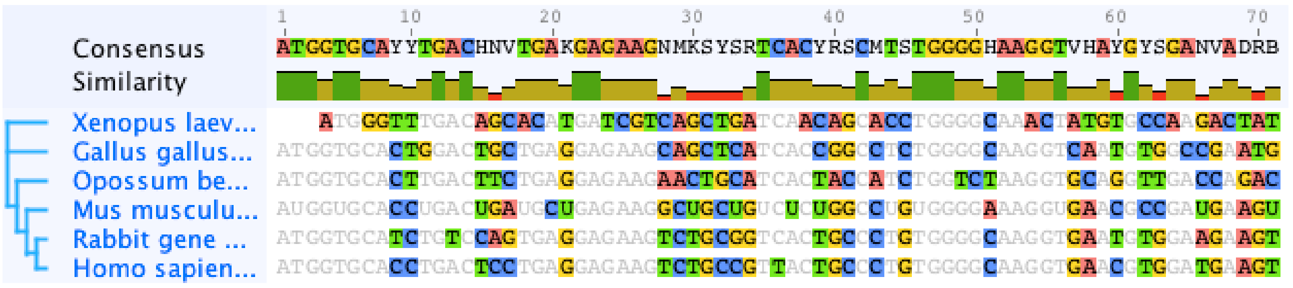
7 Multiple Sequence Alignment

The exposition was prepared by Clemens Gröpl, based on earlier versions by Daniel Huson, Knut Reinert, and Gunnar Klau. It is based on the following sources, which are all recommended reading:

1. R. Durbin, S. Eddy, A. Krogh und G. Mitchison: Biological sequence analysis, Cambridge, 1998
2. Gusfield: Algorithms on Strings, Trees, and Sequences, Cambridge University Press, 1997, chapter 14.
3. J. Stoye, P. Husemann, E. Willing, S. Rahmann, R. Giegerich, S. Kurtz, E. Ohlebusch et al.: Sequence Analysis I + II, lecture notes for WS 2009/10 and SS 2010, Universität Bielefeld, `http://wiki.techfak.uni-bielefeld.de/gi/GILectures/2009winter/SequenzAnalyse`

7.1 Introduction

What is a multiple sequence alignment? A multiple sequence alignment is an alignment of more than two sequences:



```
                   1        10        20        30        40        50        60        70
Consensus          ATGGTGCAYYTGACHNVTGAKGAGAAGNMKSYSRTCACYRSCMTSTGGGGHAAGGTVHAYGYSGANVADRB
Similarity
Xenopus laev...       ATGGGTTTGACAGCACATGATCGTCAGCTGATCAACAGCACCTGGGGCAAACTATGTGCCAAGACTAT
Gallus gallus...   ATGGTGCACTGGACTGCTGAGGAGAAGCAGCTCATCACCGGCCTCTGGGGCAAGGTCAATGTGGCCGAATG
Opossum be...      ATGGTGCACTTGACTTCTGAGGAGAAGAACTGCATCACTACCATCTGGTCTAAGGTGCAGGTTGACCAGAC
Mus musculu...     AUGGUGCACCUGACUGAUGCUGAGAAGGCUGCUGUCUCUGGCCUGUGGGGAAAGGUGAACGCCGAUGAAGU
Rabbit gene ...    ATGGTGCATCTGTCCAGTGAGGAGAAGTCTGCGGTCACTGCCCTGTGGGGCAAGGTGAATGTGGAAGAAGT
Homo sapien...     ATGGTGCACCTGACTCCTGAGGAGAAGTCTGCCGTTACTGCCCTGTGGGGCAAGGTGAACGTGGATGAAGT
```

[Sequences are truncated. Screenshot from http://www.geneious.com/]

In this example multiple sequence alignment is applied to a set of sequences that are assumed to be homologous (i. e., having a common ancestor). Homologous nucleotides are placed in the same column of a multiple alignment. Note the phylogenetic tree attached to the left.

While multiple sequence alignment (MSA) is natural generalization of pairwise sequence alignment, there are lots of new questions and issues concerning the scoring function, the significance of scores, the gap penalties, and efficient implementations.

Definitions.
Assume we are given k sequences $x_1, \ldots, x_k$ over an alphabet Σ.

Let - $\notin \Sigma$ be the *gap symbol*. Let $h\colon (\Sigma \cup \{-\})^* \to \Sigma^*$ be the *projection* mapping that removes all gap symbols from a sequence over the alphabet $\Sigma \cup \{-\}$. For example, $h($`-FPIKWTAPEAALY---GRFT`$) =$ `FPIKWTAPEAALYGRFT`.

