_4 = \text{APFVK}. \]

Performing alignments in different orders could result in:

\[
(A_1, A_2), (A_3, A_4) \quad \text{or} \quad (A_1, A_3), (A_2, A_4)
\]

ALVK - ALVK
APFK - APFK
ALFVK - ALFVK
APFVK - APFVK

3.11.3 Pseudocode for progressive alignment

The general algorithm for progressive alignments is as follows:

**Input:** a set \( A = \{A_1, \ldots, A_r\} \) of sequences

**var:**

\( C = \) current set of alignments

**begin**

\( C = \emptyset \)

for \( i = 1, 2, \ldots, r \) do

\( C := C \cup \{A_i\} \)

do

choose two sub-alignments \( A_p^*, A_q^* \) from \( C \);

\( C = C - \{A_p^*, A_q^*\} \)

\( A_s^* := \text{align}(A_p^*, A_q^*) \)

\( C = C \cup \{A_s^*\} \)
while $|C| > 1$
end

### 3.11.4 Aligning two sub-alignments

How do we align two (sub-)alignments?  

$A^*_p = \text{MSA of } \{A_{p_1}, \ldots, A_{p_n}\}$ and $A^*_q = \text{MSA of } \{A_{q_1}, \ldots, A_{q_m}\}$.

In order to now align $A^*_p$ with $A^*_q$ we either do  

- a **complete alignment**

or  

- a **pair-guided alignment**

All sequences of the two sub-alignments are used in a dynamic programming procedure. A sequence is aligned to a group of already aligned sequences by pairwise aligning that sequence to each member of the group. Then the alignment with the largest score determines how the sequence is aligned to the group. Similarly, groups are aligned to groups.

### 3.11.5 Pair-guided alignment of two sub-alignments

Pair-guided alignment: choose two specific sequences, one from $A^*_p$ and one from $A^*_q$, align, and complete final multiple alignment from this pairwise alignment.

### 3.11.6 Example pair-guided alignments

Let the two (sub-)alignments be

| ALEE | A-ERE |
| A-EE | ALER- |
| -LEE |

Align 1st sequence of first (sub-)alignment with last of second:

| ALEE- |
| ALER- |

Add gaps to other sequences in the sub-alignments. Final multiple alignment is then

| ALEE- |
| A-EE- |
| -LEE- |
| A-ERE |
| ALER- |

### 3.11.7 Scoring alignments of (sub)alignments

Suppose we are given two MSA (called *profiles* in this context) $A^*_1 = \{A_1, \ldots, A_r\}$ and $A^*_2 = \{A_{r+1}, \ldots, A_n\}$. Here, we discuss the alignment of profiles in the case of the SP-score and linear gap scores. In this case, we can set $s(-, a) = s(a, -) = g$ and $s(-, -) = 0$ for all $a \in A^*_1$ or $A^*_2$. 
3.11.8 Profile alignment

Definition 3.11.1 A profile alignment of $A_1^*$ and $A_2^*$ is an MSA

$A^* = \begin{cases} 
A_1^* = a_{11}^*, a_{12}^*, \ldots, a_{1L}^* \\
\ldots \\
A_r^* = a_{r1}^*, a_{r2}^*, \ldots, a_{rL}^* \\
A_{r+1}^* = a_{r+1,1}^*, a_{r+1,2}^*, \ldots, a_{r+1,L}^* \\
\ldots \\
A_n^* = a_{n1}^*, a_{n2}^*, \ldots, a_{nL}^* 
\end{cases}$

obtained by inserting gaps in whole columns of $A_1^*$ or $A_2^*$, without changing the alignment of either of the two profiles.

The SP-score of the profile alignment $A^*$ is:

$$\alpha_{sp}(A^*) = \sum_{1 \leq p < q \leq n} \sum_{i=1}^{L} s(a_{pi}^*, a_{qi}^*) = \sum_{i=1}^{L} \sum_{1 \leq p < q \leq n} s(a_{pi}^*, a_{qi}^*) = \left( \sum_{i=1}^{L} \sum_{1 \leq p < q \leq L} s(a_{pi}^*, a_{qi}^*) + \sum_{i=1}^{L} \sum_{1 \leq p < q \leq n} s(a_{pi}^*, a_{qi}^*) + \sum_{i=1}^{L} \sum_{1 \leq p \leq r < q \leq n} s(a_{pi}^*, a_{qi}^*) \right).$$

Alignment score of $A_1^*$ Alignment score of $A_2^*$ cross terms

The third sum can be optimized using standard pairwise alignment, with the modification that columns are scored against columns by adding their pair scores.

Clearly, either or both profiles may consist of a single sequence. In the former case, we are aligning a single sequence to a profile and in the latter case, we are simply aligning two sequences.

