A Gentle Guide to Multiple Alignment

Version 2.03
July 1996, March 1997

Georg Fuellen
fuellen@dali.mathematik.uni-bielefeld.de, fuellen@Techfak.Uni-Bielefeld.DE
fuellen@ alum.mit.edu

Please send comments, critique, flames and praise to the author, at fuellen@dali.mathematik.uni-bielefeld.de !

An HTML version is available at
`http://www.techfak.uni-bielefeld.de/bcd/Curric/MulAli/mulali.html.`
A List of WWW Resources is available at
`http://www.techfak.uni-bielefeld.de/bcd/Curric/MulAli/welcome.html.`

Instructions for obtaining the Solution Sheet are available by sending email to
majordomo@lists.uni-bielefeld.de, with no subject line, and the following message body: subscribe vsns-bcd-solutions

Prerequisites. An understanding of the dynamic programming (edit distance) approach to pairwise sequence alignment is useful for parts 1.3, 1.4, and 2. Also, familiarity with the use of Internet resources would be helpful for part 3. For the former, see Chapters 1.1 – 1.3, and for the latter, see Chapter 2 of the Hypertext Book of the GNA-VSNS Biocomputing Course at http://www.techfak.uni-bielefeld.de/bcd/Curric/welcome.html.

General Rationale. You will understand why Multiple Alignment is considered a challenging problem, you will study approaches that try to reduce the number of steps needed to calculate the optimal solution, and you will study fast heuristics. In a case study involving immunoglobulin sequences, you will study multiple alignments obtained from WWW servers, recapitulating results from an original paper.

Revision History. Version 1.01 on 17 Sep 1995. Expanded Ex.9. Updated Ex.46. Revised Solution Sheet -re- Ex.3+12. Marked more Exercises by “A” (to be submitted to the Instructor). Various minor clarifications in content

and style.
Version 1.02 on 25 Jan 1996. Revised sections 1.3 and 2.1.
Version 1.03 on 11 Mar 1996. Fixed the URL for the Clustal Alignments in 3.4. Various minor changes.
Version 2.0 on 6 Jun 1996. Added link to our very own Java-based "Multiple Alignment Visualization Tool" (in section 1.3). Clarified section 2.1 -re- nodes vs paths defining the Carrillo-Lipman polyhedron. Followed up on the new definition of "Replacement" in Ch.1. Clarified formulas in section 2.2 by adding extra parantheses. Added immunoglobulin WWW resources. Clarified Ex. 53. Various other improvements, in particular to section 1.3.
Version 2.01 on 21 Jun 1996. Reworded a few paragraphs in section 2 (see Newsletter Vol.2 No.3, item 4).
Version 2.02 on 27 Jul 1996. Using the term "polyhedron" instead of "polytope" in section 2.2. Marked Exercises by "B" if they have more biological depth.
Version 2.03 on 18 Mar 1997. Added references to alignment methods using Hidden Markov Models and Gibbs Sampling. Various minor improvements.

Contents

1 Basics of Multiple Alignment

Here is what you will learn in the following sections: Multiple Alignment will be defined, we will discuss the straightforward approach to calculating an "optimal" one, and you will understand that, unlike pairwise alignment, even an intelligent order of computation is not enough to finish the calculation in reasonable time.

1.1 What is a Multiple Alignment ?

[WhatIs.] What is a multiple alignment ? The short answer is this -

```
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCSSSGFIFSS--YAMYWVRQAPG
LSLTCTVSGTSFDD--YYSTWVRQPPG
PEVTCVVVDVSHEDPQVKFNWYVDG--
ATLVCLISDFYPGA--VTVAWKADS--
AALGCLVKDYFPEP--VTVSWNSG---
VSLTCLVKGFYPSD--IAVEWESNG--
```

That's a multiple alignment. 8 fragments from immunoglobulin sequences are displayed together. Their alignment highlights conserved residues (one of the cysteines forming the disulphide bridges, and the tryptophan are notable), conserved regions (in particular, "`Q.PG`" at the end of the first 4 sequences), and more sophisticated patterns, like the dominance of hydrophobic residues at fragment positions 1 and 3. The alternating hydrophobicity pattern is typical for the surface beta-strand at the beginning of each fragment. Indeed, multiple alignments are helpful for protein structure prediction.

The alignment can also enable us to infer the evolutionary history of the sequences. It looks like the first 4 sequences and the last 4 sequences are derived from 2 different common ancestors, that in turn derived from a "root" ancestor. Indeed, we've got 4 fragments from the so-called *variable* regions, and 4 fragments from the *constant* regions of immunoglobulins. (Don't be fooled by the term "variable". The sequences of the variable regions are about as conserved as the sequences of the constant regions, except for their antigen-binding subregions, which are composed of just a few amino acids each, and give the antibody its specificity.) However, it is necessary to inspect longer fragments than shown here, if you want to make phylogenetic observations that are statistically significant. In this chapter, we will not look into phylogenetic analysis, or statistical significance in general; they deserve chapters of their own !

Exercise 1 [05,opt.] Find out more about the role that hydrophobic and hydrophilic residues play in protein structures. Also, do you think the `G` at the end of all but one fragment is a residue conserved over time ?

Exercises are marked with numbers that are (hopefully !) proportional to their difficulty. Those that are clearly optional are marked by the acronym 'opt'. Also, an appended symbol '*' marks the essential exercises, 'M' marks more mathematical depth (relative to no depth at all :-), 'B' marks more biological depth, 'A' marks assignments you are asked to submit to your instructor, and an appended letter 'P' marks things you can write a program for (preferably in MOO-code, so that you can demonstrate it in our electronic classroom. Contact the author at fuellen@dali.mathematik.uni-bielefeld.de before you start coding !).

In case you're wondering about the '*[WhatIs.]*' at the beginning of this section: I've provided acronyms so that you can clearly specify which part of the text you're talking about in the electronic classroom. This is paragraph "WhatIs-6" (the sixth paragraph in the "WhatIs" section), and no matter whether the others studied the hypertext or the postscript version, they will all find it.

[FormalDef] Computer scientists and mathematicians prefer a longer, more formal answer to our question "What is a Multiple Alignment ?", as follows:

Let's imagine we've got k sequences, $s_1, \ldots, s_k$, each sequence consisting of characters taken from an alphabet of letters, denoted $\mathcal{A}$ (see chapter 1). $\mathcal{A}$ can be { `A,C,G,T`} for DNA sequences. Let's say k must be at least 2, since aligning zero or one sequences just doesn't make sense. (If you're a good observer, you will note that most of the chapter doesn't make sense for 2 sequences either. Well, it makes certain sense by boiling down to useless, yet obviously true observations for the pairwise case. However, in practice, multiple alignment is never done for less than 3 sequences.) Next, we need to insist that $\mathcal{A}$ does not contain the special character "–", which we want to reserve for denoting the gaps. Then we can write our alignments using the alphabet $\mathcal{A}'$, which is $\mathcal{A}$ plus the gap character "–". $\mathcal{A}'$ can be { `A,C,G,T,`–} for DNA sequence alignments. We define -

A *Multiple Alignment* of k sequences is a rectangular array, consisting of characters taken from the alphabet $\mathcal{A}'$, that satisfies the following 3 conditions:

1. There are exactly k rows.
2. Ignoring the gap character, row number i is exactly the sequence s_i.
3. Each column contains at least one character different from "–".

If the sequences are written as

$$\begin{array}{rcccc} s_1 = & s_{1,1} & s_{1,2} & \dots & s_{1,|s_1|}, \\ s_2 = & s_{2,1} & s_{2,2} & \dots & s_{2,|s_2|}, \\ & & \dots & & \\ s_k = & s_{k,1} & s_{k,2} & \dots & s_{k,|s_k|}, \end{array}$$

then the multiple alignment will be written as

$$\begin{array}{rcccc} s'_1 = & s'_{1,1} & s'_{1,2} & \dots & s'_{1,|A|}, \\ s'_2 = & s'_{2,1} & s'_{2,2} & \dots & s'_{2,|A|}, \\ & & \dots & & \\ s'_k = & s'_{k,1} & s'_{k,2} & \dots & s'_{k,|A|}. \end{array}$$

Exercise 2 [00] Can you imagine what $|A|$ is, before reading on ?

Some of the $s'_{i,j}$ are gaps, and ignoring them, the rows s'_i become the original sequences s_i, like sequence No. 8 from the example alignment:

Before:

$s'_8 =$ `V S L T C L V K G F Y P S D - - I A V E W E S N G - -`

After:

$s_8 =$ `V S L T C L V K G F Y P S D I A V E W E S N G`.

Depending on how many gaps we inserted, the multiple alignment array has got a particular width, which we have called $|A|$.

[Compare] Our next step is to compare multiple alignments. Our definition tells us what a multiple alignment is, but not whether the one from the beginning of the chapter,

```
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCSSSGFIFSS--YAMYWVRQAPG
LSLTCTVSGTSFDD--YYSTWVRQPPG
PEVTCVVVDVSHEDPQVKFNWYVDG--
ATLVCLISDFYPGA--VTVAWKADS--
AALGCLVKDYFPEP--VTVSWNSG---
VSLTCLVKGFYPSD--IAVEWESNG--
```

or the following one,

```
VTISCTGSSSNIG-AGNHVKWYQQLPG
VTISCTGTSSNIG--SITVNWYQQLPG
LRLSCSSSGFIFS--SYAMYWVRQAPG
LSLTCTVSGTSFD--DYYSTWVRQPPG
PEVTCVVVDVSHEDPQVKFNW--YVDG
ATLVCLISDFYPG--AVTVAW--KADS
AALGCLVKDYFPE--PVTVSW--NS-G
VSLTCLVKGFYPS--DIAVEW--ESNG
```

or, maybe, even this candidate

```
VTISCTGSSSNIGAG-NHVKWYQQLPG
VTISCTGTSSNIGS--ITVNWYQQLPG
LRLSCS-SSGFIFSS-YAMYWVRQAPG
LSLTCT-VSGTSFDD-YYSTWVRQPPG
PEVTCVVVDVSHEDPQVKFNWYVDG--
ATLVCLISDFYPGA--VTVAWKADS--
AALGCLVKDYFPEP--VTVSWNSG---
VSLTCLVKGFYPSD--IAVEWESNG--
```

is the "better" alignment. What is "better", anyway ? We need a *cost/weight function*, and a really simple way to start is to evaluate the costs/weights column by column, like this:

$$ColumnCost\begin{pmatrix} \mathtt{L} \\ \mathtt{L} \\ \mathtt{A} \\ \mathtt{P} \\ \mathtt{G} \\ \mathtt{S} \\ - \\ \mathtt{G} \end{pmatrix} = 26.$$

I have not made up the value "26", but I've used the "unit costs" from chapter 1 about pairwise alignment, summing up all the costs *of all possible pairs of letters*, i.e. the sum of the unit costs of the pairs (1,2), (1,3), (1,4), ..., (1,8), (2,3), (2,4), ..., (2,8), (3,4), (3,5), ..., ..., ..., and (7,8). In general, any cost/weight scheme could be used, it just needs to map pairs of characters to a numeric value. "Unit costs" is just a very convenient example.

Exercise 3 [00] Using unit costs, recalculate "26" for yourself.

Exercise 4 [10A] Use the 250 PAM similarity matrix to calculate the cost of the column. To do this, convert the PAM similarity scores into costs. For any pair of amino acids, calculate 17 minus the PAM score of that pair. ("17" is the largest value in the standard PAM similarity matrix. This is how the MSA software does the conversion, see below.) For the comparison with "–", e.g. pairs like (A,–), use a cost of **20**. The 250 PAM matrix is available at http://www.techfak.uni-bielefeld.de/bcd/Curric/PrwAli/Matrices/pam250.mat.

In this text, our goal is to minimise the total cost, or distance, for the alignment, so *smaller is better.* In contrast, sometimes people aim to maximise similarity. These people can use the PAM similarity matrix directly in their calculations.

[ColumnsFirst] A column is just a generalisation of an edit operation as introduced in Chapter 1. Indeed, we can view the operations "Replace/Match", "Insert", and "Delete" as pairs of characters, i.e. $\begin{pmatrix} s_{i,l} \\ s_{j,l} \end{pmatrix}$, $\begin{pmatrix} - \\ s_{j,l} \end{pmatrix}$ and $\begin{pmatrix} s_{i,l} \\ - \end{pmatrix}$, where $s_{i,l}$ and $s_{j,l} \in \mathcal{A}$. Therefore, the edit operations can be represented by columns with 2 entries. The replacement of one character by a different character is also called a mismatch.

Exercise 5 [00] Not each column with 2 entries represents an edit operation, though. Which one does not ?

Given a cost/weight schema w mapping pairs of characters to a numeric value, we can calculate a cost of the overall alignment, by summing up the column costs:

$$c = \sum_{l=1,\ldots,|A|} \sum_{(i,j),i<j} w(s'_{i,l}, s'_{j,l}),$$

where $w(-,-) = 0$, $|A|$ is the width of the alignment, and $\{(i,j), i < j\}$ is the set of all pairs, where the first index i is smaller than the second index j. It goes without saying that i, j are larger than zero, and less than or equal to the number of sequences, k.

Our cost model is a simple example of what is known as a *sum-of-pairs* cost, abbreviated to *SP-cost*, and we can think of the pairs as being "projections" of the column, in the same way as objects in a cubic lattice (3 dimensions) can be projected by light sources, forming shadows on the 6 different faces of the cube (2 dimensions). "Sum-of-pairs" is just one way of scoring. We'll touch another one later.

Exercise 6 [05P, opt.] In the "unit cost" model, calculate the simple SP-cost of our example alignment from the beginning of the chapter.

[PairsFirst] Equivalently, we can obtain the simple SP-cost of an alignment by calculating all pairwise costs in the same way as we did in chapter 1 (i.e., by looking at the number of replacements/matches, insertions and deletions), and *then* summing up over all pairs of sequences:

$$c' = \sum_{(i,j),i<j} \sum_{l=1,\ldots,|A|} w(s'_{i,l}, s'_{j,l}).$$

Exercise 7 [05P, opt.] Again, calculate the SP-cost of our example alignment from the beginning of the chapter, according to the last equation.