Then a *global alignment* of $x_1, \ldots, x_k$ consists of k sequences $x'_1, \ldots, x'_k$ over the alphabet $\Sigma \cup \{-\}$ such that

- $h(x'_i) = x_i$ for all i,
- $|x'_i| = |x'_j|$ for all i, j, and
- $(x'_{1,p}, \ldots, x'_{k,p}) \neq (-, \ldots, -)$ for all p.

The concept of *projections* can be extended to subsets of sequences. We remove columns that consist of gap symbols only:

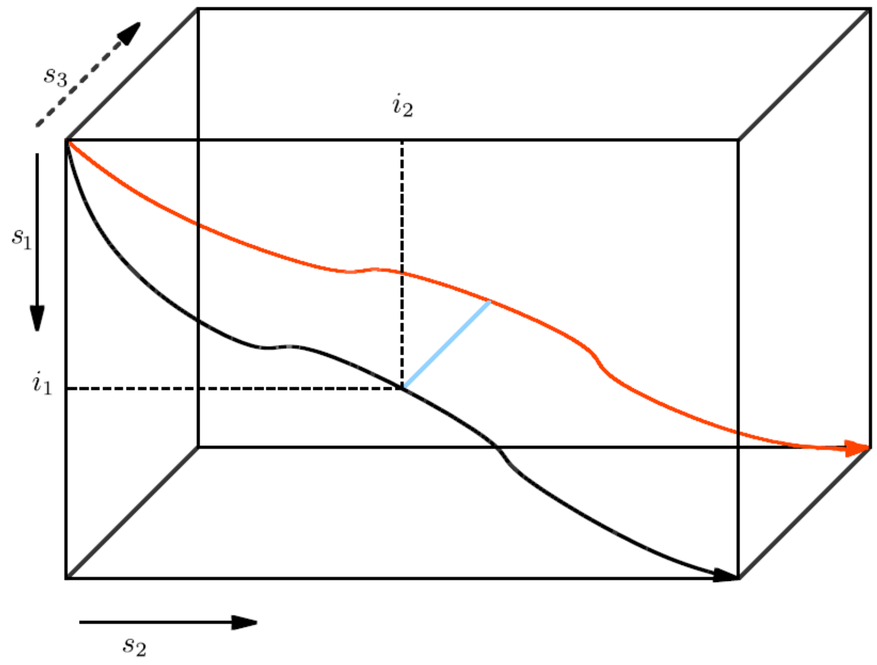
$$A = \begin{pmatrix} \texttt{-} & \texttt{A} & \texttt{C} & \texttt{C} & \texttt{-} & \texttt{-} & \texttt{A} & \texttt{T} & \texttt{G} \\ \texttt{-} & \texttt{A} & \texttt{-} & \texttt{C} & \texttt{G} & \texttt{A} & \texttt{A} & \texttt{T} & \texttt{-} \\ \texttt{T} & \texttt{A} & \texttt{C} & \texttt{C} & \texttt{-} & \texttt{-} & \texttt{A} & \texttt{G} & \texttt{G} \\ \texttt{-} & \texttt{A} & \texttt{-} & \texttt{C} & \texttt{C} & \texttt{A} & \texttt{A} & \texttt{T} & \texttt{G} \end{pmatrix}$$

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

$$\pi_{\{1,3\}}(A) = \begin{pmatrix} \texttt{-} & \texttt{A} & \texttt{C} & \texttt{C} & \texttt{A} & \texttt{T} & \texttt{G} \\ \texttt{T} & \texttt{A} & \texttt{C} & \texttt{C} & \texttt{A} & \texttt{G} & \texttt{G} \end{pmatrix}$$

$$\pi_{\{3,4\}}(A) = \begin{pmatrix} \texttt{T} & \texttt{A} & \texttt{C} & \texttt{C} & \texttt{-} & \texttt{-} & \texttt{A} & \texttt{G} & \texttt{G} \\ \texttt{-} & \texttt{A} & \texttt{-} & \texttt{C} & \texttt{C} & \texttt{A} & \texttt{A} & \texttt{T} & \texttt{G} \end{pmatrix}$$

Another view at the concept of alignment projection:



Projection of a multiple alignment of three sequences s_1, s_2, s_3 on s_1 and s_2.

An incomplete list of the many uses of multiple sequence alignment:

- detecting faint similarities in sequences that are not detected by pairwise sequence comparison.
- detecting structural homologies.
- grouping proteins into families.
- computing the consensus sequence in assembly projects.
- inferring evolutionary trees.
- and more ...