3.11.9 Guide trees

Most important part of progressive alignment algorithms: the order of which sequences to align first.

Idea: start with pairs of most similar sequences, then continue with pairs of less similar, ...

Determine similarity of sequences with a guide trees = a binary phylogenetic alignment tree

root node = full multiple alignment

leaves = individual sequences

question: How to determine tree

3.11.10 Feng-Doolittle

First progressive alignment algorithms published 1987 by Feng and Doolittle\textsuperscript{5}.

Idea: a pair of sequences with minimal distance has also evolutionary diverged most recently.

To compute a multiple alignment one should therefore follow the evolutionary path of the sequences.

Algorithm:

\textsuperscript{5}Feng, D-F & Doolittle, RF. Progressive sequence alignment as a prerequisite to correct phylogenetic trees. J. Mol. Evol. 25:351-360, 1987
1. Calculate all \( \binom{n}{2} \) pairwise alignment scores and convert them into a distance score.


3. Proceed along the guide tree: start from the first node that was added to the tree and align the two children nodes (which maybe either two sequences, or one sequence and one sub-alignment, or two sub-alignments). Repeat for all other nodes in their tree order until one reaches the root, i.e. until all sequences have been aligned.

### 3.11.11 Distance score of Feng-Doolittle

\[
D = -\log S_{\text{eff}} = -\log \frac{S_{\text{obs}} - S_{\text{rand}}}{S_{\text{max}} - S_{\text{rand}}}
\]

\( S_{\text{obs}} \) = observed score for a pair of sequences

\( S_{\text{max}} \) = the maximum score (exercise: how computed?)

\( S_{\text{rand}} \) = expected score of an alignment of two random sequences of equal length and composition as the pair in question (Exercise: how computed?)

\( S_{\text{eff}} \) = normalised percentage similarity: with increasing evolutionary distance this score decays exponentially against zero.

The sequence-sequence alignments are conducted using the complete alignment approach.

Also here gaps that have already been inserted remain:

“Once a gap, always a gap”.

### 3.12 CLUSTALW

CLUSTALW is one of the most popular programs for computing an MSA. The weighted version, CLUSTALW, is the successor of CLUSTAL, which was introduced about ten years ago. It is based on the Feng-Doolittle method.

**Algorithm** CLUSTALW progressive alignment:

1. Construct a distance matrix of all \( \binom{n}{2} \) pairs by pairwise dynamic programming alignment followed by approximate conversion of similarity scores to evolutionary distances.

2. Construct a guide tree using the Neighbor Joining tree-building method from the distance matrix.

3. Progressively align sequences at nodes of tree in order of decreasing similarity, using sequence-sequence, sequence-profile and profile-profile alignment.

ClustalW uses two distance scores derived from an optimal pairwise alignment of sequences \( A_i \) and \( A_j \). One is the observed distance:

\[
D_{ij} = 1 - \left( \frac{s_{ij}}{L} \right)
\]


where \( s_{ij} \) = number of identities in the best alignment between \( A_i \) and \( A_j \) divided by \( L \), the number of positions considered (gap positions are excluded). Thus this distance score gives the relative number of differences per site.

The other score uses the corrected distance: the corrected distance is calculated using the Kimura method (Kimura 1983):

\[
D_{ij} = -\ln(1 - d_{ij} - d_{ij}^2/5)
\]

where \( d_{ij} \) = number of mismatches in optimal pairwise alignment of the two sequences, not counting gaps.

There are no provable performance guarantees associated with the program. However, it works well in practice and the following features contribute to its accuracy:

- Sequences are weighted to compensate for the defects of the SP score.
- The substitution matrix used is chosen based on the similarity expected of the alignment, e.g. BLOSUM80 for closely related sequences and BLOSUM50 for less related ones.
- Position-specific gap-open profile penalties are multiplied by a modifier that is a function of the residues observed at the position (hydrophobic residues give higher gap penalties than hydrophilic or flexible ones.)
- Gap-open penalties are also decreased if the position is spanned by a consecutive stretch of five or more hydrophilic residues.
- Gap-open and gap-extension penalties increase, if there are no gaps in the column, but gaps nearby. (This tries to force gaps to occur in the same places.)

3.13 When will progressive alignments work and when not?

adapted from Notredame, Pharmacogenomics (2002) 3 (1)
3.14 T-COFFEE

T-COFFEE\(^7\) - short for Tree-based Consistency Objective Function for alignment Evaluation - is also a program that progressively aligns sequences in order to build an MSA.

T-COFFEE follows the idea of consistency: An MSA is consistent if it agrees best with all optimal pairwise alignments, similar to the Carrillo-Lipman approach.

Main difference to ClustalW: in T-Coffee the extended library of scores replaces a substitution matrix.