I hope you obtained the same results for the last 2 exercises ! We've just rearranged the order of summation, so this cost is the same as the one calculated before. This rearrangement is possible because we've got a cost/weight scheme w defined on pairs of characters, *and nothing more.*

In most cases, only one of the 2 ways of summation can be used to express a particular cost model, and this makes a standard formulation of multiple alignment a little difficult. Some approaches are best cast "columns first", whereas others can only be cast "pairs first".

Exercise 8 [10A] Imagine you want to calculate the *shortest common supersequence*, i.e. the shortest sequence to which all sequences can be aligned without mismatches. Which cost/weight scheme can you use ? Hint: a) Despite its name, the shortest common supersequence cannot be shorter than the longest of the sequences. b) Cast the problem "columns first", and imagine how the cost of a column should be defined. If you do it right, there will be a simple relation between the cost of the whole multiple alignment and the length of the supersequence. (cf. [Kec93].)

Exercise 9 [05A] Come up with a cost model that must be treated "pairs first". Hint: In the following alignment, is it fair to treat all 5 gaps as seperate entities ?

```
G-----S
GNAVSNS
GNANANS
```

Indeed, we will concentrate on the "pairs first" approach, and in the next section, a pair of rows will be called a *projection* of the whole multiple alignment, like a pair of letters is a projection of an alignment column.

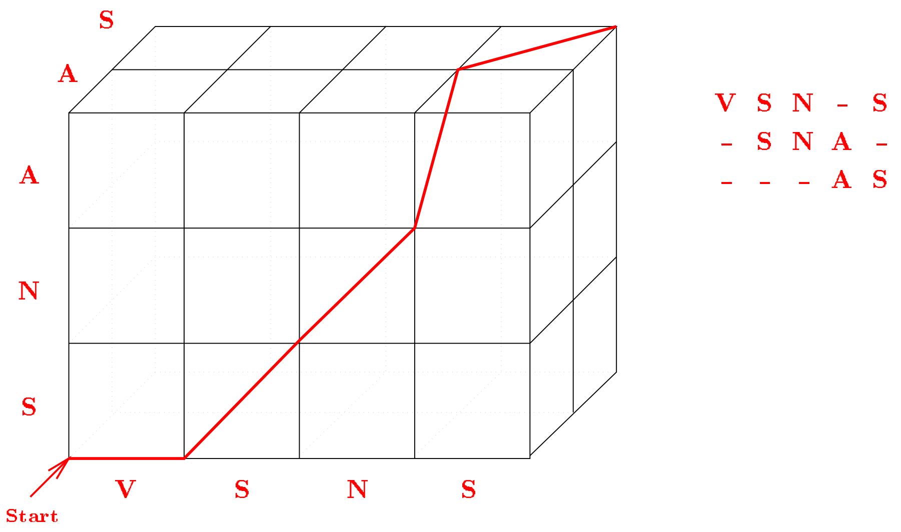
Now we're ready to talk about "Optimal Multiple Alignments".

An *Optimal Multiple Alignment* is an alignment with minimum overall cost, or maximum overall similarity. We'll denote the cost of the optimal alignment of sequences $s_1, s_2, \ldots, s_k$ by $d(s_1, s_2, \ldots, s_k)$.

1.2 The Dynamic Programming Hyperlattice.

[Lattice] Recall the "path through a distance lattice/matrix" concept from Chapter 1. For the case of 3 sequences, every alignment can be cast as a unique path through a 3-dimensional lattice (See Fig. 1 for an example).

Figure 1: Alignment Path for 3 Sequences.



We can denote a path through a lattice in a simple way. For each node visited, we list, component by component, the distance from the starting point in the bottom left (i.e. from the source (0,0,0) of the lattice). For example, the path in Fig. 1 is written down as (0,0,0), (1,0,0), (2,1,0), (3,2,0), (3,3,1), (4,3,2). As you can see, the distance from the starting point is the number of letters already aligned. For example, column 4 of the alignment corresponds to node (3,3,1). Indeed, 3 letters from the first and second sequence are aligned at that point, and one letter from the third.

You can create alignment paths and 3-dimensional lattices, and rotate them in the window of your WWW-browser, using our very own Java-based "Multiple Alignment Visualization Tool", at http://oleander.techfak.uni-bielefeld.de:8080/java/biomoo/visu.html .

Exercise 10 [10, opt.] Draw the hyperlattice for the following alignment, and write down the alignment path.

```
GN-S
GNA-
-N-S
```

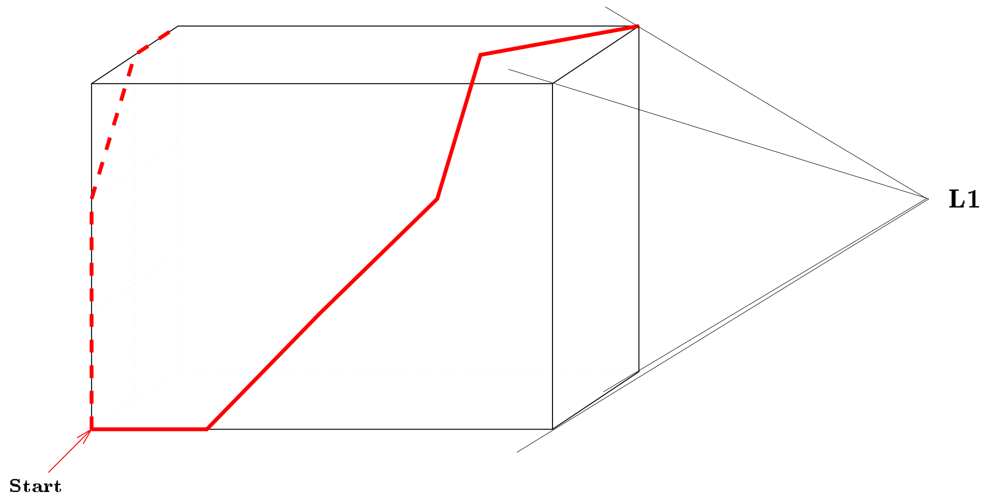
Now I'd like to invite you to relax and imagine an 8-dimensional hyperlattice... In our example from the beginning of this chapter, we're first walking straight ahead into the lattice, following the main diagonal. Starting again at node (0,0,0,0,0,0,0,0) we move to node (1,1,1,1,1,1,1,1), and so forth, until we've reached the node (14,14,14,14,14,14,14,14). The next three nodes are as follows, (**15**,14,14,14,**15**,14,14,14), (**15**,14,14,14,**16**,14,14,14), and (**16**,15,15,15,**17**,15,15,15).

Exercise 11 [02*] And the next node is ??

Exercise 12 [05M, opt.] Using the convention that for any letter L in A, $0 \cdot L = -$, and $1 \cdot L = L$, can you cast an alignment column as the combination of a vector of binary numbers e and the letters at the incoming edges of a node in the lattice ? (cf. [Wat89]).

In Fig. 1, a three-dimensional lattice is displayed, with sequences starting at the bottom-left end. If you imagine light sources on the top, front, and right-hand side of the lattice, "shadows" of the alignment will be projected to the opposing faces (walls). In Fig. 2, only the light source on the right is "on", projecting the path onto the face on the left. In Fig. 3, all light sources

Figure 2: Projection of the Alignment from the Right-Hand-Side.



are "on". (The light sources should ideally be much farther away from the lattice, so that the shadows are projected without distortion.)

[ProjAndGaps] We've already dealt with the projections of a column; here the whole alignment path is being projected ! The projections of a multiple alignment to pairwise alignments will play an important role in speeding up the calculation of the optimal one.

The projection of an alignment may be "shorter" than the original one. For example, the alignment of

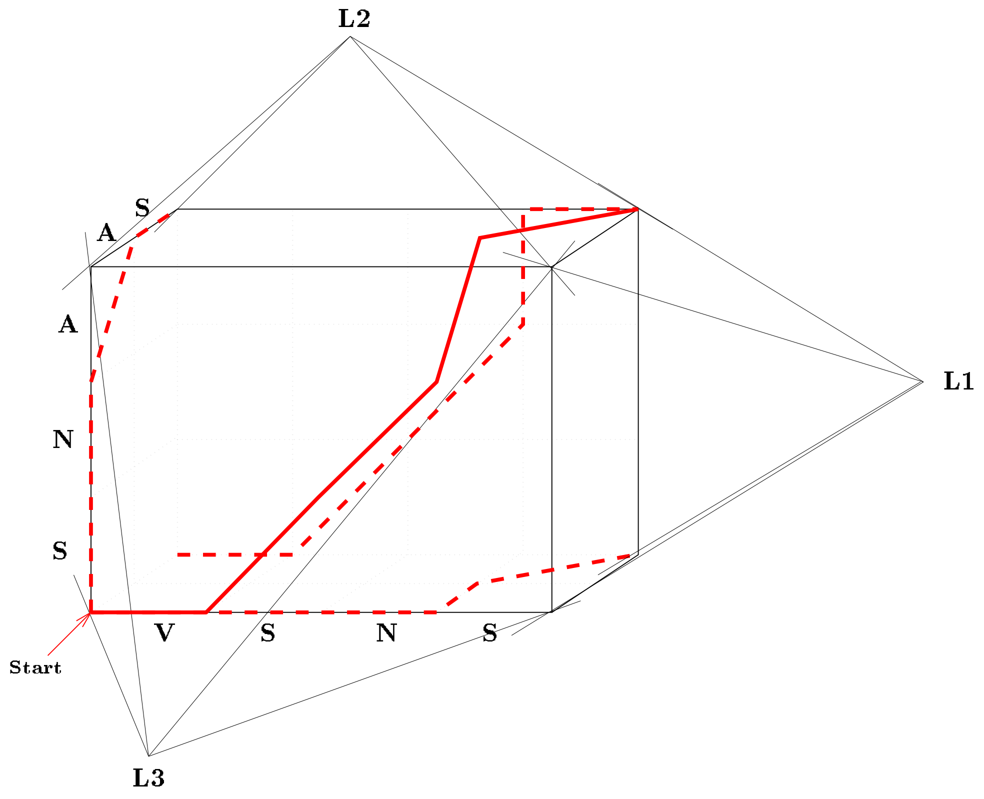
```
G---SNS
GN----S
GNAVSNS
```

projected in the direction of the first 2 sequences, is as follows,

```
G-SNS
GN--S
```

Aligned gaps are ignored ! This is exactly what happens if the alignment path progresses in the direction of the projection; there is no shadow left ! For example, if the alignment path in Fig. 3 at first progresses towards the right of the hyperlattice, in the direction of the first sequence, and is

Figure 3: All 3 Pairwise Projections of the Alignment



perpendicular to the other 2 sequences, the projection by light source L1 is not a line, but a single point. And indeed, "`V`" is aligned to 2 gaps.

Exercise 13 [02*A] Can the projected path of the *optimal* multiple alignment ever be less costly than the optimal pairwise alignment ? Explain why, or why not.

1.3 Calculation of a Multiple Alignment by Standard Dynamic Programming.

[CalculDynPgr] In a straightforward extension of the pairwise case (see chapter 1) an optimal multiple alignment can be calculated by dynamic programming. Let us reconsider the pairwise case. Here, we can visualize dynamic programming as a calculation that visits every node in a 2-dimensional lattice, in a way that obeys the order of dependencies between the nodes, as indicated by the arrows in Fig. 4.

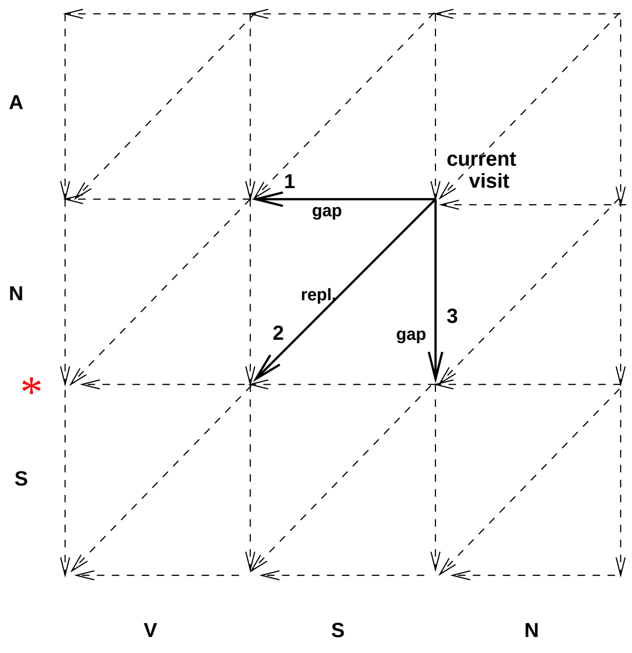
The lattice is the 2-dimensional equivalent of the hyperlattice introduced earlier, and each lattice node can be identified with the corresponding position in the "distance matrix" discussed in chapter 1. Once the calculation is finished, we have assigned a cost to each node. This cost is the minimum cost of aligning the two sequences, here `VSN` and `SNA`, up to the point defined by the node.

As in Fig. 1, but in contrast to chapter 1, we've arranged the sequences such that the calculation of the alignment costs *starts* in the *bottom-left* corner, and not the top-left corner. In this setting, one way to obey the order of dependencies is the following: Start bottom-left, then move to the right until the bottom row is finished, then visit the node marked by an asteriks (*), move to the right as before, etc.

The arrows in Fig. 4 denote the dependencies, e.g. the calculation for the node labeled "current visit" depends on 3 other nodes; we are looking back at 3 possible edges from which we can reach it. These, in turn, correspond to either a replacement/match (diagonal arrow), or the introduction of a gap in one of the sequences. Which of these 3 "edit operations" gives rise to the minimum overall cost for reaching the current node, and thereby suggests the best path to it, i.e. the best alignment of the sequences up to this point, `VS` and `SN` ? We have to look at

1) the weight of the edit operation,

Figure 4: Looking Back at Previously Visited Nodes.



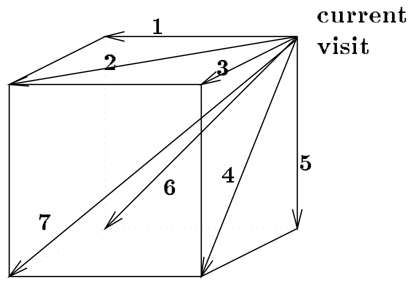
2) the costs that incurred before, for reaching the node to which the arrow is pointing.