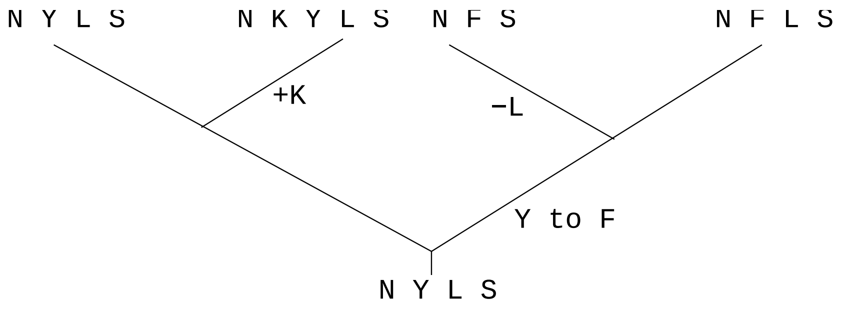
We now give some more details about the different uses.

One or two homologous sequences whisper ... a full multiple alignment shouts out loud. [1]

7.2 MSA and evolutionary trees

One main application of multiple sequence alignment lies in phylogenetic analysis. Given an MSA, we would like to reconstruct the evolutionary tree that gave rise to these sequences, e.g.:

$a_1 =$	N	-	F	L	S
$a_2 =$	N	-	F	-	S
$a_3 =$	N	K	Y	L	S
$a_4 =$	N	-	Y	L	S



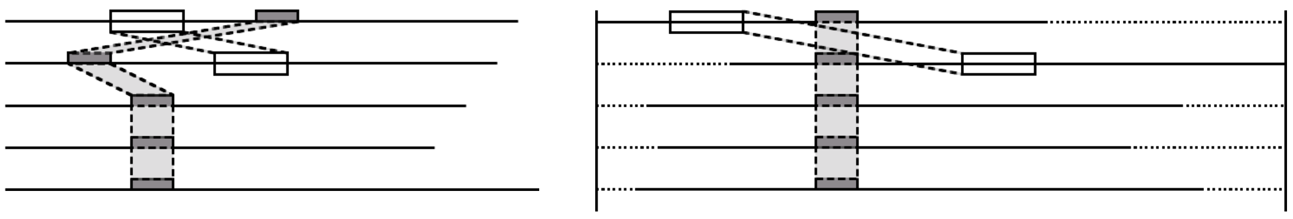
We need to find out how the positions of the sequences correspond to each other. The problem of computing phylogenetic trees (for each column) will be discussed later.

7.3 Protein families

Assume we have established a family $s_1, s_2, \dots, s_r$ of homologous protein sequences. Does a new sequence s_0 belong to the family?

One method of answering this question would be to align s_0 to each of $s_1, \dots, s_r$ in turn. If one of these alignments produces a high score, then we may decide that s_0 belongs to the family.

However, perhaps s_0 does not align particularly well to any one specific family member, but does well in a multiple alignment, due to common motifs etc.



7.4 Sequence Assembly

Assume we are given a layout of several genomic reads in a sequencing project that were produced using the shotgun sequencing method. These fragments will be highly similar, and hence easy to align. Nevertheless we

[1] T. J. P. Hubbard, A. M. Lesk, and A. Tramontano. Gathering them into the fold. Nature Structural Biology, 4:313, 1996.

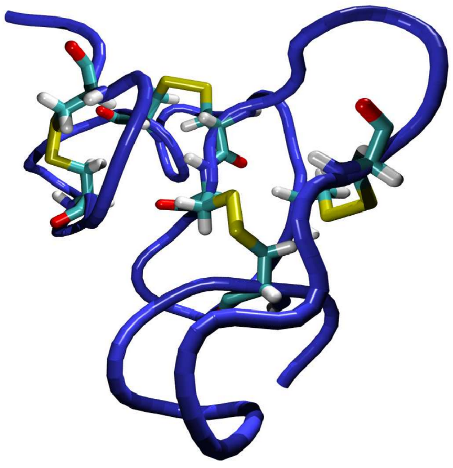
want to do this with great speed and accuracy:

```
f1  ACCACAACCCTGCATGGGGCAT-ATTTGGCCTAGCT
f2                 AGGGCCTTATATG-GCTAGCT-CGTTCCCGGGCATGGC
f3             GCATGGGGCATTATCTGGCCTAGCT--GAT
f4                                      CCGTTCCCGG-CTTGGCAACG
=============================================================
cns ACCACAACCCTGCATGGGGCATTATCTGGCCTAGCT-CGTTCCCGGGCATGGCAACG
```