The library concept of T-COFFEE:

First a primary (reference) library is built from pairwise alignments of all input sequences. For each pair of sequences a global alignment is computed as well as a the top local alignments that are not overlapping.

Then a weight is assigned: percent identity of the shorter of the two aligned sequences.

Example (from T-COFFEE paper):

\[
\begin{align*}
\text{Seq X: GARFIELD THE LAST FAT CAT} & \quad w=88 \\
\quad | | | | | | | | | & \\
\text{Seq Y: GARFIELD THE FAST CAT ---} & \\
\text{Seq X: GARFIELD THE LAST FA-T CAT} & \quad w=77 \\
\quad | | | | | | | | | & \\
\text{Seq V: GARFIELD THE VERY FAST CAT} & \\
\text{Seq Y: GARFIELD THE ---- FAST CAT} & \quad w=100 \\
\quad | | | | | | | | | & \\
\text{Seq V: GARFIELD THE VERY FAST CAT} &
\end{align*}
\]

3.14.1 Extended library in T-COFFEE

Extension of primary library to extended library.

Combine alignments (scores) into one by using other alignments:

\[
\begin{align*}
\text{Seq X: GARFIELD THE LAST FAT CAT} & \quad w=88 \\
\quad | | | | | | | | | & \\
\text{Seq Y: GARFIELD THE FAST CAT ---} & \\
\text{Seq X: GARFIELD THE LAST FA-T CAT} & \quad w=77 \\
\quad | | | | | | | | | & \\
\text{Seq V: GARFIELD THE VERY FAST CAT} & \\
\text{Seq Y: GARFIELD THE ---- FAST CAT} & \quad w=100 \\
\quad | | | | | | | | | & \\
\text{Seq V: GARFIELD THE VERY FAST CAT} &
\end{align*}
\]

Combine all alignments of X and Y into one: consider also alignments of X with V as well as Y with V. Residues that are aligned identically in all three sequences get weight of the previous alignment of X and Y (88), plus the lowest weight of added ones (here min(77,100)=77).

This results in regions of the alignment that are consistent with the aligned regions in other library alignment.

In summary, the weight associated with a pair of residues will be the sum of all the weights gathered through the examination of all the triplets involving that pair. Weights will be zero for any residue pairs that never occur.

The complete set of pairs constitutes the extended library.

Multiple alignment is then done progressively, using a DP algorithm with the scores of the library.

---

3.15 Segment-based alignment methods

Progressive methods such as ClustalW or T-COFFEE have properties that are not necessarily desirable in all applications:

Alignment depends on many parameters: cost function or substitution matrix, gap penalties, and a scoring function (e.g. sum of pairs) for the multiple alignment.

Problem No 1: these parameters need to be chosen in advance.

Problem No 2: global method may not align local similarities well

Example:

\[
A = \begin{pmatrix}
A & F & A & T & C & A & T & C & A \\
A & C & A & T & - & - & - & A \\
\end{pmatrix}
\]

If instead of individual positions, whole units of local similarities are compared and used to build a global multiple alignment, the result will depend much less on the particular alignment parameters, and instead reflect more the local similarities in the data, like in the following example:

\[
A' = \begin{pmatrix}
A & F & A & T & C & A & T & C & A \\
A & - & - & C & A & T & - & A \\
\end{pmatrix}
\]

This is the idea of segment-based sequence alignments.

3.15.1 Segment identification

The first step in a segment-based alignment approach is to identify the segments. There are different possibilities to obtain segments:

- use Smith-Waterman local alignments
- use gap-free local alignments. In this case, the segments correspond to diagonal segments of dynamic programming matrix.

3.15.2 DIALIGN’s segment identification

The approach followed by Morgenstern et al. in the DIALIGN method: use a statistical objective function that assigns the score \( P_D(l_D, s_D) \) to a diagonal \( D \) of length \( l_D \) with \( s_D \) matches (see also BLAT):

\[
P_D(l_D, s_D) = \sum_{i=s_D}^{l_D} \binom{l_D}{i} p^i (1-p)^{l_D-i}
\]

Choose \( p = 1/4 \) for nucleotides and \( p = 1/20 \) for amino acids.

Compute weight of a diagonal to be the negative logarithm of score:

\[
w_D = - \ln(P_D)
\]

3.15.3 Segment assembly

After a set of segments is identified, it is not necessarily possible to assemble them into a global alignment, since they might be inconsistent: in the case of two sequences, two segments are inconsistent if they “cross”:
More generally, a set of segments of multiple sequences \( s_1, s_2, \ldots, s_k \) is inconsistent if there exists no multiple alignment of these sequences that induces all these segments. Several methods exist to select from a set of segments a subset of consistent segments.