Then we take the minimum possible sum of 1) and 2), and store it in the current node, just like the costs that incurred before were stored in previously visited nodes. Following the order of dependencies guarantees that these costs are always known. Note that only the previous costs are important, not the paths by which they were achieved. The alignment is spelled out by tracing back, i.e. by starting in the top-right corner, and following the nodes from which the minimum cost was contributed while taking the minimum possible sum of 1) and 2).

Extending this technique to 3 or more dimensions leads to formulas very much like the formulas from chapter 1. We've just got more nodes to look back, e.g. 7 nodes for three sequences. Correspondingly, the minimum needs to be taken from 7 possible values (see Fig. 5).

Exercise 14 [10M, opt.] If you've done the last exercise in the "math track", you can cast the recursive (dynamic programming) formula in a very

Figure 5: Looking Back in the Case of 3 Sequences.



elegant way, taking the minimum, over all vectors of binary numbers $e \neq (0, 0, ..., 0)$, of an expression dependent on e. (cf. [Wat89]).

Whether we investigate the formulas, or just visualize all the nodes of a, say, 8-dimensional hyperlattice, we can easily see that our calculation is pretty much time- and space-consuming !

In the next subsection, we will formalize "pretty much", and then we'll look at a technique to cut down the number of nodes that need to be visited to calculate the alignment.

Exercise 15 [10, opt.] Imagine that you want to use a hill-climbing strategy like Simulated Annealing, or a Genetic Algorithm, to find minimum-cost alignments. Which obstacle(s) need to be overcome ? (See Chapter 5, at http://www.techfak.uni-bielefeld.de/bcd/Curric/ProtEn/contents.html, for an introduction to Genetic Algorithms. Cf. [Vin91], p.3).

1.4 Computational Complexity of Multiple Alignment by Standard Dynamic Programming.

[ComplexityDynPgr] The basic argument for the time analysis of standard dynamic programming is that each node in the k-dimensional hyperlattice is visited once, and therefore the running time must be proportional to the number of nodes in the lattice.

Looking at the 3-dimensional lattice we've used for visualisation, this number is the product of the lengths of the sequences. The important question remaining is thus: How many steps does the algorithm "rest" at each node ? Dynamic programming organizes the visiting of nodes in such a way that we just need to "look back" one single step, at the nodes that we've visited before, to look up the values we need for calculating the minimum. (See Fig. 4 and 5.) So the time we spend for retrieving the minima and calculating the sum does not depend on the length of the sequences ! However, it depends on the number of sequences. We've had 3 values (Fig. 4) for 2 sequences, 7 values (Fig. 5) for 3 sequences, and, you may have guessed it, 15 values for 4 sequences. This goes up exponentially, just like the $2^k - 1$! Using our reasoning, one can formally prove that the running time is

$$O(2^k \cdot \prod_{i=1,\ldots,k} |s_i|),$$

where $O(\ldots)$ denotes proportionality.

All this is bad news ! If the proportionality factor is 1 nanosecond, then for 6 sequences of length 100, we'll have a running time of $2^6 \cdot 100^6 \cdot 10^{-9}$, that's roughly 64000 seconds. Just add 2 sequences, and the running time is $2.6 \cdot 10^9$ seconds !

Exercise 16 [02A] What's the running time for 10 sequences of length 200 ?

Even worse, let's have a look at the memory space we're using... If we want to trace back the alignment, we need to store the whole lattice, a datastructure the size of a multidimensional skyscraper. In fact, space is the No.1 problem here, bogging down multiple alignment methods that try to achieve optimality. Furthermore, incorporating a realistic gap model (see chapter 1), we will further increase our demands on space and running time, although there is a reasonable gap cost model that does not involve too many excess computations, see [Alt89].

1.5 Some Bibliographic Hints.

Some introductory texts on multiple alignment are the following. Chapter 10 from the new book by Michael Waterman [Wat95] gives you a good overview, including some more mathematics. [Wat89] is an older introductory text by the same author. Less mathematical, but still from the viewpoint of Computer Science is the text [Mye91]. Some papers with a nice introductory section are [GKS95], [Gus93], and [Kec93]. Reviews from the biologists' perspective are centered around heuristic methods; see section 3.5 for more hints.

Two approaches to Multiple Alignment that are not treated here are the use of Hidden Markov Models ([BCHM94]), [KBM+94], [Edd95]) and Gibbs Sampling ([LAB+93]); both are based on statistical methods. The former is also discussed in [Wat95].

2 The Carrillo-Lipman Method for Optimal Multiple Alignment

Here is what you will learn in the following sections: You will understand geometrically the idea of using pairwise projections for obtaining bounds on the "volume" the dynamic programming algorithm needs to explore. Then you will be shown the math, and correlate both forms of presentation. (In section 3.4, you can apply your knowledge to a real-life example. In any case, you may read and explore section 3, "Heuristic Alignment Procedures and Examples", in parallel.)

2.1 A Visual Explanation of the Carrillo Lipman Bound.

[CLVisual] Recall the dynamic programming hyperlattice introduced in the last sections. If many sequences are to be aligned, it becomes too large, so that we can no longer visit every node in it. Do we really need to ? Do we need to care about nodes close to the extreme ends of the lattice, like E1, E2, E3 or E4 (Fig. 6) ? We're driven to the extreme ends if we introduce a lot of gaps; the alignment path of

```
AAAACCCCCC----
----CCCCCCTTTT
```

is 4 steps away from the main diagonal, because every gap introduced in the beginning moves us one step away.

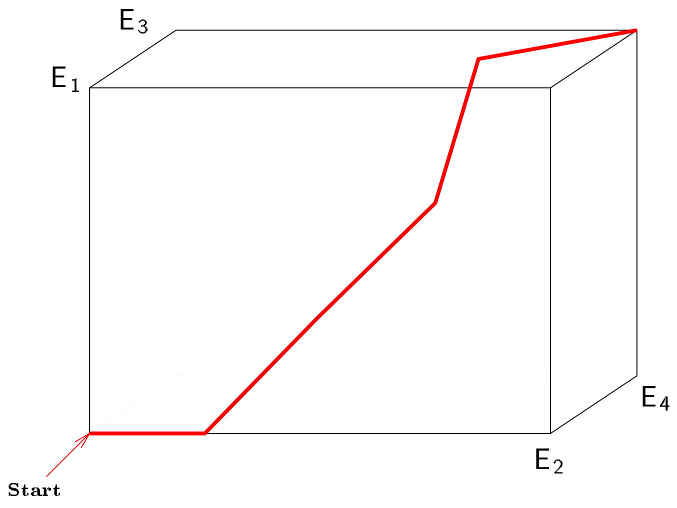
Exercise 17 **[00]** How many steps away from the main diagonal was the standard example from Chapter 1.2, Pairwise Alignment via Dynamic Programming, see http://www.techfak.uni-bielefeld.de/bcd/Curric/PrwAli/prwali.html ?

Working with the "sum-of-pairs" cost model introduced in the last section, the technique presented in the following can be very useful if the sequences are rather similar, and the cost for introducing gaps is not too low. Then, the optimal multiple alignment does not look anything like

```
AAAA--------
----CCCC----
--------TTTT
```

In other words, we can expect a priori that the optimal alignment path is contained in a "polyhedron" close to the main diagonal (Fig. 7, bottom.

Figure 6: Extreme Ends of the Hyperlattice.



Here, a polyhedron is a solid formed by plane faces, or more complicated 2-dimensional surfaces. For better visualisation, the polyhedron's shadows are displayed, together with a tube symbolizing the optimal alignment path.) While visiting a node and looking for the minimum along all the incoming edges, we can ignore those edges that are "coming from outside the polyhedron", as in the top part of Fig. 7. On its top-left side, the cube is "covered" by the polyhedron. The edges 1, 2, 3, 6 and 7 are coming from the inside, and edges 4 and 5 can be ignored (and are therefore not labeled in the figure).

The question is, how can we obtain such a polyhedron ? Imagine we've got a heuristic alignment that is a path hopefully close to the optimal alignment. If we knew that the optimal is max. 30 units away, we could establish a simple polyhedron of radius 30 around the heuristic alignment, and search for the optimal inside this polyhedron. But how are we to know ?

In fact, we will *not* make such simple assertions. Here is what we will do. 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 upper bounds. We will

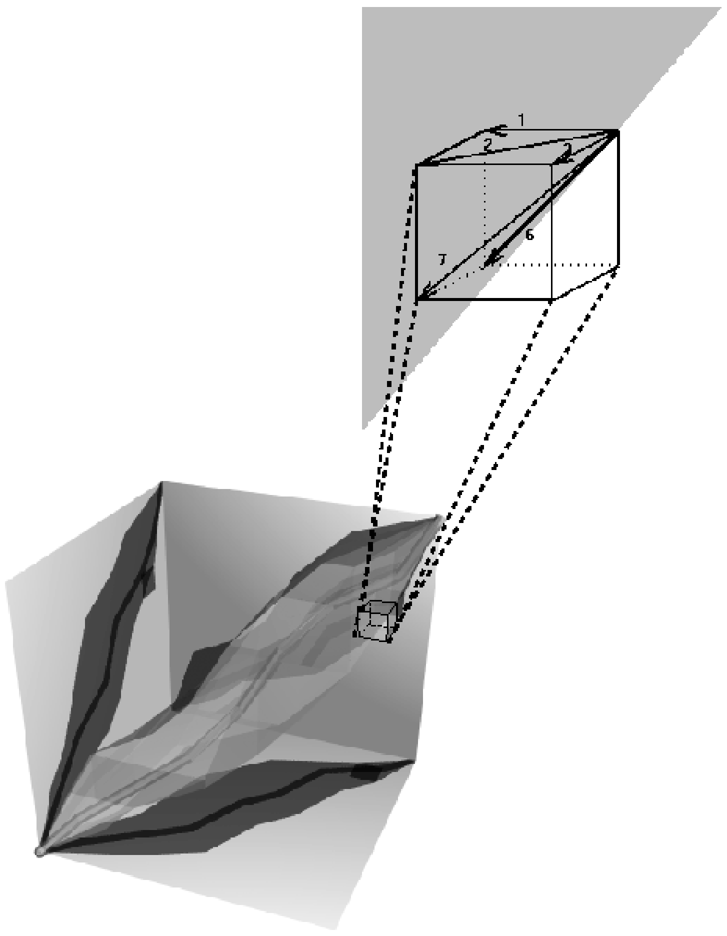


Figure 7: Cutting down the Exploration of the Dynamic Programming Lattice.

show that these bounds are obeyed by the projections of the optimal multiple alignment, so that it is indeed pointless to consider nodes through which only paths go that have higher costs in some projection.

[GetBounds] To obtain the bounds, we will look at the pairwise projections of a heuristic alignment, and of the hypothetical optimal multiple alignment. *(For the moment, we'll behave as if the optimal multiple alignment is known. However, we'll make no use of this knowledge in the final equation.)*

So let's fix a heuristic alignment, and let's say, we're projecting in the direction of the sequences 1 and 2. We do not know how close the two paths of the projected heuristic and the projected optimal alignment are. Even worse, for the original paths, we could at least say that the heuristic alignment is more costly than the optimal one (otherwise the optimal one was misnamed). For the projected paths, this does not need to hold true; imagine the heuristic starts with aligning the pair (1,2) optimally, and then, in some way, adds the other sequences one by one. The projection of this heuristic multiple alignment to the first 2 sequences *is* the optimal pairwise alignment, and the *projection* of the optimal multiple alignment can only be *more* costly (otherwise, the optimal pairwise one was misnamed). In other words, aligning multiply puts constraints on the pairwise projections, they are no longer independent, they *all* need to fit together, forming the multiple alignment in a consistent way.

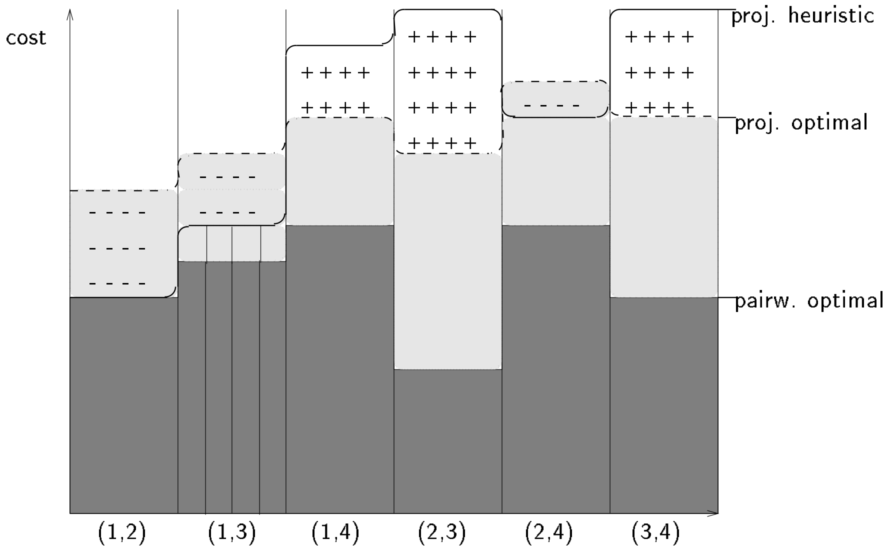
If you are pedantic, you may have noticed that whenever we said "more costly" in the preceding paragraph, what we really meant was "more costly or costing the same". To simplify our presentation, we'll continue to use this convention.

Exercise 18 [05A] For 3 short sequences, make up an artificial example where the projection of the optimal multiple alignment in the direction of the pair (1,2) is more costly than the optimal pairwise alignment.

Let us now visualize the 3 types of costs we're dealing with for each of the 6 pairs / projections that are possible in the 4-sequence case, as in Fig. 8. These are the costs of the projected heuristic (normal line), the projected optimal (dashed line), and the pairwise optimal alignment (borderline between the grey areas), each of them for the pairs (1,2), (1,3), (1,4), (2,3), (2,4) and (3,4).

[FakeUppBnd] For each column, pair by pair, we will soon establish an *upper* bound on the grey area (dark grey and light grey taken together), i.e.

Figure 8: Costs of the (Projected) Pairwise Alignments for 4 Sequences.



the area below the projected optimal alignment cost. Our bound will consist of

- the cost of the projected heuristic alignment, *plus*
- a "compensation term", defined below.