7.5 Conservation of structural elements

The below figure shows the alignment of N-acetylglucosamine-binding proteins and the tertiary structure of one of them, the hevein.

```
AATAHAQRCG EQGSNMECPN NLCCSQYGYC GMGGDYCGKG ..CQNGACYT
VAATNAQTCG KQNDGMICPH NLCCSQFGYC GLGRDYCGTG ..CQSGACCS
VGLVSAQRCG SQGGGGTCPA LWCCSIWGWC GDSEPYCGRT ..CENK.CWS
AATAQAQRCG EQGSNMECPN NLCCSQYGYC GMGGDYCGKG ..CQNGACWT
AATAQAQRCG EQGSNMECPN NLCCSQYGYC GMGGDYCGKG ..CQNGACWT
......QRCG EQGSGMECPN NLCCSQYGYC GMGGDYCGKG ..CQNGACWT
SETVKSQNCG .......CAP NLCCSQFGYC GSTDAYCGTG ..CRSGPCRS
RGSAE..QCG RQAGDALCPG GLCCSSYGWC GTTVDYCGIG ..CQSQ.CDG
AGPAAAQNCG .......CQP NFCCSKFGYC GTTDAYCGDG ..CQSGPCRS
AGPAAAQNCG .......CQP NVCCSKFGYC GTTDEYCGDG ..CQSGPCRS
RGSAE..QCG QQAGDALCPG GLCCSSYGWC GTTADYCGDG ..CQSQ.CDG
RGSAE..QCG RQAGDALCPG GLCCSFYGWC GTTVDYCGDG ..CQSQ.CDG
TGVAIAEQCG RQAGGKLCPN NLCCSQWGWC GSTDEYCSPD HNCQSN.CK.
......EQCG RQAGGKLCPN NLCCSQYGWC GSSDDYCSPS KNCQSN.CK.
```



The example exhibits 8 cysteins that form 4 disulphid bridges (shown yellow in the 3d structure) and are an essential structural part of those proteins.

7.6 The dynamic programming algorithm for MSA

Although local alignments are biologically more relevant, it is easier to discuss global MSA. Dynamic programming algorithms developed for pairwise alignment can be modified to MSA. For concreteness, let us discuss how to compute a global MSA for three sequences, in the case of a linear gap penalty. Given:

$$A = \begin{cases} a_1 = & (a_{1,1}, a_{1,2}, \ldots, a_{1,n_1}) \\ a_2 = & (a_{2,1}, a_{2,2}, \ldots, a_{2,n_2}) \\ a_3 = & (a_{3,1}, a_{3,2}, \ldots, a_{3,n_3}). \end{cases}$$

We proceed by computing the entries of a $(n_1 + 1) \times (n_2 + 1) \times (n_3 + 1)$-"'matrix"' $F(i, j, k)$. Actually F should be called a *multidimensional array*, or tensor.

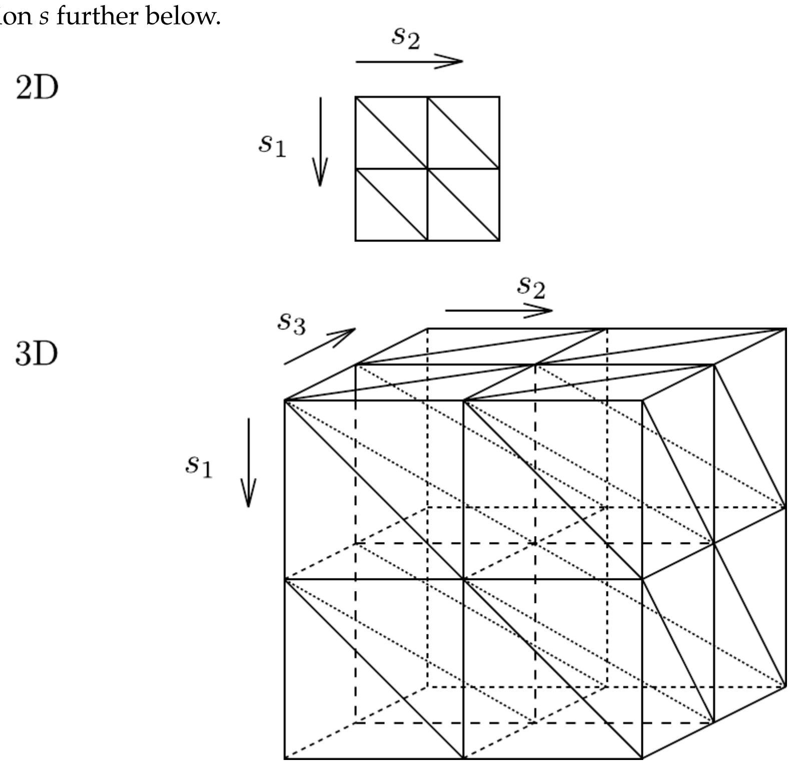
After the computation, $F(n_1, n_2, n_3)$ will contain the best score α for a global alignment A^*. As in the pairwise case, we can use traceback to recover an actual alignment.