### 3.15.4 DIALIGN’s segment assembly

DIALIGN employs the following greedy algorithm for the alignment:

1. Compute all pairwise local alignment segments (diagonals) and compute their respective weights
2. Rank diagonals by weight
3. Compute a global alignment by adding diagonals in order of decreasing weight as long as consistency is achieved

Given a set of consistent segments, in general there will still remain unaligned characters that are not contained in any of the consistent segments. In order to arrive at a global multiple alignment, it is desirable to also add these characters to the alignment. Different strategies can be followed. The DIALIGN method simply fills the remaining regions with gaps, letting those characters unaligned that are not contained in any segment.

In any case, a global alignment is to be computed that respects the segments as fixed anchors, while for the remaining characters freedom is still given, as long as the resulting alignment is consistent with the segments.

### 3.16 Consensus sequences

Often a multiple sequence alignment is computed to deduce a so-called consensus sequence. A consensus sequence is supposed to describe the characteristic similarities within a multiple alignment.

**Definition 3.16.1** Let \( A^\ast \) be a multiple alignment of a set \( A \) of sequences. The consensus symbol of column \( i \) of \( A^\ast \) is the symbol for which the sum of all distances to all symbols existing in the column is minimal.

**Definition 3.16.2** The consensus sequence \( \text{Cons}_{A^\ast} \) of a multiple alignment \( A^\ast \) is the concatenation of all consensus symbols.

Of course from the definition it remains unclear how to compute the distance between symbols. For the distance computation one can use scoring matrices. However, remember that typical scoring matrices such as BLOSUM or PAM are similarity rather than distance scoring matrices. Therefore the similarity score must first be converted into a distance-based score.

The simplest distance score, the Hamming distance then results in the symbol that appears most frequently in the column (the so-called simple majority rule).
Question: how to compute the distance between symbols?
Solution 1: use scoring matrices,
BUT typical scoring matrices such as BLOSUM or PAM are similarity scoring matrices, not distance scoring matrices.
Solution 2: use Hamming distance: then consensus symbol is the symbol that appears most frequently in the column.
Example: The following consensus sequence was computed using the simple majority rule.

\[
\begin{array}{cccc}
A & T & A & C \\
A & - & A & T \\
A & - & G & C
\end{array}
\]

Consensus: \(A - A A G C\) or \(A - A T G C\) or \(A - A G G C\)

This example shows that of course the consensus sequence need not be unique.
Next we define

**Definition 3.16.3** Let \(d(i) = \sum_{j=1}^L D(Cons_{A^*}(i), A_j^*(i))\). The alignment error of \(Cons_{A^*}\) is equal to \(\sum_{i=1}^L d(i)\), with \(L\) equal to the length of the alignment.

**Definition 3.16.4** The optimal consensus multiple alignment is a multiple alignment \(A^*\), whose consensus sequence \(Cons_{A^*}\) has a minimal alignment error.

Note: So far no exact method is known that allows to compute the optimal consensus alignments.

### 3.17 Conserved patterns in sequences

Examples are:

- Transcription factor binding sites

\[
\begin{align*}
SP & \ldots gcctt AATTTTACTATATAC TATAA ccatt \ldots \\
ST & \ldots cagat ATAAATGATATAGT GGT TATA gtaaa \ldots \\
ST & \ldots atcctt TTTTATTAATAATCGTATTA gcagc \ldots \\
EC & \ldots aggcgt ATAAATGATATAGTG TTTA gttag \ldots \\
EC & \ldots accttt TTTTATTAATAATCGTATTA gt cac \ldots \\
VC & \ldots ttata ACTAATATTATATAATATGT gtgtc \ldots \\
YP & \ldots gctga TGAAATGATATATCGTTATA taaga \ldots
\end{align*}
\]

- Protein domains
Protein domain architecture of tyrosine kinases (Figure from Robinson et al., 2000):

3.18 Local MSA applications

- Discovery: local MSA allows to *identify* conserved patterns in sequences
- Modeling: construct a probabilistic model from local MSA (representation of local MSA)
- Recognition: find new instance of a pattern based on given model

- Discovery:
  - Hidden Markov Models (HMM)
  - PSI-Blast
  - Expectation Maximization (EM) algorithm
- Modeling
  - HMMs
  - Position Specific Scoring Matrix (PSSM) - profiles
- Recognition
  - Profile alignment
  - HMM - Viterbi

3.19 Profiles and motifs

Let us look at the following two quotes:

There are many short sequences that are often (but not always) diagnostics of certain binding properties or active sites. These can be set into a small subcollection and searched against your sequence. (R.F. Doolittle)
In some cases, the structure and function of an unknown protein which is too distantly related to any protein of known structure to detect its affinity by overall sequence alignment may be identified by its possession of a particular cluster of residues types classified as a motifs. The motifs, or templates, or fingerprints, arise because of particular requirements of binding sites that impose very tight constraint on the evolution of portions of a protein sequence. (A.M. Lesk)

A *motif* is a short signature pattern identified in the conserved region of the multiple alignment. A consensus sequence as we saw it is a rather incomplete representation of an MSA. Think of consensus as an “ancestor” motif, from which mutated motifs emerged. A much better representation is a *profile*. A profile displays the frequency of each amino acid or nucleotide at each position of the multiple alignment.