The areas filled with minus symbols denote the "defect" of the projected optimal in comparison to the projected heuristic alignment; only for pairs (1,4), (2,3), and (3,4) the projected heuristic is more costly, i.e. is an upper bound. For these cases, we've marked the "surplus" by rows of plus symbols.

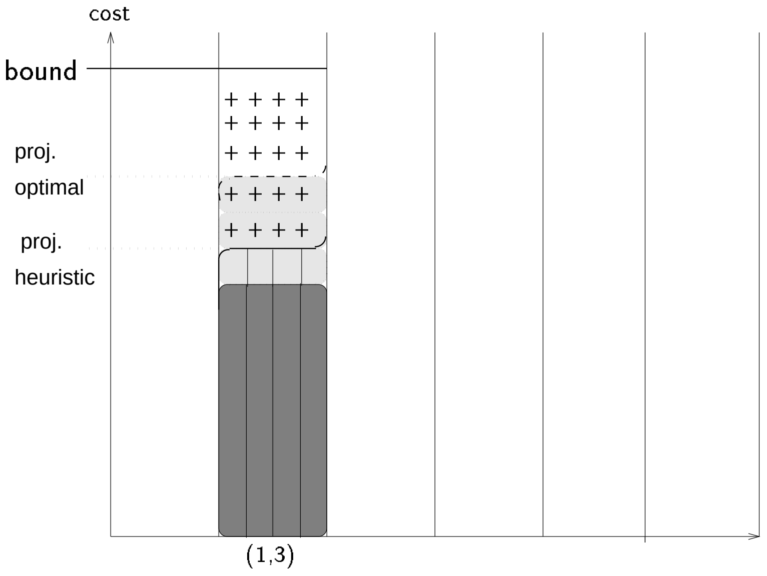
Let's start with the pair (1,3). How can we arrive at an upper bound for its corresponding projected optimal alignment ? Is there a never-failing "compensation" that could be added to the cost of the projected heuristic alignment, i.e. added to the area marked by vertical lines ? The critical observation here is that the areas filled with plus symbols, summed up over *all* pairs, are *larger* than the areas filled with minus symbols.

Exercise 19 [00*] Recall why this must be the case.

We can conclude the following: If for one pair there is a defect (like the 2 rows of minus symbols in pair (1,3)), then there is a "compensation" that is *larger* than the defect. The "compensation" is the sum of the surpluses, minus the defects taken over *all remaining* columns, if there are any.

Therefore, adding the "compensation" to the projected cost of the heuristic (marked by vertical lines), we have achieved an upper bound on the grey area, as shown in Fig. 9. In our example, the "compensation" is 5 rows of plus symbols (9 layers of "surplus" minus 4 layers of "defect" in columns other than (1,3)). Indeed, it is larger than the (1,3)-"defect" of 2 rows of minus symbols, as shown in Fig. 8.

Figure 9: Adding the "Compensation" to the Costs of the Projected Heuristic Alignment



Exercise 20 [05] Calculate the upper bound for the pair (2,4).

It's easy to see that our argument is true for the pair (1,2) as well; it is irrelevant whether the projected heuristic and the pairwise optimal have the same cost, or not. (Recall that same cost does not imply identity of the alignment path.) In the figure, 9 layers of plus symbols minus 3 layers of minus symbols are larger than 3 layers of minus symbols.

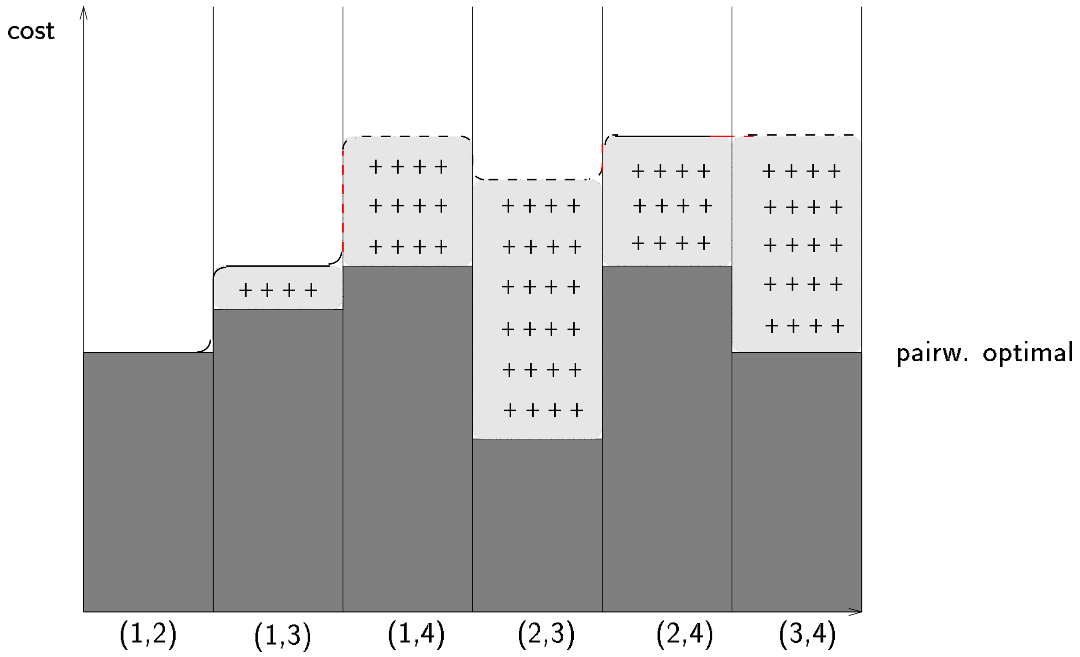
For the pair (1,4) the bound holds as well. Here we add a "compensation" of one layer (7 layers of plus symbols, 6 layers of minus symbols) to the cost of projected heuristic. The "compensation" may actually be negative, if there is no defect to be covered. This is the case for pair (2,3).

Exercise 21 [05] Calculate the upper bound for the pair (2,3).

[UppBnd] Things are looking good, but we're not ready yet; in fact, we're pretending that we're aware of the cost of the optimal alignment; we've been playing around with a hypothetical value !

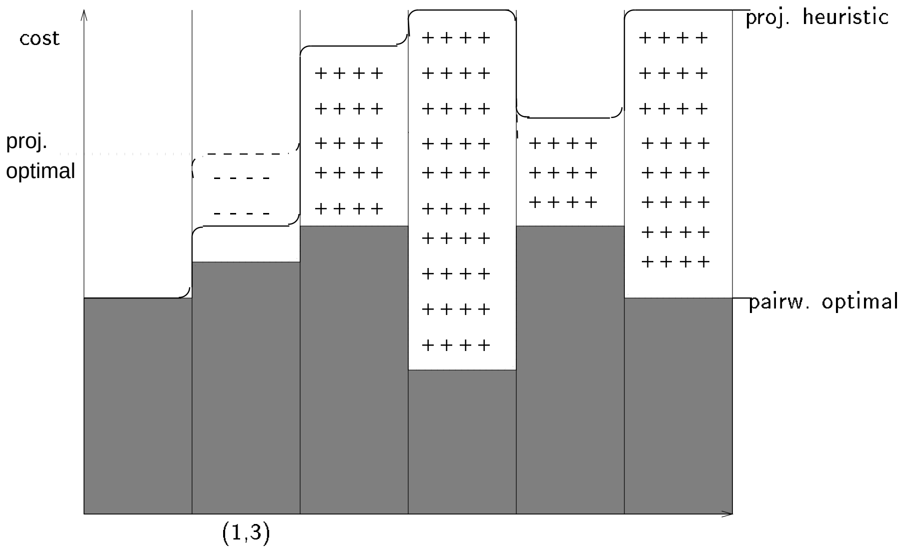
There is a quick fix to this; while calculating the "compensation", let's just add more layers of plus symbols as shown in Fig. 10, calculating the surplus "in relation to the cost of the optimal pairwise alignment" instead of "in relation to the projection of the unknown optimal multiple alignment". This means all the area between the optimal pairwise alignment and the projected heuristic multiple alignment is defined as surplus. Our change has the extra "benefit" that there are no negative contributions to the "compensation" any more.

Figure 10: Getting Rid of the References to the Projected Optimal Alignment.



In other words, we've just replaced the references to the projections of the optimal alignment with references to the pairwise optimal alignments. Their costs are not only known, but well-known as a lower bound to any alignment that could be done pairwise. In the case of pair (1,3), the "compensation" now consists of all the rows of pluses in Fig. 11, and if you add that area to the cost of the projected heuristic in column (1,3), you'll see in Fig. 12 that it compensates well, too well ! In realistic examples the plus symbols can outnumber the minus symbols to a high degree as well, and the search for tighter bounds is still going on !

Figure 11: The New "Compensation" Term for Pair (1,3).

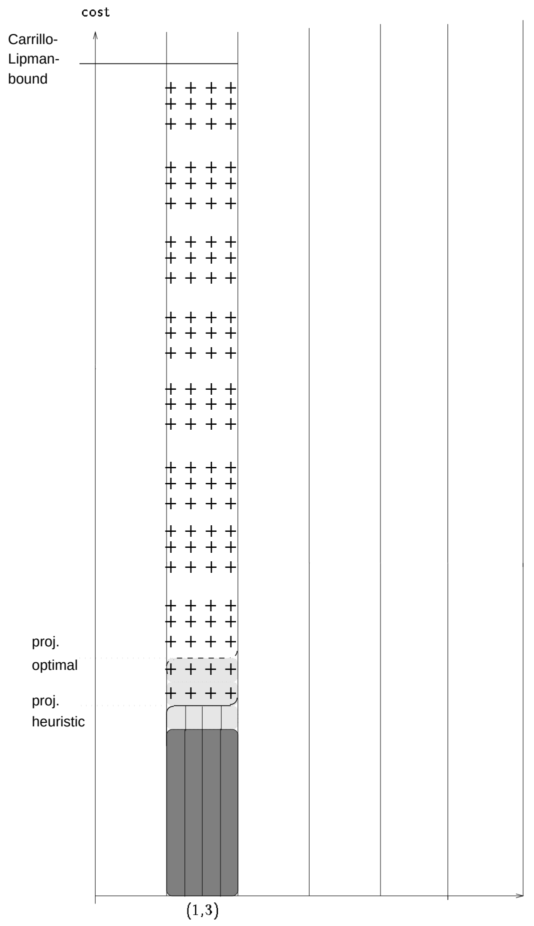


Exercise 22 [10A*] Calculate the new upper bounds for the pairs (1,3) and (2,4).

Exercise 23 [10A] Calculate the new upper bounds for the pairs (1,2) and (2,3).

[Intersect] Having accomplished bounds on the optimal alignment in all its pairwise projections, we have established a *set of constraints* on the path of the optimal alignment through the whole hyperlattice. Only paths that have, in every projection, a cost lower than the bound can be a candidate for an optimal alignment path; only nodes visited by at least one of these paths need to be considered during the calculation of the minimum that is part of the dynamic programming step. In other words, we do not need to look back at the edges coming from the other nodes, as shown in Fig. 7.

Figure 12: The Carrillo-Lipman Bound for Pair (1,3).



Figuratively, the 2-dimensional strips on the faces of the lattice are "projected back", and *intersected*. In Fig. 7(bottom), light sources from the top, right, and front projected shadows of the polyhedron onto the wall. This time, imagine the opposite process, i.e. a back-projection being done by light sources from the bottom, left, and back side shining through the areas *outside* the strips on the wall. Then the darkest region (not illuminated by any of the 3 light sources) is the 3-dimensional polyhedron displayed in the middle that must contain the optimal multiple alignment.

2.2 Mathematical Derivation of the Carrillo-Lipman Bound.

[CLMathUppBnd] We will now follow up on the visual presentation of the last section, and we need to introduce some notation first.

(The following section is not needed for the rest of the chapter; nevertheless it would be a pity if you missed it :-)

Given sequences $s_1, \ldots, s_k$, and a cost function c, let A denote any alignment, A^{o} the optimal one, and $A^{\mathbf{h}}$ a heuristic one. Their costs are $c(A)$, $c(A^{\mathrm{o}}) = d(s_1, \ldots, s_k)$, and $c(A^{\mathbf{h}})$. The projection of a multiple alignment A, in the direction of the sequences s_i and s_j is written as $A_{|i,j}$. Three frequent terms in the following text are $c(A^{\mathrm{o}}{}_{|i,j})$, the cost of the (i,j)-projection of the optimal alignment; $d(s_i, s_j)$, the cost of the pairwise optimal alignment; and $c(A^{\mathbf{h}}{}_{|i,j})$, the cost of the projected heuristic alignment. Our goal is to find for any fixed pair $(p,q), p < q$, a bound on $c(A^{\mathrm{o}}{}_{|p,q})$, in terms of all the other $d(s_i, s_j)$ and $c(A^{\mathbf{h}}{}_{|i,j})$.

Let us express some of our observations of the previous subsection in mathematical terms. First of all, by optimality of $d(s_1, \ldots, s_k)$,

$$c(A^{\mathbf{h}}) - d(s_1, \ldots, s_k) \geq 0.$$

Since we're working in the SP-cost framework, this is equivalent to

$$\sum_{(i,j), i<j} \left(c(A^{\mathbf{h}}{}_{|i,j}) - c(A^{\mathrm{o}}{}_{|i,j}) \right) \geq 0.$$

This is the formal account of the fact that, in Fig. 8, "the areas filled with plus symbols, summed up over *all* columns, are *larger* than the areas filled with minus symbols."

The equation can be rearranged to

$$\left(\sum_{\substack{(i,j), i<j \\ (i,j) \neq (p,q)}} \left(c(A^{\mathbf{h}}{}_{|i,j}) - c(A^{\mathrm{o}}{}_{|i,j}) \right) \right) + c(A^{\mathbf{h}}{}_{|p,q}) - c(A^{\mathrm{o}}{}_{|p,q}) \geq 0,$$

or,

$$\left(\sum_{\substack{(i,j),i<j \\ (i,j)\neq(p,q)}} \Big(c({A^{\mathbf{h}}}_{|i,j}) - c({A^{\mathrm{o}}}_{|i,j}) \Big) \right) + c({A^{\mathbf{h}}}_{|p,q}) \geq c({A^{\mathrm{o}}}_{|p,q}).$$

In Fig. 8 and 9, (p, q) has been chosen as $(1, 3)$. Then, the left-hand-side of the above equation is the bound in Fig. 9, and the right-hand-side is the grey area to be bounded. The bound in Fig. 9 consists of the "compensation", and the term $c({A^{\mathbf{h}}}_{|p,q})$.