The main recursion is:

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

$$\text{for } 1 \le i \le n_1,\ 1 \le j \le n_2,\ 1 \le k \le n_3,$$

where $s(x,y,z)$ returns a score for a given column of symbols $x, y, z \in \Sigma \cup \{-\}$. We will discuss reasonable choices for the function s further below.



Clearly, this algorithm generalizes to r sequences.

However, the time complexity is $O(n_1 \cdot n_2 \cdot \ldots \cdot n_r)$ Hence, it is only practical for small numbers of short sequences, say $r = 5$ or 6.

2

The initialization of the boundary cells will be discussed in the exercises.

[2]Computing an MSA with optimal sum-of-pairs-score has been proved to be NP-complete (so, most likely, no polynomial time algorithm exists that solves this problem).

7.7 WSOP score

One of the most common scoring functions for MSA is the *(weighted) sum of pairs*, which is defined as the (weighted) sum of the score of all pairwise projections of the MSA, that is,

$$c(A) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} w_{i,j} \cdot c(\pi_{i,j}(A))$$

Each pair of sequences (i, j) can be given a different *weight* $w_{i,j}$. Note that $c(\pi_{i,j}(A))$ involves another summation over the columns of $\pi_{i,j}(A)$. However, if we assume linear gap costs, then we can rewrite this as follows:

$$c(A) = \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{h=1}^{l} w_{i,j} \cdot s(a_{i,h}, a_{j,h}) = \sum_{h=1}^{l} \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} w_{i,j} \cdot s(a_{i,h}, a_{j,h})$$

The nice thing about WSOP is that we can move the "inner" summation (running over the column index h) "outside", to the front.

In the case $k = 3$, the DP algorithm mentioned above would use

$$s(x, y, z) = w_{1,2}s(x, y) + w_{1,3}s(x, z) + w_{2,3}s(y, z)\,.$$

Example.

$$\text{Let } s(x, y) = \begin{cases} 3 & \text{for } x = y \text{ (match)} \\ -2 & \text{for } x \neq y, (x, y) \in \Sigma \times \Sigma \text{ (mismatch)} \\ 0 & \text{for } x = y = '-' \text{ (such columns are eliminated in projection)} \\ -1 & \text{otherwise (gaps)}\ . \end{cases}$$

All the weight factors are 1.

```
            a1 = -  G  C  T  G  A  T  A  T  A  A  C  T
            a2 = G  G  G  T  G  A  T  -  T  A  G  C  T
            a3 = A  G  C  G  G  A  -  A  C  A  C  C  T
                 -----------------------------------
score of column: -4  9 -1 -1  9  9  1  1 -1  9 -6  9  9 = 43
```

The sum of pair score takes all pairwise information into account; however it is easily biased if sequences from the same family are over-representated in the input. This disadvantage has to be dealt with by choosing the weights accordingly.

7.8 Scoring Schemes

At present, there is no conclusive argument that gives any one scoring scheme more justification than the others. The sum-of-pairs score is widely used, but is problematic.

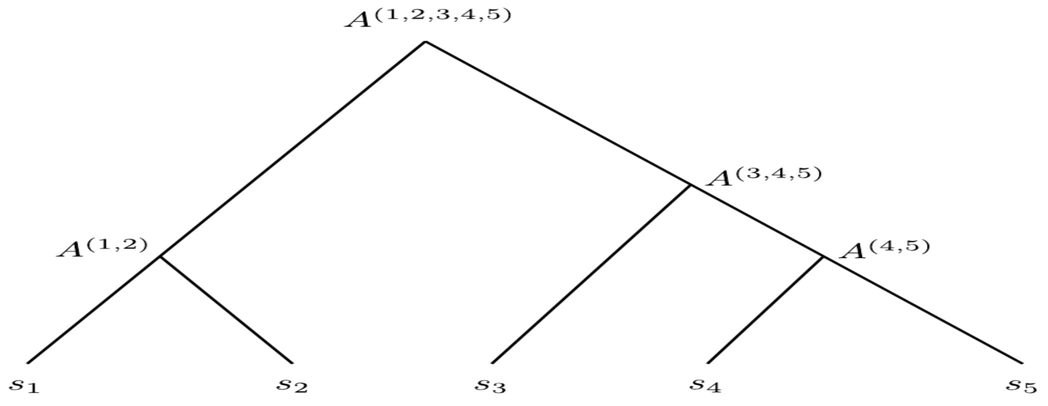
One advantage of the WSOP score (or cost) function is the ease with which gaps are modelled, since the score reduces to the summation of the pairwise scores, for which the handling of gaps is well understood.