**Definition 3.19.1** Let $A^*$ be a multiple alignment of length $L$ of a set of $r$ sequences over the alphabet $\Sigma' = \Sigma \cup \{-\}$. A profile is an $L \times |\Sigma'|$ matrix, whose columns are vectors $f(k), k = 1, \ldots, L$ with coordinates

$$f(a, k) = n_{ak}/r$$

where $n_{ak}$ is equal to the frequency of symbol $a \in \Sigma'$ at position $k$.

Note that again the independence of the columns is assumed. Profiles defined as above are often called

- Position-specific weight matrix (PSWM)

or

- Position-specific scoring matrix (PSSM)

when no gaps in the multiple alignment occur. (In fact for PSSMs gaps are not permitted).

**Example:**

```
A T A A G C
A - A T G C
A - A A G C
```

Then the profile/PSWM/PSSM is

$$
\begin{array}{cccccc}
 & p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \\
A & 1.0 & 0.0 & 1.0 & 0.6 & 0.0 & 0.0 \\
C & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\
G & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\
T & 0.0 & 0.3 & 0.0 & 0.3 & 0.0 & 0.0 \\
- & 0.0 & 0.6 & 0.0 & 0.0 & 0.0 & 0.0 \\
\end{array}
$$

The frequency vectors are often converted either into *propensity* values:

$$w(a, k) = f(a, k)/q_a$$

or into log-odds-ratio values:

$$w(a, k) = \log_2(f(a, k)/q_a)$$
\( q_a = \text{frequency of } a \text{ in the whole alignment or precalculated background frequency.} \)

Problem with this definition: \( \log(0) \)

To avoid this one can use \( \log(1) \).

Example: the PSWM with \( q(A) = q(T) = q(G) = q(C) = 1/4, q(\text{--}) = 1/4 \) of the MSA

\[
\begin{aligned}
\text{A} & \quad \text{T} & \quad \text{A} & \quad \text{A} & \quad \text{G} & \quad \text{C} \\
\text{A} & \quad \text{--} & \quad \text{A} & \quad \text{T} & \quad \text{G} & \quad \text{C} \\
\text{A} & \quad \text{--} & \quad \text{A} & \quad \text{A} & \quad \text{G} & \quad \text{C} \\
\end{aligned}
\]

\[
\begin{align*}
p_1 & \quad p_2 & \quad p_3 & \quad p_4 & \quad p_5 & \quad p_6 \\
1 & \quad 2 & \quad 3 & \quad 4 & \quad 5 & \quad 6 \\
\text{A} & \quad 4.0 & \quad 0.0 & \quad 4.0 & \quad 8/3 & \quad 0.0 & \quad 0.0 \\
\text{C} & \quad 0.0 & \quad 0.0 & \quad 0.0 & \quad 0.0 & \quad 0.0 & \quad 4.0 \\
\text{G} & \quad 0.0 & \quad 0.0 & \quad 0.0 & \quad 0.0 & \quad 4.0 & \quad 0.0 \\
\text{T} & \quad 0.0 & \quad 4/3 & \quad 0.0 & \quad 4/3 & \quad 0.0 & \quad 0.0 \\
\text{--} & \quad 0.0 & \quad 8/3 & \quad 0.0 & \quad 0.0 & \quad 0.0 & \quad 0.0 \\
\end{align*}
\]

Another possibility is to use a scoring matrix \( S \) and compute an average score

\[
w(a, k) = \sum_{b \in \Sigma'} S(a, b)f(a, k)
\]

Example:

### 3.19.1 TransFac

The TransFac database hosts eukaryotic transcription factors, their target genes and regulatory binding sites. Binding sites are represented in form of PSSM.

Example:
The weight matrix for the TF HNF-4 (hepatic nuclear faktor 4) analyzing 32 binding sites in 24 genes:

\[
\begin{array}{cccccccccccccc}
1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 \\
\text{A} & 0.31 & 0.06 & 0.16 & 0.09 & 0.03 & 0.84 & 0.91 & 0.81 & 0.09 & 0.09 & 0.06 & 0.06 \\
\text{C} & 0.09 & 0 & 0.06 & 0.25 & 0.72 & 0.03 & 0 & 0 & 0 & 0.09 & 0.09 & 0.06 \\
\text{G} & 0.43 & 0.91 & 0.53 & 0.32 & 0.03 & 0.09 & 0.09 & 0.16 & 0.88 & 0.5 & 0.19 & 0.03 & 0.12 \\
\text{T} & 0.09 & 0.03 & 0.25 & 0.34 & 0.22 & 0.03 & 0 & 0 & 0.03 & 0.38 & 0.56 & 0.22 & 0.06 \\
\end{array}
\]

### 3.19.2 SeqLogo

Profiles can be used for alignments (see below), but are hard to look at, ie. it is not easy to quickly see the conservation of positions etc.