By optimality of $d(s_i, s_j)$,

$$c({A^{\mathrm{o}}}_{|i,j}) \geq d(s_i, s_j), \quad for \quad any \quad i, j.$$

This can be put in use in the former equation, yielding the Carrillo-Lipman bound:

$$\left(\sum_{\substack{(i,j),i<j \\ (i,j)\neq(p,q)}} \Big(c({A^{\mathbf{h}}}_{|i,j}) - d(s_i, s_j) \Big) \right) + c({A^{\mathbf{h}}}_{|p,q}) \geq c({A^{\mathrm{o}}}_{|p,q}),$$

for any (p, q) we may have selected.

On the left side, we can find the new "compensation" (as in Fig. 11) plus $c({A^{\mathbf{h}}}_{|p,q})$.

Some authors define an upper and lower bound as follows,

$$U = \sum_{(i,j),i<j} c({A^{\mathbf{h}}}_{|i,j}),$$

$$L = \sum_{(i,j),i<j} d(s_i, s_j).$$

(This upper bound U should not be confused with the "upper bounds" that we've been talking about; these have been the Carrillo-Lipman bounds !)

The Carrillo-Lipman bound now looks like

$$U - L + d(s_p, s_q) \geq c({A^{\mathrm{o}}}_{|p,q}).$$

This equation looks easier, but the term $d(s_p, s_q)$ now appears in *two* places in the equation.

Exercise 24 [00M] Where ?

[CLMathIntersect] Let $X_{i,j}$ be the set of paths which have, in projection (i,j), costs smaller than or equal to $U - L + d(s_i, s_j)$. (We've just "recycled" the indices i and j. They are now used instead of p and q that are no longer needed. This is a kind of standard "procedure" among mathematicians, but may be confusing to others. You will find it in the original papers, too.) 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

$$X = \bigcap_{(i,j), i<j} X_{i,j}.$$

X describes the polyhedron in which we will find the optimal multiple alignment.

Exercise 25 [10M, opt.] How can we convert the set of paths to the set of nodes that defines the polyhedron ?

2.3 Where do we go from here ?

There are a lot of details we're glossing over just because we're using a very simple "sum-of-pairs" cost model. For example, gap costs should be biologically meaningful, and longer gaps therefore penalized less (so-called *subadditivity* of the gap cost function). Also, we would like to score (mis)matches between distant sequences less than (mis)matches between closely related ones. (Usually, an approximate distance between sequences is estimated by doing pairwise alignments, see section 3). All this can be incorporated into the standard dynamic programming algorithm, and it can also be incorporated into the Carrillo-Lipman technique for cutting futile paths through the dynamic programming lattice. Finally, a lot of work has been done on implementing the exploration of the lattice in a time- and space-efficient way, by adopting the *single-source shortest paths* algorithm by Dijkstra, and on a technique called "lower-bound / return-cost pruning" (cf. [GKS95], [Kec95]). Interestingly, the latter technique is more powerful than the one we have discussed here (more futile paths can be cut out), but its implementation consumes more memory per lattice node, and therefore the algorithm "actually runs faster without the [lower-bound] pruning" [GKS95].

Exercise 26 [02A] What happens if you add a bunch of copies of the first sequence to the input of our multiple alignment technique employing the simple SP-cost model ?

Exercise 27 [05M, opt.] In the beginning we said that for the case of 2 sequences, our math boils down to useless, yet obviously true observations. What does the Carrillo-Lipman bound assert for the 2-sequence case ?

2.4 Some Bibliographic Hints.

The standard reference for Carrillo-Lipman is [CaL88], followed up by [AlL89], the latter dealing with the more sophisticated cost model of scoring along a tree. The standard reference for its implementation, known as MSA, is [GKS95]. WWW-server-access, paper and code (Release 2.1) are currently available at http://ibc.wustl.edu/msa.html . The transcript of an online discussion about MSA, with John Kececioglu, is available [Kec95]. [Gus93] introduces an alternative lower bound L'.

3 Heuristic Alignment Procedures and Examples

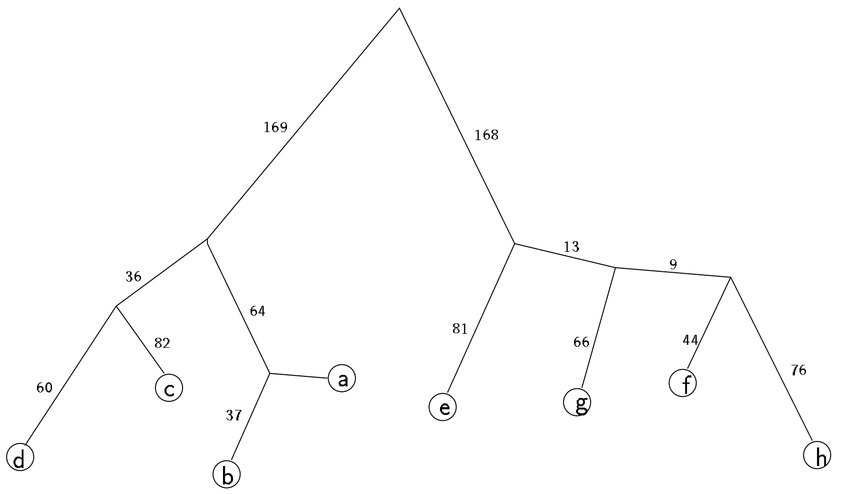
Here is what you will learn in the following sections: You will understand how the most popular Multiple Alignment heuristic works, and following an example, you will investigate optimal, heuristic, and structurally verified multiple alignments obtained from WWW servers, recapitulating results from an original paper.

3.1 Alignment along a Tree.

[AliAlongTree] For more than approximately 8 sequences of average size and similarity, even employing Carrillo-Lipman bounds may not result in a manageable demand on time and memory space, so that an optimal alignment cannot be obtained. (This is the state-of-the-art in 1996.) In such cases, alignment along a tree can be the alternative of choice.

Imagine that you have obtained a phylogenetic tree for the sequences (Fig. 13). This tree may be the result of morphological studies, or it may be obtained from the sequences themselves by one of the methods described in

Figure 13: Phylogenetic Tree.



chapter 4. One popular approach (employed by the Clustal software package, http://dot.imgen.bcm.tmc.edu:9331/multi-align/multi-align-vsns.html) , is the generation of all optimal pairwise alignments, the costs of which form the estimated distances between the sequences. From these distances, a tree can then be obtained.

Exercise 28 [02] How many pairwise comparisons need to be done for k sequences ?

Alignment along a tree is just this; a tree is used to decide about the sequences that shall be aligned first, because of their close relation. After the first step, more sequences are added by aligning them to the existing alignment; we may also align an alignment to an alignment. Alignment along a tree does not necessarily yield an optimal alignment, even if the tree is "perfect". For example, errors may be made in the very first pairwise alignment and they do not get corrected because information from the other sequences is overshadowed during the later steps.

Exercise 29 [02, opt] For which kind of trees may you need to align an alignment to an alignment ? Or, alternatively, for which kind of trees do you *not* need to bother with this ?

The technique for aligning alignments is to simulate standard pairwise alignment, but use *profiles* instead of sequences. For each position, a *profile* holds a list of the relative frequencies (i.e. values between 0 and 1) of the 20 amino acids (and gap), and the cost of matching a position in profile A with a position in B is calculated by multiplying the (mis)match scores, for each pair of amino acids, by the said amino acids' frequencies at these positions, and summing up.

Exercise 30 [05A] Calculate the score of matching the following two positions in profiles A, and B, respectively:

$$\begin{pmatrix} \mathtt{V} \\ \mathtt{V} \\ \mathtt{C} \\ \mathtt{L} \end{pmatrix}, \quad and \quad \begin{pmatrix} \mathtt{P} \\ \mathtt{A} \end{pmatrix}.$$

Use the PAM250 similarity matrix, yielding a similarity score.

Exercise 31 [10M, opt.] Develop the mathematical formula for the alignment of profiles. If you like, begin with the formula for aligning one sequence to a profile. To this end, you need to introduce frequency vectors of length 21, one vector per position of the profile.

Normally, the alignments obtained thus far are fixed; gaps may only be added. Then, we follow the rule "Once a gap, always a gap" [FeD87], also known as "Progressive Alignment".

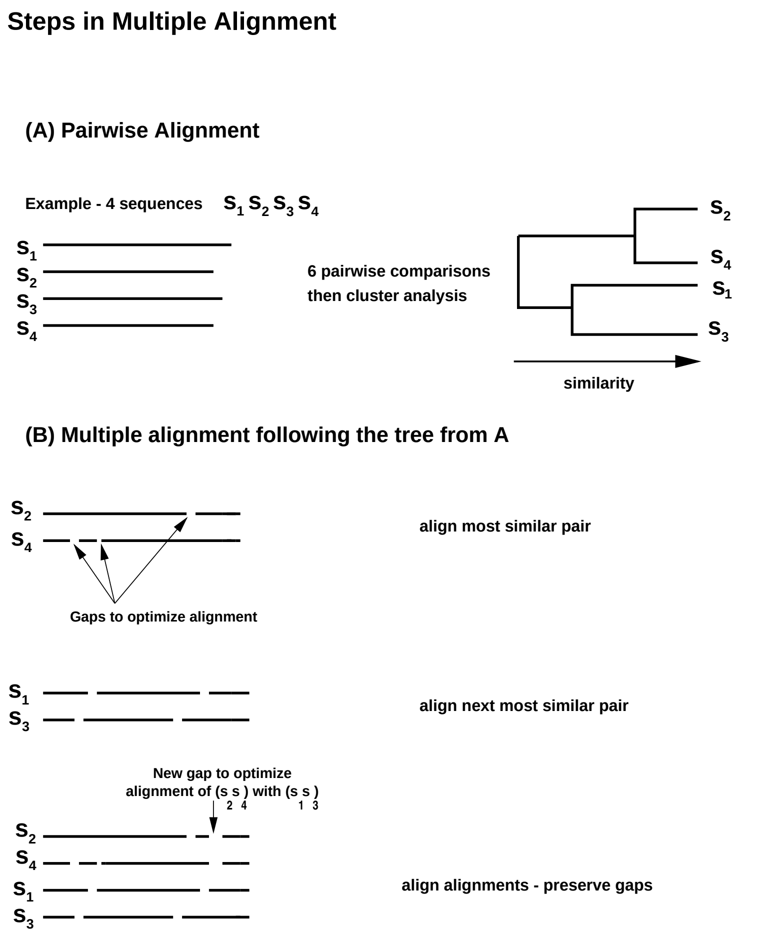
Our technique is illustrated by Fig. 14, adapted with permission from [Bar95], the original of which is available at
`http://geoff.biop.ox.ac.uk/papers/rev93_1/Figure5.ps.`

Some methods (e.g. [BaS87]) do an iterative refinement of the alignment after the initial pass; now gaps may move.

The following concepts may easily be confused:

- Alignment along a tree,
- Scoring along a tree, and
- Tree Alignment.

Figure 14: Progressive Alignment


Steps in Multiple Alignment
(A) Pairwise Alignment
Example - 4 sequences S1 S2 S3 S4
S1
S2
S3
S4
6 pairwise comparisons then cluster analysis
S2
S4
S1
S3
similarity
(B) Multiple alignment following the tree from A
S2
S4
align most similar pair
Gaps to optimize alignment
S1
S3
align next most similar pair
New gap to optimize alignment of (s2 s4) with (s1 s3)
S2
S4
S1
S3
align alignments - preserve gaps

Scoring along a tree is the main alternative to the simple "sum-of-pairs" cost model; only pairs of sequences that are adjacent (neighboring) in the tree are taken into consideration (or, at least they're weighted higher). Indeed, by weighting the pairs differently, we can score along a tree, yet employ Carrillo-Lipman and try out all possibly optimal alignment paths in the hyperlattice, see [AlL89] ! "Tree Alignment" subsumes methods that involve reconstructing ancestral sequences, too.

3.2 A Hands-On Example: Aligning Immunoglobulin Sequences.

[ImmIntro] We will now apply our knowledge about heuristic and optimal alignment methods to a real-life example. The example is more real-life than usual for a textbook; we will deal with a lot of problems you may face in your own investigations, like hard-to-find sequences, inconsistent data, etc. The author hopes that this has got some advantages, too :-)

Our example is taken from the paper "A Strategy for the Rapid Multiple Alignment of Protein Sequences. Confidence Levels from Tertiary Structure Comparisons." by G.J. Barton and M.J.E. Sternberg, J Mol Biol 1987;198:327-337.

We will discuss alignments of the immunoglobulin sequences they are using; fragments of these sequences have already been featured in the introduction.

Exercise 32 [05*] Get the paper ! J Mol Biol, the Journal of Molecular Biology, is an absolute "must" for any university library. Students of the GNA-VSNS Biocomputing Course may receive a copy from the instructors/organizers, if needed. Nevertheless, care has been taken to ensure that the following section is self-contained.

Exercise 33 [10*] Inform yourself about the molecular biology of immunoglobulins; light chain, heavy chain, disulphide bridges, constant region, so-called variable region, and how they fit together. (See also Fig. 15, below.)

3.3 Getting the Immunoglobulin Sequences from the Internet.

[ImmDescr] The Barton & Sternberg paper is now 9 years old; it's from the early days of Multiple Alignment ! 9 years can be a long time for sequences, too, as we will find out really soon.

The authors write the following about their selection of sequences; formatting their description was done by the textbook author. "Eight domains were selected (Brookhaven Data Bank codes).