7.9 Progressive alignment

The idea of progressive multiple sequence alignment is to compute a multiple alignment in a bottom-up fashion, starting with pairwise alignment and then combining them along a *guide tree*. The most popular multiple alignment programs follow this strategy.

In its simplest fashion, the given sequences a_i are added one after another to the growing multiple alignment, i.e. first a_1 and a_2 are aligned, then a_3 is added (we will see below how “adding” is actually done), then a_4, and so forth.

Ideally, the phylogenetic tree should be known and be used as the guide tree. In practice, we may have to use heuristic guide trees. Note that multiple alignments are often used to infer phylogenetic trees; both depend upon each other.



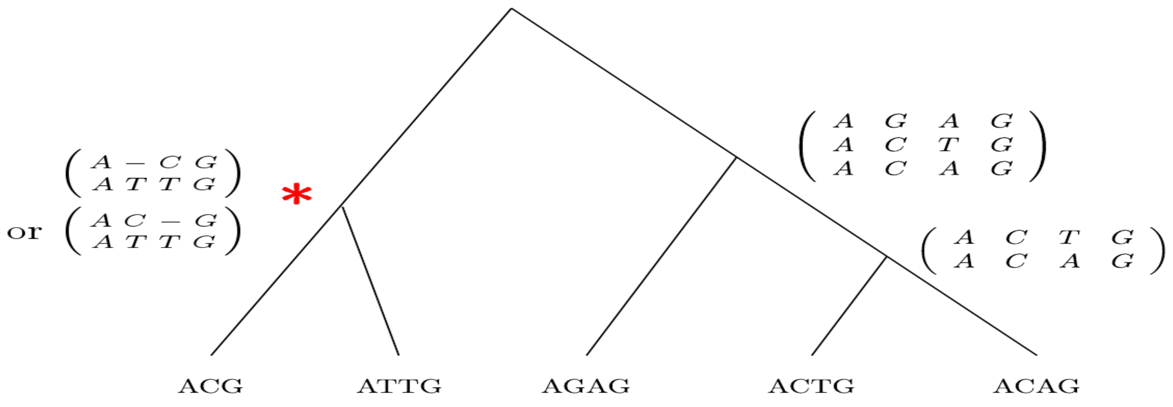
Progressive alignment along the branches of a phylogenetic tree. Sequences s_1 and s_2 are aligned, giving alignment $A^{(1,2)}$. Sequences s_4 and s_5 give $a^{(4,5)}$ which then is aligned with s_3 giving $A^{(3,4,5)}$. Finally, aligning $A^{(1,2)}$ and $A^{(3,4,5)}$ gives the multipe alignment of all sequences, $A^{(1,2,3,4,5)}$.

The main advantage of progressive methods is speed: Only pairwise alignments are computed.

A potential disadvantage is the “procedural” definiton: There is no well-founded global objective function being optimized. It may be difficult to evaluate the quality of the results.

The main disadvantage is of course the dependency on a guide tree.

The strict bottom-up progressive computation can also lead to problems. In the example below, it cannot be decided at the time of computing the alignment at the node marked with the asterisk (∗) which of the two alternatives is better. Only the rest of the tree indicates that the second one is probably the correct one.


$\begin{pmatrix} A & - & C & G \\ A & T & T & G \end{pmatrix}$
or $\begin{pmatrix} A & C & - & G \\ A & T & T & G \end{pmatrix}$
*
$\begin{pmatrix} A & G & A & G \\ A & C & T & G \\ A & C & A & G \end{pmatrix}$
$\begin{pmatrix} A & C & T & G \\ A & C & A & G \end{pmatrix}$
ACG
ATTG
AGAG
ACTG
ACAG

7.10 Aligning alignments

An important subtask in progressive alignment is to align two existing multiple alignments. This is a *pairwise* alignment problem, but instead of simple letters we now have alignment columns to align.

We need the equivalent of a scoring matrix for alignment columns. One way to define such an extended score is to add the scores for all pairs of letters between the two columns.

For example the score for the two columns

$$A_{\cdot,i} = \begin{pmatrix} A \\ C \\ - \end{pmatrix} \quad \text{and} \quad \begin{pmatrix} G \\ T \end{pmatrix} = B_{\cdot,j}$$

would be

$$s(A_{\cdot,i}, B_{\cdot,j}) := \sum_{\alpha} \sum_{\beta} s(A_{\alpha,i}, B_{\beta,j})\,,$$

in the above example, $s(A,G) + s(A,T) + s(C,G) + s(C,T) + s(-,G) + s(-,T)$.