Profiles can be visualized using sequence logos\(^8\). For each position in the alignment the characters are placed upon each other, the height of each letter is proportional to its frequency. The letter with the largest frequency appears at the top. The total height of the columns are adapted to the information content.

### 3.19.3 Information content

Uncertainty measure according to Shannon:

Suppose there are $n$ mutually exclusive events $x_i$ that occurs with probability $p(x_i)$, then clearly $\sum_i p(x_i) = 1$.

Shannon’s entropy $H$ measures the uncertainty of the set of all possible events

$$H(X) = - \sum_i p(x_i) \log_2 p(x_i)$$

Thus $H(X) = 0$ indicates that with certainty 1 one event will occur, and therefore the uncertainty is 0. If all events are equally likely, ie. $p(x_i) = 1/n$, then $H(X) = \log_2(n)$, and thus takes the maximum possible value.

Exercise: what is $H_{max}$ for DNA and/or amino acid alphabets? From a profile we now compute the entropy at position $k$:

$$H(k) = - \sum_{b \in \Sigma} f(b, k) \log_2 f(b, k) \quad \text{(bits per position)}$$

Conservation (Gain of information) of a column is then

$$C(k) = \max(H) - H(k)$$

This is measured in bits. Example: 4 bases = 2 bits needed.

Large variability $\Rightarrow$ Large entropy $\Rightarrow$ Low conservation

No variability $\Rightarrow$ Low entropy $\Rightarrow$ Large conservation

## 3.19.4 SeqLogo

Height of a column is equal to $C(k)$.

Height of a letter is the relative amount in $C(k)$: $h(b, k) = f(b, k)C(k)$

The following figure shows the sequence logo for the profile matrix of HNF-4.

![SeqLogo](image)

## 3.19.5 Profile alignment

Alignment of a sequence $B$ with a profile sequence $P(A^*)$ works similarly to local alignment methods.

One goal of profile analysis is to use an MSA of a family of related sequences, build a profile of it and then search a database for more members of the family.
Here we turn again shortly to the question how this search is done. We will see that again a modification of the local pairwise alignment method, and thus a dynamic programming algorithm, applies.

Suppose we are given a profile sequence \( P(A^*) = p_1 p_2 \ldots p_L \) where \( p_j = (p_{1j}, p_{2j}, \ldots, p_{|\Sigma'|j}) \) and a sequence \( B = b_1 \ldots b_m \). In addition we have a scoring function \( s : \Sigma' \times \Sigma' \to \mathbb{R} \). Then the similarity of a letter \( b_i \) with the profile at position \( j \) is defined as

\[
\hat{s}(p_j, b_i) = \sum_{a \in \Sigma'} s(a, b_i)p_{aj}
\]

Now we adapt the overlap dynamic programming algorithm that finds the best similarity fit of the profile \( P(A^*) \) into the sequence \( B \). We assume for the following a linear gap penalty \( d \).

Set \( F(0, j) = 0, 0 \leq j \leq m \), and \( F(i, 0) = -id, 1 \leq i \leq L \). Then

\[
F(i, j) = \max\{F(i-1, j-1) + \hat{s}(p_i, b_j), F(i, j-1) - d, F(i-1, j) - d\}
\]

This gives the best alignment with score

\[
F(P, B) = \max\{F(L, j) : 1 \leq j \leq m\}
\]

Example:

\[
\begin{array}{ccccccc}
  p_1 & p_2 & p_3 & p_4 & p_5 & p_6 \\
  1 & 2 & 3 & 4 & 5 & 6 \\
  A & 1.0 & 0.0 & 1.0 & 0.6 & 0.0 & 0.0 \\
  C & 0.0 & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 \\
  G & 0.0 & 0.0 & 0.0 & 0.0 & 1.0 & 0.0 \\
  T & 0.0 & 0.3 & 0.0 & 0.3 & 0.0 & 0.0 \\
  - & 0.0 & 0.6 & 0.0 & 0.0 & 0.0 & 0.0 \\
  \end{array}
\]

\( B = \) GGAATGCAG,

the scoring function \( s(a, b) = -1, s(a, a) = +2, \) and \( s(a, -) = -1 \)

Compute the entries of the matrix \( F \), and search for the cell on the bottom row with largest score, from which the traceback is started.