- Four from **3FAB**:
  `(a) light chain constant region` $C\lambda$ `(FABCL);`

(b) `light chain variable region` $V\lambda$ `(FABVL);`
(c) `heavy chain constant reg. 1` $C\gamma 1$ `(FABCH1);`
(d) `heavy chain variable region` $V\gamma$ `(FABVH).`

- Two from **1FC1**:
  (a) `heavy chain constant reg. 2` $C\gamma 2$ `(FCCH2);`
  (b) `heavy chain constant reg. 3` $C\gamma 3$ `(FCCH3).`

- Two from **1FB4**:
  (a) `light chain variable region` $V\lambda$ `(FB4VL);`
  (b) `heavy chain variable region` $V\gamma$ `(FB4VH).`"

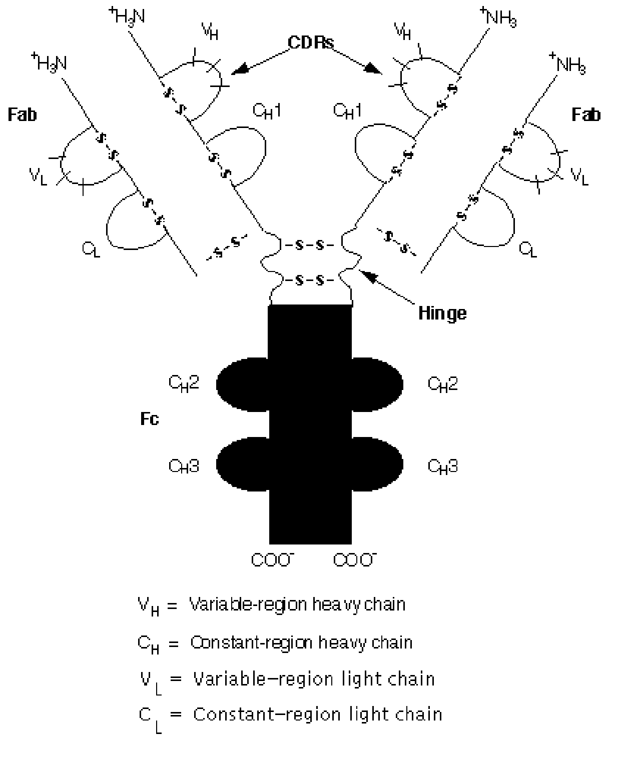
The chains from FAB and FC1 make up one of the identical halves of an antibody; one light ("λ") chain, and one heavy ("γ") chain, the heavy chain consisting of 3, and the light chain consisting of 1 constant region, see Fig. 15. For more details, please try Kevin Shreders's Antibody Resource Page, http://www-chem.ucsd.edu/Faculty/goodman/antibody.html/abpage.html, in particular the link to Mike Clark's page featuring Images of Immunoglobulin Molecules. As an example of a relevant database, you may explore the Kabat Database of Sequences of Proteins of Immunological Interest, http://immuno.bme.nwu.edu/ .

The FB4 regions are added to the collection in order to have an equal amount of variable and constant regions. Let me stress that the "variable" regions get their name from the antigen-binding subregions ("CDRs", complementarity- determining regions), which are composed of just a few amino acids each, and give the antibody its specificity. Most of the variable region of an antibody is about as conserved as the constant regions are !

Exercise 34 [5, opt.] Using the Molecules R Us server, http://molbio.info.nih.gov/cgi-bin/pdb , get some images of the 1FC1 immunoglobulin.

Exercise 35 [15, opt.] Using technology from the VSNS-PPS course, http://www.cryst.bbk.ac.uk/PPS/index.html, you can take a closer look at 1FC1. (This may take some time, though, if you need

Figure 15: Schematic Structure of an Antibody (Immunoglobulin)


^+H_3N
V_H
CDRs
V_H
$^+NH_3$
^+H_3N
$^+NH_3$
Fab
C_H1
C_H1
Fab
V_L
V_L
C_L
-s-s-
-s-s-
C_L
Hinge
C_H2
C_H2
Fc
C_H3
C_H3
COO^-
COO^-
V_H = Variable-region heavy chain
C_H = Constant-region heavy chain
V_L = Variable-region light chain
C_L = Constant-region light chain

to install software, etc. Right now, the GNA-VSNS Biocomputing Course organizers have not got enough time resources to help you intensively.)

In the alignments from the paper and from our introduction, the sequences are arranged as follows:

- Variable regions:
  (BS1) 3FAB light chain variable region $V\lambda$ (FABVL);
  (BS2) 1FB4 light chain variable region $V\lambda$ (FB4VL);
  (BS3) 1FB4 heavy chain variable region $V\gamma$ (FB4VH);
  (BS4) 3FAB heavy chain variable region $V\gamma$ (FABVH).

- Constant regions:
  (BS5) 1FC1 heavy chain constant reg. 2 $C\gamma2$ (FCCH2);
  (BS6) 3FAB light chain constant region $C\lambda$ (FABCL);

```
(BS7) 3FAB heavy chain constant reg. 1 Cγ1 (FABCH1);
(BS8) 1FC1 heavy chain constant reg. 3 Cγ3 (FCCH3).
```

The arrangement of the variable and constant sets is done to maximize similarity of adjacent sequences: both FB4 variable regions go together, and both FAB constant regions go together. We will use this numbering (BS1–BS8) throughout.

Up until the beginning of the next subsection (3.4) the following is an optional part of the chapter, in which you will retrieve the sequences from the net, and check your results.

[ImmRetrieval] **Exercise 36 [15, opt.]** For this exercise, note that there are quite a lot of differences between the sequences you retrieve and the sequences from the paper. What's more, the sequences will be different depending on the data bank you searched ! But don't despair, you will have a scout with you !
Obtain the 8 immunoglobulin sequences, using what you learned in chapter 2. If you've not read chapter 2 (What a shame !), start with Pedro's list, http://www.public.iastate.edu/˜ pedro/research_tools.html and try out the various PDB resources. Hint: 2 of the entries have been superseded, and once you know the new entry IDs, you can search via SRS-WWW, http://www.embl-heidelberg.de/srs/srsc. If you'd like to obtain sequences with the one-letter code directly, (and you want to end up with exactly the same sequences as the author), you can access a nice databank for this via SRS: PDBFINDER. PDBFINDER however does not distinguish variable and constant regions; they are just concatenated ! (But you don't need to worry about this.)

Let's take a look at the 3 PDBFINDER files you retrieved:
`http://www.embl-heidelberg.de/srs/srsc?[PDBFINDER-id:7FAB]`
`http://www.embl-heidelberg.de/srs/srsc?[PDBFINDER-id:1FC1]`
`http://www.embl-heidelberg.de/srs/srsc?[PDBFINDER-id:2FB4]`
They're quite regular, 7FAB and 2FB4 listing one heavy and one light chain each, and 1FC1 listing 2 identical chains A and B.

Exercise 37 [05B*] Why are chains A and B identical ?

Exercise 38 [05B*] "Why do light and heavy chains suddenly have (approximately) the same length in 7FAB and 2FB4 ? I thought, the *heavy* chain is twice as long ?!"

[ImmRetrievalVerif] We will now do a plausibility check on whether we've retrieved the right sequences. To this end, we'll align the fragments from the introduction (they are listed in the order BS1–BS8, taken directly from the paper) to the retrieved sequences. Variable and constant regions are still stuck together !

Exercise 39 **[10]** Using the Clustal Query Form, http://dot.imgen.bcm.tmc.edu:9331/multi-align/multi-align-vsns.html, align the fragments with the chains. *Note that the above query provides a Clustal Interface with the 1995 default parameters, so that your alignments match exactly the ones cited in this text ! If you use the standard BCM Launcher page, you will get different results.* Your Query, in Fasta-Format, should look like:

```
>7FAB_light_chain
ASVLTQPPSVSGAPGQRVTISCTGSSSNIGAGHNVKWYQQLPGTAPKLLIFHNNARFSVSKSGTSATLAITGLQAEDEAD
YYCQSYDRSLRVFGGGTKLTVLRQPKAAPSVTLFPPSSEELQANKATLVCLISDFYPGAVTVAWKADGSPVKAGVETTTP
SKQSNNKYAASSYLSLTPEQWKSHKSYSCQVTHEGSTVEKTVAP
>2FB4_light_chain
QSVLTQPPSASGTPGQRVTISCSGTSSNIGSSTVNWYQQLPGMAPKLLIYRDAMRPSGVPDRFSGSKSGASASLAIGGLQ
SEDETDYYCAAWDVSLNAYVFGTGTKVTVLGQPKANPTVTLFPPSSEELQANKATLVCLISDFYPGAVTVAWKADGSPVK
AGVETTKPSKQSNNKYAASSYLSLTPEQWKSHRSYSCQVTHEGSTVEKTVAPTECS
>2FB4_heavy_chain
EVQLVQSGGGVVQPGRSLRLSCSSSGFIFSSYAMYWVRQAPGKGLEWVAIIWDDGSDQHYADSVKGRFTISRNDSKNTLF
LQMDSLRPEDTGVYFCARDGGHGFCSSASCFGPDYWGQGTPVTVSSASTKGPSVFPLAPSSKSTSGGTAALGCLVKDYFP
QPVTVSWNSGALTSGVHTFPAVLQSSGLYSLSSVVTVPSSSLGTQTYICNVNHKPSNTKVDKRVEPKSC
>7FAB_heavy_chain
AVQLEQSGPGLVRPSQTLSLTCTVSGTSFDDYYWTWVRQPPGRGLEWIGYVFYTGTTLLDPSLRGRVTMLVNTSKNQFSL
RLSSVTAADTAVYYCARNLIAGGIDVWGQGSLVTVSSASTKGPSVFPLAPTAALGCLVKDYFPEPVTVSWNSGALTSGVH
TFPAVLQSSGLYSLSSVVTVPSSSLGTQTYICNVNHKPSNTKVDKKVEP
>1FC1
PSVFLFPPKPKDTLMISRTPEVTCVVVDVSHEDPQVKFNWYVDGVQVHNAKTKPREQQYNSTYRVVSVLTVLHQNWLDGK
EYKCKVSNKALPAPIEKTISKAKGQPREPQVYTLPPSREEMTKNQVSLTCLVKGFYPSDIAVEWESNGQPENNYKTTPPV
LDSDGSFFLYSKLTVDKSRWQQGNVFSCSVMHEALHNHYTQKSLSLS
>BS1-fragment
VTISCTGSSSNIGAGNHVKWYQQLPG
>BS2-fragment
VTISCTGTSSNIGSITVNWYQQLPG
>BS3-fragment
LRLSCSSSGFIFSSYAMYWVRQAPG
>BS4-fragment
```

```
LSLTCTVSGTSFDDYYSTWVRQPPG
>BS5-fragment
PEVTCVVVDVSHEDPQVKFNWYVDG
>BS6-fragment
ATLVCLISDFYPGAVTVAWKADS
>BS7-fragment
AALGCLVKDYFPEPVTVSWNSG
>BS8-fragment
VSLTCLVKGFYPSDIAVEWESNG
```

Here is the result you will get:

```
Page 1.1
               1             15 16            30 31            45 46            60 61            75 76            90
 1 7FAB_light --------------- --------------- --------------- --------------- --------------- ---------------
 2 2FB4_light QSVLTQPPSASGTPG QRVTISCSGTSSNIG SSTVNWYQQLPGMAP KLLIYRDAMRPSGVP DRFSGSKSGASASLA IGGLQSEDETDYYCA     9
 3 2FB4_heavy --------------- --------------- --------------- --------------- --------------- ---------------
 4 7FAB_heavy --------------- --------------- --------------- --------------- --------------- ---------------

 5 1FC1       --------------- --------------- --------------- --------------- --------------- ---------------
 6 BS1-fragme --------------- --------------- --------------- --------------- --------------- ---------------
 7 BS2-fragme --------------- --------------- --------------- --------------- --------------- ---------------
 8 BS3-fragme --------------- --------------- --------------- --------------- --------------- ---------------

 9 BS4-fragme --------------- --------------- --------------- --------------- --------------- ---------------
10 BS5-fragme --------------- --------------- --------------- --------------- --------------- ---------------
11 BS6-fragme --------------- --------------- --------------- --------------- --------------- ---------------
12 BS7-fragme --------------- --------------- --------------- --------------- --------------- ---------------

13 BS8-fragme --------------- --------------- --------------- --------------- --------------- ---------------


Page 2.1
               91           105 106          120 121          135 136          150 151          165 166          180
 1 7FAB_light --------------- -----------ASVL TQPPSVSGAPGQRVT ISCTGSSSNIGAG-H NVKWYQQLPGTAPKL LIFHNNARFSVSKSG     6
 2 2FB4_light AWDVSLNAYVFGTGT KVTVLGQPKANPTVT LFPPSSEELQANKAT LVCLISDFYPGA--V TVAWKADGSPVKAGV ETTKPSKQSNNKYAA    17
 3 2FB4_heavy --------------- -----------EVQL VQSGGGVVQPGRSLR LSCS-SSGFIFSS-Y AMYWVRQAPGKGLEW VAIIWDDGSDQHYAD     6
 4 7FAB_heavy --------------- -----------AVQL EQSGPGLVRPSQTLS LTCT-VSGTSFDD-Y YWTWVRQPPGRGLEW IGYVFYTG-------     5

 5 1FC1       --------------- ---------PSVFLF PPKPKDTLMISRTPE VTCVVVDVSHEDPQV KFNWYVDGVQVHNAK TKPREQQYNSTYRVV     6
 6 BS1-fragme --------------- --------------- -------------VT ISCTGSSSNIGAG-N HVKWYQQLPG----- ---------------     2
 7 BS2-fragme --------------- --------------- -------------VT ISCTGTSSNIGS--I TVNWYQQLPG----- ---------------     2
 8 BS3-fragme --------------- --------------- -------------LR LSCS-SSGFIFSS-Y AMYWVRQAPG----- ---------------     2

 9 BS4-fragme --------------- --------------- -------------LS LTCT-VSGTSFDD-Y YSTWVRQPPG----- ---------------     2
10 BS5-fragme --------------- --------------- -------------PE VTCVVVDVSHEDPQV KFNWYVDG------- ---------------     2
11 BS6-fragme --------------- --------------- -------------AT LVCLISDFYPGA--V TVAWKADS------- ---------------     2
12 BS7-fragme --------------- --------------- -------------AA LGCL-VKDYFPEP-V TVSWN---SG----- ---------------     2

13 BS8-fragme --------------- --------------- -------------VS LTCLVKGFYPSD--I AVEWESNG------- ---------------     2

(continues alignment of full chains)
```

Exercise 40 [00B] Why did we not just use our text editor to find the fragments in the sequences ?