If the alignments are "deep" compared to the alphabet size, one can store the frequencies (= number of occurrences) of the letters in a table of size $|\Sigma|$ and replace the iteration over the rows of the columns by an iteration over the alphabet. This is also called *profile alignment*.

The following example illustrates the procedure for the alignment of

$$A = \begin{pmatrix} T & A & G \\ G & - & C \end{pmatrix} \quad \text{and} \quad B = \begin{pmatrix} A & T & C & A & G \\ A & G & C & - & G \end{pmatrix}$$

using the unit cost model.

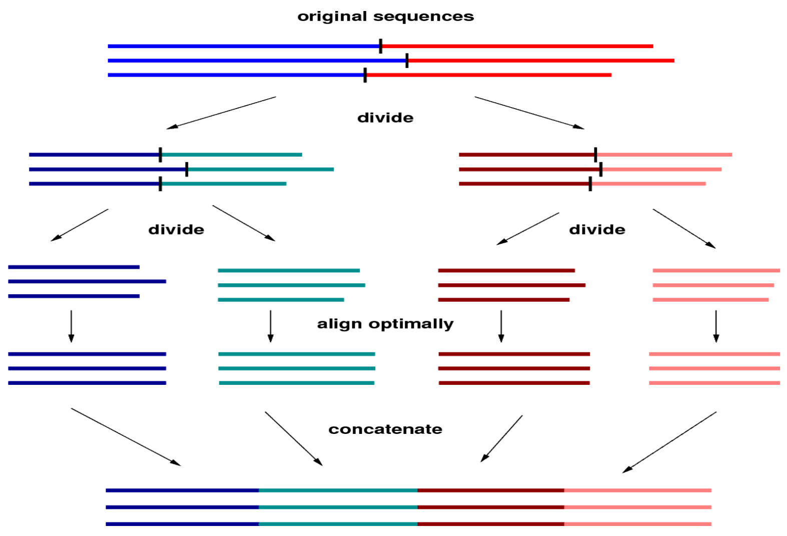
	$\begin{smallmatrix}-\\-\end{smallmatrix}$	$\begin{smallmatrix}A\\A\end{smallmatrix}$	$\begin{smallmatrix}T\\G\end{smallmatrix}$	$\begin{smallmatrix}C\\C\end{smallmatrix}$	$\begin{smallmatrix}A\\-\end{smallmatrix}$	$\begin{smallmatrix}G\\G\end{smallmatrix}$
$\begin{smallmatrix}-\\-\end{smallmatrix}$	0	4	8	12	14	18
$\begin{smallmatrix}T\\G\end{smallmatrix}$	4	4	6	10	12	16
$\begin{smallmatrix}A\\-\end{smallmatrix}$	6	6	8	...		
$\begin{smallmatrix}G\\C\end{smallmatrix}$	10					

7.11 Divide-and-conquer alignment

Another heuristic approach is to divide all sequences at suitable cut points into left and right parts, and then to solve the two subproblems separately using either recursion or, if the sequences are short enough, an exact algorithm.

Of course the main question is how to choose good cut positions. We will use the notation ++ for the concatenation of alignments.

Overview of divide-and-conquer multiple sequence alignment:



Algorithm 13.1 $DCA(s_1, s_2, \ldots, s_k; L)$

```
1: Let $n_p \leftarrow |s_p|$ for all $p$, $1 \le p \le k$
2: if $\max\{n_1, n_2, \ldots, n_k\} \le L$ then
3:     return $MSA(s_1, s_2, \ldots, s_k)$
4: else
5:     $\hat{c}_1 \leftarrow \lceil n_1/2 \rceil$
6:     compute $(c_2, \ldots, c_k)$ such that $(\hat{c}_1, c_2, \ldots, c_k)$ is a $C$-optimal cut
7:     return $DCA(s_1[1..\hat{c}_1], s_2[1..c_2], \ldots, s_k[1..c_k]; L) +\!\!+ DCA(s_1[\hat{c}_1+1..n_1], s_2[c_2+1..n_2],$
       $\ldots, s_k[c_k+1..n_k]; L)$
```

The cut positions are found heuristically using the so-called score-loss matrices, defined as follows:

Definition 11.2 Given two sequences s and t of lengths m and n, respectively, their **additional cost matrix** $C = (C(i,j))_{0 \le i \le m, 0 \le j \le n}$ is defined by

$$C(i,j) \quad = \quad \min \left\{ D(A +\!\!+ B) \;\middle|\; \begin{array}{l} A \text{ is an alignment of the prefixes} \\ \quad s[1 \ldots i] \text{ and } t[1 \ldots j] \\ B \text{ is an alignment of the suffixes} \\ \quad s[i+1 \ldots m] \text{ and } t[j+1 \ldots n] \end{array} \right\} - d(s,t)$$

where "$+\!\!+$" denotes concatenation of alignments. The **score loss matrix** is defined similarly in terms of maximal score.