First compute the (profile,Sequence)-scoring matrix \( \hat{s}(p, B) \):

\[
\hat{s}(p_j, b_i) = \sum_{a \in \Sigma'} s(a, b_i)p_{aj}.
\]

For example

\[
\hat{s}(p_1, b_1) = \sum_{a \in \Sigma'} s(a, G)p_{a1} = s(A, G) \cdot 1 + s(G, G) \cdot 0 + \ldots + s(\cdot, G) \cdot 0 = -1.
\]

\[
\hat{s}(p_2, b_2) = s(T, G) \cdot 0.3 + s(\cdot, G) \cdot 0.6 = -1 \text{ and so on.}
\]

\[
\begin{array}{ccccccc}
  b_1 & b_2 & b_3 & b_4 & b_5 & b_6 & b_7 & b_8 & b_9 \\
  G & G & A & A & T & G & C & A & G \\
  p_1 & -1 & -1 & 2 & 2 & -1 & -1 & -1 & -1 \\
  p_2 & -1 & -1 & -1 & -1 & 0 & -1 & -1 & -1 \\
  p_3 & -1 & -1 & 2 & 2 & -1 & -1 & -1 & -1 \\
  p_4 & -1 & -1 & 1 & 1 & 0 & -1 & -1 & 1 \\
  p_5 & 2 & 2 & -1 & -1 & 2 & -1 & -1 & 2 \\
  p_6 & -1 & -1 & -1 & -1 & -1 & 2 & -1 & -1 \\
  \end{array}
\]
Next we have to fill the following matrix $F$:

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>G</th>
<th>C</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$p_1$</td>
<td>−1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_2$</td>
<td>−2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_3$</td>
<td>−3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_4$</td>
<td>−4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_5$</td>
<td>−5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_6$</td>
<td>−6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

which results in the following local profile-sequence alignment:

The resulting matrix is then

<table>
<thead>
<tr>
<th></th>
<th>O</th>
<th>G</th>
<th>G</th>
<th>A</th>
<th>A</th>
<th>T</th>
<th>G</th>
<th>C</th>
<th>A</th>
<th>G</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>$p_1$</td>
<td>−1</td>
<td>−1</td>
<td>−1</td>
<td>2</td>
<td>2</td>
<td>−1</td>
<td>−1</td>
<td>−1</td>
<td>2</td>
<td>−1</td>
</tr>
<tr>
<td>$p_2$</td>
<td>−2</td>
<td>−2</td>
<td>−2</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>−1</td>
<td>−2</td>
<td>−1</td>
<td>1</td>
</tr>
<tr>
<td>$p_3$</td>
<td>−3</td>
<td>−3</td>
<td>−3</td>
<td>0</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_4$</td>
<td>−4</td>
<td>−4</td>
<td>−4</td>
<td>−1</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_5$</td>
<td>−5</td>
<td>−5</td>
<td>−5</td>
<td>−5</td>
<td>−5</td>
<td>5</td>
<td>4</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$p_6$</td>
<td>−6</td>
<td>−6</td>
<td>−6</td>
<td>−6</td>
<td>−6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>

The local alignment is $A \quad - \quad A \quad T \quad G \quad C$

3.20 PSI-BLAST

A carefully constructed PSSM can be also used to enhance the sensitivity for database searches.

BLAST has been enhanced by incorporating PSSMs, resulting in

PSI-BLAST = Position specific iterated BLAST

used to search for distant protein sequences.

This is how PSI-BLAST works:

1. Start of with a query and blastp against DB.
2. A profile (PSSM) is constructed (automatically) from the multiple alignment of the highest scoring hits in the initial BLAST search
3. BLAST reports on number of newly found subject sequences
4. Repeat 2 until no new sequences are found

3.21 Planted Motif Problem

A much more difficult computational problem is to determine a motif by analyzing a set of sequences that contain possible instances of the motif.

We formalize the problem as follows (Pevzner and Sze):
Planted \((l, d)\)-Motif Problem:: Suppose there is a fixed but unknown nucleotide sequence \(M\) (the motif) of length \(l\). The problem is to determine \(M\), given \(t\) sequences each of length \(n\), and each containing a planted variant of \(M\). More precisely, each such planted variant is a substring of length \(l\) which differs from \(M\) at up to \(d\) positions.

### 3.21.1 Signatures to represent profiles/motifs

Another possibility of representing amino acid profiles is the signature. A signature is a regular expression constructed according to the following rules:

- The IUPAC one-letter code is used for the amino acids.
- The symbol ‘x’ represents any amino acid.
- Brackets ‘[ ]’ are used for displaying ambiguity. Example: [ALT] stands for Ala or Leu or Thr.
- Ambiguities can also be expressed by ‘\{ \}\'; it is the complementary symbol to ‘[ ]’. Example: \{AM\} stands for each amino acid but Ala and Met.
- Each symbol in the signature is separated by an ‘-’ from its neighbor.
- Repeated symbols are represented by the symbol and a bracket in which the repetitions are given. Example: x(3) is equal to to x-x-x.
- The signature ends with a dot.