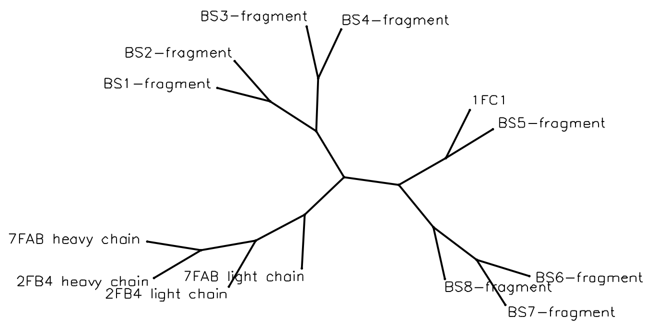
Exercise 41 **[00]** Why did we not use a local alignment technique like the one presented in chapter 1 ? That would have worked much better !

[ImmRetrievalVerifDisc] Let us interpret the Clustal Alignment. First of all, 2FB4 light got shifted; its *constant* region seems to be very similar to the *variable* regions of other chains !

Exercise 42 **[02B]** How do we know that the PDBFINDER files list the *variable* region followed by the *constant* region, and not vice versa ?

BS1, BS3 and BS4 align as expected to the variable regions of the 7FAB / 2FB4 chains. (There is an `HN/NH` difference between fragment BS1 and 7FAB, at positions 150-151, Also, Pos. 152 is inconsistent for BS4.) BS2 is supposed to align with the variable chain of 2FB4 light, but it doesn't ! Indeed, taking a look at the tree used by Clustal (Fig. 16), we see that it aligns the *profiles* containing BS2 and 2FB4 light at a rather late stage, so that BS2's high similarity (not identity, due to whatever errors) with the subsequence `VTISCSGTSSNIG SSTVNWYQQLPG` in the 2FB4 light variable region (pos. 18-42) has been overshadowed during profile alignment.

Figure 16: Phylogenetic Tree used by Clustal.



Next, observe that BS5 and BS6 are aligned properly to 1FC1 and 2FB4 light, respectively. For BS5 that's OK (1FC1, as we've said in the beginning, is indeed the concatenation of the heavy chain's second and third constant region, and BS5 is a fragment from the second constant region; see also Fig. 15.) For BS6, this is a little miracle; after all, it aligns to the *constant* region of 2FB4 light, which is *not* in our collection of 8 immunoglobulin sequences !

Exercise 43 [02] So which two constant regions have an identical (sub)-sequence ? If you're not sure, use your text editor, searching this text for even smaller fragments like "`LVCL`". Can you find a reason for this identity ?

BS7 and BS8 are obviously misaligned; you will find copies of them at the end of 7FAB heavy, and 1FC1, in the constant regions (these ends are cut away in the Clustal Alignment shown.)

The following 3 exercises are concerned with some problems you encounter when doing databank retrieval.

Exercise 44 [05, opt.] Find sequence 7FAB light (variable region) in SwissProt and confirm that it's got `NH` again, just as in fragment BS1, pos. 150-151. (I'm not exactly sure about the difference between "IG LAMBDA CHAIN V-VI REGION (NIG-48)" and "IG LAMBDA CHAIN V-I REGION (NEWM)". The latter is the correct one. If you've got the paper handy, you will note that this sequence is exactly the one from the paper, whereas the one we obtained from PDBFINDER differs in 4 positions !) If you want to detect further problems, you can go on and retrieve what seem to be the SwissProt equivalents of the 2FB4 heavy chain variable region (BS3) and 7FAB heavy chain variable region (BS4). Now they're farther away; I guess this must have got something to do with the labels "V-III(KOL)" and "V-II(NEWM)" of the SwissProt sequences. Can an immunologist help out ?

Exercise 45 [25, opt.] Try to find the 2FB4 light variable region in another databank, i.e. not in PDB/PDBFINDER. This seems to be a challenge, and I couldn't find it, not even using Blast/Fasta searches. If you find it, or know why it's not in SwissProt, the author will email you a beer !

Exercise 46 [15, opt.] Try to find the 7FAB light constant region in another databank, i.e. not in PDB/PDBFINDER. Another challenge !

If you've done the last few exercises, you have got some justification to cite for your cutting point between the variable and constant regions of our sequences; equivalently you could have searched for the respective constant regions in SwissProt. This would give you the "exact" cutting points *between* the constant regions, too; BS7, BS5, and BS8 are in one SwissProt file (why ?), and the headers list the exact cut-points ! Or you can be as lazy as the author and use the cut-marks employed in the Barton & Sternberg paper, as follows. (For perfectionists, BS6 and BS7 got a few residues added, and BS8 got one residue deleted. Now they've got the same length as the ones in the original paper.)

```
>BS1, 7FAB light chain variable region
ASVLTQPPSVSGAPGQRVTISCTGSSSNIGAGHNVKWYQQLPGTAPKLLIFHNNARFSVSKSGTSATLAITGLQAEDEAD
YYCQSYDRSLRVFGGGTKLTVLR
>BS2, 2FB4 light chain variable region
QSVLTQPPSASGTPGQRVTISCSGTSSNIGSSTVNWYQQLPGMAPKLLIYRDAMRPSGVPDRFSGSKSGASASLAIGGLQ
SEDETDYYCAAWDVSLNAYVFGTGTKVTVLGQ
>BS3, 2FB4 heavy chain variable region
EVQLVQSGGGVVQPGRSLRLSCSSSGFIFSSYAMYWVRQAPGKGLEWVAIIWDDGSDQHYADSVKGRFTISRNDSKNTLF
LQMDSLRPEDTGVYFCARDGGHGFCSSASCFGPDYWGQGTPVTVSS
>BS4, 7FAB heavy chain variable region
AVQLEQSGPGLVRPSQTLSLTCTVSGTSFDDYYWTWVRQPPGRGLEWIGYVFYTGTTLLDPSLRGRVTMLVNTSKNQFSL
RLSSVTAADTAVYYCARNLIAGGIDVWGQGSLVTVSS
>BS5, 1FC1 heavy chain constant region
PSVFLFPPKPKDTLMISRTPEVTCVVVDVSHEDPQVKFNWYVDGVQVHNAKTKPREQQYNSTYRVVSVLTVLHQNWLDGK
EYKCKVSNKALPAPIEKTISKAKG
>BS6, 7FAB light chain constant region
QPKAAPSVTLFPPSSEELQANKATLVCLISDFYPGAVTVAWKADGSPVKAGVETTTPSKQSNNKYAASSYLSLTPEQWKS
HKSYSCQVTHEGSTVEKTVAPtscs
>BS7, 7FAB heavy chain constant region
ASTKGPSVFPLAPTAALGCLVKDYFPEPVTVSWNSGALTSGVHTFPAVLQSSGLYSLSSVVTVPSSSLGTQTYICNVNHK
PSNTKVDKKVEPksa
>BS8, 1FC1 heavy chain constant region
QPREPQVYTLPPSREEMTKNQVSLTCLVKGFYPSDIAVEWESNGQPENNYKTTPPVLDSDGSFFLYSKLTVDKSRWQQGN
VFSCSVMHEALHNHYTQKSLSL
```

3.4 Optimal, Heuristic, and Structurally Verified Alignments of the Immunoglobulin Sequences.

[ImmMsa] Now we've finally retrieved and confirmed the data (see the listing above, i.e. at the end of the previous section) and can start aligning ! These data are not exactly the ones from the paper, due to whatever errors, but they're close enough so that we can "reproduce" a few results from the Barton & Sternberg paper. This "reproduction" will be qualitative, because we've mainly got Clustal at our disposal. Clustal aligns along a tree, whereas Barton & Sternberg add one sequence at a time to a "growing" profile.

Let's start with an MSA alignment, since we cannot do any better than optimal if the underlying cost model is appropriate. However, we cannot get an MSA alignment on the net at the Washington University MSA server, http://ibc.wustl.edu/msa.html; the process gets killed after some time, we're using up too many resources ! All we can get is the heuristic alignment calculated by MSA (see 3.6).

Do not overload the Washington University MSA server by trying the MSA alignment yourself ! This time, only use the form with the "optimal alignment" option set to "off". If you've got MSA 1.0 or the newly released MSA 2.1 at your computer, and nobody's watching, you can try it out. Even with MSA 2.1, the author's workstation couldn't finish the job. I guess a supercomputer is needed ?!

Exercise 47 [10*] Nevertheless, the author has obtained an alignment from the MSA server that he believes is "optimal". Very simple trick, explained a little later ;-)

So, here's the optimal MSA alignment (well, sort-of...).

```
-----asVLTQPPsvsgapgqrvTISCTGsssnigag-hNVKWYqqlpgtapk--llifhnn----------arf
-----qsVLTQPPsasgtpgqrvTISCSGtssnigs--sTVNWYqqlpgmapk--lliyrda---mrpsgvpdrf
-----evQLVQSGggvvqpgrslRLSCSSsgfifss--yAMYWVrqapgkglewvaiiwddgsdqhyadsvkgrf
-----avQLEQSGpglvrpsqtlSLTCTVsgtsfdd--yYWTWVrqppgrglewigyvfytg-ttlldpslrgrv
-p--SVFLFPpkpkdtlmisrtpEVTCVVvdvshedpqvKFNWYvd--gvqvh--naKTKPR----------eqq
qpkaapSVTLFPpsseelqankaTLVCLIsdfypga--vTVAWKadg-spvka--GVETTtp----------skq
--------astkgpSVFPLAptaALGCLVkdyfpep--vTVSWNs---galts--GVHTFpa----------vlq
qpr-epQVYTLPpsreemtknqvSLTCLVkgfypsd--iAVEWEsn--gqpen--NYKTTpp----------vld

SVSKSgTSAT--LAItglqaedeadYYC--QSYdr--------slr--VFGggtkltvlr-
SGSKSgASAS--LAIgglqsedetdYYC--AAWdv--------slnayVFGtgtkvtvlgq
TISRNdskNTLFLQMdslrpedtgvYFCARDgghgfcssascfgpd--YWGqgtpvtvss-
TMLVNtskNQFSLRLssvtaadtavYYCARNliag--------gid--VWGqgslvtvss-
ynstyrVVSV--LTVlhqnwldgkeYKC--KVSnk--------alp--apIEKtiskakg-
snnkyaASSY--LSLtpeqwkshksYSC--QVThe--------gst----VEKtvaptscs
ssglysLSSV--VTV-pssslgtqtYIC--NVNhk--------psn--tkVDKkvepksa-
sdgsffLYSK--LTVdksrwqqgnvFSC--SVMhe--------alh--nhyTQKslsl---
```

We can easily recognize the correct alignment of the two Cysteine residues, and the Tryptophane. So this alignment is at least not completely off, i.e. it reproduces some features that a working immunologist easily recognizes.

(Capitalized residues are part of the structurally verified alignment, see below.)

Exercise 48 [05, opt.] Get a colorful visualisation of the alignment, by using the Weblogo server,
http://www.bio.cam.ac.uk/cgi-bin/seqlogo/logo.cgi. For your convenience, the Fasta format of our alignment is available, see 3.7.

Although this does not do justice to Tom Schneider's "Sequence Logo" theory, we just note that the large characters in the Weblogo output denote the conserved residues.

[ImmMsaEpsAll] The "optimal" alignment is pretty much different from the heuristic one calculated by MSA before bogging down (see 3.6). Indeed, the polyhedron that needs to be explored is huge, as you can see from looking at the differences between the projected heuristic and the optimal pairwise alignments. (These differences give rise to the "compensation term" that is used to establish the Carrillo-Lipman bounds that in turn influence the polyhedron. See section 2.1 on the theory of the Carrillo-Lipman Bound). They are called "epsilon" in the following MSA 2.0 printout (this was printed out before the author's computer started the insurmountable task of exploring the polyhedron). "I" and "J" are, of course, the direction of the projection for which the difference is given.

```
----Estimated epsilons----
I = 1  J = 2  epsilon =  8
I = 1  J = 3  epsilon = 50
I = 1  J = 4  epsilon = 34
I = 1  J = 5  epsilon = 50
I = 1  J = 6  epsilon = 50
I = 1  J = 7  epsilon = 28
I = 1  J = 8  epsilon = 50
I = 2  J = 3  epsilon = 50
I = 2  J = 4  epsilon = 26
I = 2  J = 5  epsilon = 50
I = 2  J = 6  epsilon = 50
I = 2  J = 7  epsilon = 34
I = 2  J = 8  epsilon = 50
I = 3  J = 4  epsilon =  5
I = 3  J = 5  epsilon = 50
I = 3  J = 6  epsilon = 50
```

```
I = 3  J = 7  epsilon = 50
I = 3  J = 8  epsilon = 50
I = 4  J = 5  epsilon = 50
I = 4  J = 6  epsilon = 50
I = 4  J = 7  epsilon = 50
I = 4  J = 8  epsilon = 50
I = 5  J = 6  epsilon =  5
I = 5  J = 7  epsilon = 50
I = 5  J = 8  epsilon = 25
I = 6  J = 7  epsilon =  9
I = 6  J = 8  epsilon = 22
I = 7  J = 8  epsilon = 43
```

Exercise 49 [05, opt.] What does `epsilon = 5` mean ? 5 units of what ? How has this been standardized ? (Hint: See the next exercise.)

Epsilon = 50 is a threshold, larger values are just cut ! Therefore, it is possible that even if the computer were not bogged down, the full-size polyhedron would not have been explored, and the alignment would not necessarily have been optimal. The reason for all our trouble is now becoming clear: Our immunoglobulin sequences are too dissimilar to even suggest a heuristic alignment that is indeed close to the optimal one; "expert knowledge" at least about the Cys and Trp (W) residues is needed. We will soon see that the "optimal" MSA alignment (which the author obtained by cutting all sequences in two parts, and piecing the alignments together :-) is approximately as far away from the "biological truth" as MSA's heuristic one, and Clustal's (see below).

[ImmMsaEpsPartial] Although the MSA *server* does not inform you about the "epsilon"- values if the process gets killed, you can still get an idea of these values yourself, by submitting subsets. For example, aligning the constant regions only, the following information is returned (the alignment will be displayed and discussed below.)

```
Costfile:                    pam250
Alignment cost:   13103      Lower bound:   12933
Delta:              170      Max. Delta:      199

Sequences  Proj. Cost  Pair. Cost  Epsilon  Max. Epsi.  Weight  Weight*Cost
  1  2        1672        1670         2        19         1        1672
  1  3        1624        1622         2         5         1        1624
  1  4        1656        1656         0         8         2        3312
  2  3        1633        1586        47        41         2        3266
  2  4        1608        1592        16        27         1        1608
  3  4        1621        1565        56        50         1        1621
Elapsed time =    1.469
```

The quantity called `epsilon` *in the last table is now called* `Max. Epsi.` *!*

Exercise 50 [05A] Given the information above, it's now easier to answer the question "What does `epsilon = 5` mean ?" (see the last exercise).