The idea is that $C(i,j)$ describes what we lose when the alignment is forced to pass through the (i,j) entry of the DP matrix.

Example (for additional cost):
The sequences are *CT* and *AGT*. We use unit cost. D is the DP matrix for global alignment. Likewise, D^{rev} is the DP matrix computed in reverse direction. Then the additional cost matrix C is optained by adding D and D^{rev} pointwise and subtracting the optimal score (here: 2) from each entry.

$$D: \quad \begin{array}{c|cccc} & & A & G & T \\ \hline & 0 & 1 & 2 & 3 \\ C & 1 & 1 & 2 & 3 \\ T & 2 & 2 & 2 & 2 \end{array} \qquad D^{rev}: \quad \begin{array}{c|cccc} & & A & G & T \\ \hline & 2 & 1 & 1 & 2 \\ C & 2 & 1 & 0 & 1 \\ T & 3 & 2 & 1 & 0 \end{array}$$

$$C: \quad \begin{array}{c|cccc} & & A & G & T \\ \hline & 0 & 0 & 1 & 3 \\ C & 1 & 0 & 0 & 2 \\ T & 3 & 2 & 1 & 0 \end{array}$$

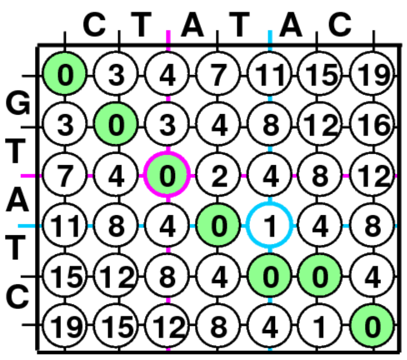
Note that an optimal alignment uses the 0-entries of C only.

		C	T	A	T	A	C
	0	2	4	6	8	10	12
G	2	1	3	5	7	9	11
T	4	3	1	3	5	7	9
A	6	5	3	1	3	5	7
T	8	7	5	3	1	3	5
C	10	8	7	5	3	2	3

Figure 2a

	C	T	A	T	A	C	
G	3	4	3	4	6	8	10
T	4	2	3	2	4	6	8
A	6	4	2	2	2	4	6
T	8	6	4	2	1	2	4
C	10	8	6	4	2	0	2
	12	10	8	6	4	2	0

Figure 2b



0	3	4	7	11	15	19
3	0	3	4	8	12	16
7	4	0	2	4	8	12
11	8	4	0	1	4	8
15	12	8	4	0	0	4
19	15	12	8	4	1	0

Figure 2c

Figures 2a and 2b show the standard dynamic programming distance matrices of the sequences **s** = GTATC and **t** = CTATAC (using unit cost and penalty +2 for each single insertion/deletion), computed from the upper left to the lower right (Fig. 2a) and from the lower right to the upper left (Fig. 2b). Figure 2c displays the *additional–cost* matrix, containing the values $C_{\mathbf{s},\mathbf{t}}[c,d]$ for each pair of slicing positions (c,d), e.g.

$$C_{\mathbf{s},\mathbf{t}}[2,2] = w_{opt}[\text{CT,GT}] + w_{opt}[\text{ATAC,ATC}] - w_{opt}[\text{CTATAC,GTATC}] = 1 + 2 - 3 = 0$$

$$C_{\mathbf{s},\mathbf{t}}[4,3] = w_{opt}[\text{CTAT,GTA}] + w_{opt}[\text{AC,TC}] - w_{opt}[\text{CTATAC,GTATC}] = 3 + 1 - 3 = 1$$

In each matrix, the optimal alignment path is colored green. Its additional–cost matrix entries are, of course, zero.

The **multiple additional cost** of a family of cut positions $(c_1, c_2, \ldots, c_k)$ is defined as

$$C(c_1, c_2, \ldots, c_k) := \sum_{1 \le p < q \le k} C_{(p,q)}(c_p, c_q),$$

where $C_{(p,q)}$ is the additional cost matrix for the pair (p, q).

The DQA tries to minimize the mulitple additional cost when choosing the cut positions. However, simple examples exist showing that an optimal cut need not have multiple additional cost zero, and cut with smallest possible multiple additional cost need not be an optimal cut.