Example: [AC]-x-V-x(4)-{ED}.

This translates into: [Ala or Cys]-any-Val-any-any-any-nay-{any but not Glu or Asp}

### 3.21.2 DNA motif databases

**Jaspar** - The high-quality transcription factor binding profile database, http://jaspar.genereg.net/


The existing databases of protein motifs differ in their generation methods of the profiles, their search methods against the profiles and their biological annotation.

**PROSITE** Representation: Signatures

**PFAM, SMART** Representation: Hidden Markov Models (HMMs)
Curated local MSAs

**Conserved Domain Database (CDD)** Representation: Position specific scoring matrices (PSSMs)
Structurally corrected local MSAs

**Many more** see internet
3.21.3 PROSITE

PROSITE, the “dictionary of sites and patterns in proteins”, is one of the most important collections of patterns.

It consists of a database of biologically significant sites and patterns formulated in such a way that with appropriate computational tools it can rapidly and reliably identify to which known family of protein (if any) the new sequence belongs. (from the ProSite User manual).

PROSITE is hosted at the Expasy (Expert Protein Analysis System) Server of the Swiss Institute of Bioinformatics.

The URL is: http://www.expasy.org/prosite/

The ScanProsite tool allows to scan protein sequence(s) for the occurrence of patterns, profiles and rules (motifs) stored in the PROSITE database.

3.21.4 Example: PROSITE

Squash family of serine protease inhibitors signature:
Consensus pattern: C-P-x(5)-C-x(2)-[DN]-x-D-C-x(3)-C-x-C

(PDOC00258)
(PS00286; )
(REGIR)
******************************************************************************
* Squash family of serine protease inhibitors signature *
******************************************************************************
The squash family of serine protease inhibitor [1] is one of the numerous families of serine proteinase inhibitors. ... The basic structure of such a type of inhibitor is shown in the following schematic representation:

```
\[ C^\circ \] conserved cysteine involved in a disulfide bond.
\#^\circ \] active site residue.
*^\circ \] position of the pattern.
... Consensus pattern: C-P-x(5)-C-x(2)-[DN]-x-D-C-x(3)-C-x-C
(EDR)
```

3.21.5 BLOCKS:

A block is a multiple alignment (without gaps) of a highly conserved region of a group of related proteins. The blocks in the BLOCKS database are generated from groups of proteins that are defined in the PROSITE database, though the signature is not used. Blocks are also generated even when no biological function of the motif is known. It is solely sufficient to have a certain degree of conservation. The BLOCKS database is the basic data collection for the generation of the BLOSUM matrices.

http://blocks.fhcrc.org/

3.21.6 PFAM (Protein Family Database)

Pfam is a large collection of multiple sequence alignments and hidden Markov models covering many common protein domains and families. Pfam can be used to view the domain organisation of proteins.

URL: http://pfam.sanger.ac.uk/

Pfam contains two types of families, Pfam-A and Pfam-B. Pfam-A families are manually curated HMM based families. To give Pfam a more comprehensive coverage of known proteins a supplement called Pfam-B is generated. This contains a large number of small families taken from the PRODOM database (see below) that do not overlap with Pfam-A.

Each dataset in Pfam-A consists of 4 parts: the annotation, the seed alignment, a profile HMM and a full alignment. The seed alignment consists of an alignment of representative members of the protein family. The profile HMM is derived from the seed alignment. Using the profile HMM protein databases are searched for sequences exceeding a specific score. From all these sequences the full alignment of the protein family is generated.

Example: Pfam alignment of the trypsin inhibitor signature:

![Seed sequence alignment for PF00299](image)

Alignments can be visually represented by an adapted Sequence Logo:

![Sequence Logo](image)

Recognition: search based on BLAST.

Result: if unknown sequences contains any domains of Pfam, how many domains, where they lie in the sequence and annotation about the domain.
Example:

>Unknown
RVCPRLMECKKDSDCLEACVCHELHGYCG

Result:

Alignment of Squash vs UNKNOWN-QUERY/1-29

```
*->rICPrlmeCkrDClaeCvCleegeyCG<--*
  r+CPrlmeCk+DClaeCvCle+g yCG
UNKNOWN-QU  1  RVCPRLMECKKDSDCLEACVCHELHGYCG  29
```

3.21.7 ProDom:

ProDom is a comprehensive set of protein domain families automatically generated from the SWISS-PROT and TrEMBL sequence databases. Here one finds both multiple sequence alignments of protein families as well as corresponding phylogenetic trees. The profiles have been generated with PSI-BLAST.