Exercise 51 [05*A] Which quantities do `Lower bound`, `Proj. Cost`, `Pair. Cost`, `Delta` and `Max. Delta` represent ? Hint: The Delta's have to do with the epsilons, just looking at their size.

Exercise 52 [10A] Calculate the Carrillo-Lipman bound for pair (1,2), under the assumption that the difference between the costs of the projected heuristic and the pairwise optimal alignment for pair (3,4) is indeed 50 (i.e. that no cutting down to 50 took place). Pairs (1,4) and (2,3) have weight 2 !

[ImmStructVerif] Looking at the 3-dimensional structures of our protein domains, experts have derived so-called "structurally verified alignments" for parts of them (called "motifs" hereafter). The following are listed in the Barton & Sternberg paper; they correspond to the different β-chains of the

immunoglobulin domains, and will be taken as the "standard of truth".

A	B	C	D	E	F	G
VLTQPP	TISCTG	NVKWY	SVSKS	TSATLAI	YYCQSY	VFG
VLTQPP	TISCSG	TVNWY	SGSKS	ASASLAI	YYCAAW	VFG
QLVQSG	RLSCSS	AMYWV	TISRN	NTLFLQM	YFCARD	YWG
QLEQSG	SLTCTV	YWTWV	TMLVN	NQFSLRL	YYCARN	VWG
SVFLFP	EVTCVV	KFNWY	KTKPR	VVSVLTV	YKCKVS	IEK
SVTLFP	TLVCLI	TVAWK	GVETT	ASSYLSL	YSCQVT	VEK
SVFPLA	ALGCLV	TVSWN	GVHTF	LSSVVTV	YICNVN	VDK
QVYTLP	SLTCLV	AVEWE	NYKTT	LYSKLTV	FSCSVM	TQK

Exercise 53 [10*] Looking at the "optimal" MSA alignment, which β-chains were aligned correctly ? How many residues were misaligned ? For the latter, count residues as misaligned if they don't align with the majority of residues that are following the column of a motif, and count all residues if the column got completely scrambled (i.e. if there are no two residues that are aligned according to the motif). In other words, for all of the 38 columns displayed above, look whether you can at least identify a relative majority of residues aligned in the same way, and count those residues that are not aligned to them.

Exercise 54 [05] Take a look at MSA's heuristic alignment (see 3.6), and/or its Weblogo diagram. Compared to the "standard-of-truth" data, which seemingly conserved residue is just an artifact, i.e. the result of misalignments ?

[ImmClustal] Here is the Clustal alignment (again using the BCM Search Launcher,
http://dot.imgen.bcm.tmc.edu:9331/multi-align/multi-align-vsns.html, 1995 default settings), for comparison. The motifs are given in capital letters, but it's nevertheless a good idea to print out the alignment, and put the beta-

sheets into boxes in the same way as in the Barton & Sternberg paper (p.331).

```
-----asVLTQPPsv--sgapgqrvTISCTGsssnigag-hNVKWYqqlpg--tapkllifhnnar--------
-----qsVLTQPPsa--sgtpgqrvTISCSGtssnigs--sTVNWYqqlpg--mapklliyrdamrpsgvpdr--
-----evQLVQSGgg--vvqpgrslRLSCS-Ssgfifss-yAMYWVrqapgkglewvaiiwddgsdqhyadsvkg
-----avQLEQSGpg--lvrpsqtlSLTCT-Vsgtsfdd-yYWTWVrqppgrglewigyvfytg-ttlldpslrg
-----pSVFLFPpkpkdtlmisrtpEVTCVVvdvshedpqvKFNWYvd--g----vqvhnaKTKPReqq------
qpkaapSVTLFPpss--eelqankaTLVCL-Isdfypga-vTVAWKad--g---spvkaGVETTtpsk-------
-------astkgpS---VFPLAptaALGCL-Vkdyfpep-vTVSWNsg-------altsGVHTFpavlq------
qpre-pQVYTLPpsr--eemtknqvSLTCL-Vkgfypsd-iAVEWEsn--g----qpenNYKTTppvld------

-fSVSK--SgTSATLAItglqaedeadYYCQ--------SYdrslr--VFGggtkltvlr-
-fSGSK--SgASASLAIgglqsedetdYYCA--------AWdvslnayVFGtgtkvtvlgq
rfTISRNdskNTLFLQMdslrpedtgvYFCARDgghgfcssascfgpdYWGqgtpvtvss-
rvTMLVNtskNQFSLRLssvtaadtavYYCARN--------liaggidVWGqgslvtvss-
--ynst--yrVVSVLTVlhqnwldgkeYKCK----------VSnkalpapIEKtiskakg-
-qsnnk--yaASSYLSLtpeqwkshksYSCQ----------VThegstVEKtvaptscs--
---ssg--lysLSSVVTVpssslgtqtYICN----------VNhkpsntkVDKkvepksa-
--sdgs--ffLYSKLTVdksrwqqgnvFSCS----------VMhea--lhnhyTQKslsl-
```

Exercise 55 [05, opt.] Obtain the Clustal alignment from the WWW.

Exercise 56 [05A] In the Clustal alignment, which motifs were aligned correctly ? How many residues were misaligned ? (See Exercise 53.)

Exercise 57 [10, opt.] Find out about the tree along which Clustal did the alignment. (Unfortunately, the WWW Forms I know do not return a picture of the tree along which Clustal aligned; you need to interpret or convert the text description of the tree returned by the Washington University server, unless you have Clustal/Phylip on your computer. Alternatively, you may look at the tree from ETH Zurich's All-All service,
http://cbrg.inf.ethz.ch/subsection3_1_1.html, i.e. Fig. 13. Topologically, it's the same as the Clustal tree.)

[ImmMsaPartial] Let us now align variable and constant regions alone;

```
asVLTQPPsvsgapgqrvTISCTGsssnigaghNVKWYqqlpgtapkll--ifhnn----------arfSVSKSg
qsVLTQPPsasgtpgqrvTISCSGtssnig-ssTVNWYqqlpgmapkll--iyrda---mrpsgvpdrfSGSKSg
evQLVQSGggvvqpgrslRLSCSSsgfifs-syAMYWVrqapgkglewvaiiwddgsdqhyadsvkgrfTISRNd
avQLEQSGpglvrpsqtlSLTCTVsgtsfd-dyYWTWVrqppgrglewigyvfytg-ttlldpslrgrvTMLVNt

TSAT--LAItglqaedeadYYCQS--------Ydrslr--VFGggtkltvlr-
ASAS--LAIgglqsedetdYYCAA--------WdvslnayVFGtgtkvtvlgq
skNTLFLQMdslrpedtgvYFCARDgghgfcssascfgpdYWGqgtpvtvss-
skNQFSLRLssvtaadtavYYCAR--------NliaggidVWGqgslvtvss-
```

is the optimal MSA alignment of the variable regions, and

```
-----pSVFLFPpkpkdtlmisrtpEVTCVVvdvshedpqvKFNWYvdgvqv-hnaKTKPReqqynstyrVVSVL
qpkaapSVTLFPpssee--lqankaTLVCLIsdfypga--vTVAWKadgspvkaGVETTtpskqsnnkyaASSYL
----------astkgpSVFPLAptaALGCLVkdyfpep--vTVSWNsgalt--sGVHTFpavlqssglysLSSVV
qpre-pQVYTLPpsree--mtknqvSLTCLVkgfypsd--iAVEWEsngqpe-nNYKTTppvldsdgsffLYSKL

TVlhqnwldgkeYKCKVSnkalpapIEKtiskakg-
SLtpeqwkshksYSCQVTheg--stVEKtvaptscs
TV-pssslgtqtYICNVNhkpsntkVDKkvepksa-
TVdksrwqqgnvFSCSVMhealhnhyTQKslsl---
```

is the optimal MSA alignment of the constant regions. Calculating the accuracy for these 2 alignments separately, we count 9 misaligned residues in the variable regions, and 14 in the constant regions, from a total of 152 each. Modifying the accuracy scores of the 8-sequence-alignments (see exercise 53), counting only misaligned amino acids within one group (either constant, or variable), we obtain 14 and 20 errors, respectively. These alignments were made with the help of the other group of sequences, and in fact multiple alignment deteriorates accuracy scores ! Barton & Sternberg perform a more detailed analysis, comparing the scores of *all pairwise* alignments within one group (without taking the other sequences into consideration) with the accuracy obtained from the 8-sequence alignment, and observe the same deterioration.

[ImmClustalPartial] The same phenomenon can be observed using Clustal alignments, viz.

```
asVLTQPPsvsgapgqrvTISCTGsssnigaghNVKWYqqlpg--tapkllifhnnar----------fSVSK--
qsVLTQPPsasgtpgqrvTISCSGtssnigs-sTVNWYqqlpg--mapklliyrdamrpsgvpdr---fSGSK--
evQLVQSGggvvqpgrslRLSCS-SsgfifssyAMYWVrqapgkglewvaiiwddgsdqhyadsvkgrfTISRNd
avQLEQSGpglvrpsqtlSLTCT-VsgtsfddyYWTWVrqppgrglewigyvfytg-ttlldpslrgrvTMLVNt

SgTSATLAItglqaedeadYYCQSY--------drslr--VFGggtkltvlr-
SgASASLAIgglqsedetdYYCAAW--------dvslnayVFGtgtkvtvlgq
skNTLFLQMdslrpedtgvYFCARDgghgfcssascfgpdYWGqgtpvtvss-
skNQFSLRLssvtaadtavYYCARN--------liaggidVWGqgslvtvss-
```

is the Clustal alignment of the variable regions, and

```
-----pSVFLFPpkpkdtlmisrtpEVTCVVvdvshedpqvKFNWYvdgvqvhn-aKTKPReqqynstyrVVSVL
qpkaapSVTLFPpsse--elqankaTLVCLIsdfypg--avTVAWKadgspvkaGVETTtpskqsnnkyaASSYL
astkgpSVFPLApt----------aALGCLVkdyfpe--pvTVSWN-sgaltsG-VHTFpavlqssglysLSSVV
qpre-pQVYTLPpsre--emtknqvSLTCLVkgfyps--diAVEWEsngqpenN-YKTTppvldsdgsffLYSKL

TVlhqnwldgkeYKCKVSnkalpapIEKtiskakg-
SLtpeqwkshksYSCQVTheg--stVEKtvaptscs
TVpssslgt-qtYICNVNhkpsntkVDKkvepksa-
TVdksrwqqgnvFSCSVMhea--lhnhyTQKslsl-
```

is the Clustal alignment of the constant regions. Calculating accuracy for this case, we observe only 4 misaligned residues in the variable regions, and only 9 misaligned residues in the constant regions.

[ImmQualityAndRelatedness] If multiple alignment gives us worse results, why bother with it ? As the previous examples show, *distant* sequences can have a malign influence on the alignment of more *related* sequences, but we are hopeful that by adding *related* sequences, we can improve the alignment of *distant* sequences.

Indeed, the Clustal alignment of BS3 and BS8 is as follows,

```
evQLVQSGggvvqpgr------slRLSCSSsgfifssyAMYWVr-qapgkglewvaiiwd-dgsdqhyadsvkgr
--qprepQVYTLPpsreemtknqvSLTCLVkgfypsdiAVEWEsngqpenNYKTTppvldsdgs----------f

fTISRNdskNTLFLQMdslrpedtgvYFCARDgghgfcssascfgpdYWGqgtpvtvss
fLYSK--------LTVdksr---------wqqgnvFSCSVMhealhnhyTQKslsl---
```

It contains 24 misaligned residues (out of 38), and it's obvious that adding related sequences here improves the alignment significantly. Using their own alignment method, Barton & Sternberg perform all pairwise alignments, one variable aligned to one constant, and note that adding the remaining 6 sequences and aligning multiply improves accuracy from 41 to 63 percent, on average.

Exercise 58 [15, opt.] Use Geoffrey Barton's AMAS utility, http://geoff.biop.ox.ac.uk/servers/amas_server.html, to analyse the multiple alignments from this section. Start with the "optimal" MSA alignment we pieced together. AMAS will give you an idea of the physical properties that are conserved at various positions. Can you find residues with hydrophobic properties at $i, i+2, i+4$ separated by unconserved or hydrophilic residues at $i+1, i+3$? Such a pattern is typical for a surface $\beta-$strand. AMAS currently accepts FASTA format, provided that you add the character "*" to the end of each sequence, like this:

```
>BS1, 7FAB light chain variable region
-----ASVLTQPPSVSGAPGQRVTISCTGSSSNIGAG-HNVKWYQQLPGTAPK--LLIFHNN----------ARF
SVSKSGTSAT--LAITGLQAEDEADYYC--QSYDR--------SLR--VFGGGTKLTVLR-*
>BS2, 2FB4 light chain variable region
-----QSVLTQPPSASGTPGQRVTISCSGTSSNIGS--STVNWYQQLPGMAPK--LLIYRDA---MRPSGVPDRF
SGSKSGASAS--LAIGGLQSEDETDYYC--AAWDV--------SLNAYVFGTGTKVTVLGQ*
[...]
```

3.5 Some Bibliographic Hints.

Review papers with an emphasis on heuristic multiple alignment are [CWC92] and [MVF94], the latter comparing the results of various implementations on 4 standard datasets. ClustalW is described in [THG94]. For MSA references, see the theory part of this chapter. A general survey on the sequence analysis of immunoglobulins is given in [Wil87]. Some papers dealing with the alignment of immunological sequences are [Tay86], [BaS87] (of course!), and [ViA91].

3.6 Appendix 1. Another Heuristic Alignment of the Immunoglobulin Sequences.

Here's the heuristic alignment that is calculated by the MSA preprocessing; the author is currently looking for some exact documentation. [LAK89] write about the MSA 1.0 implementation, that they use "a progressive alignment strategy similar to those described by Waterman and Perlwitz [WaP84], Feng and Doolittle [FeD87] and Taylor [Tay87]". "Progressive alignment" obviously refers to the "Once a gap, always a gap" rule mentioned above. However, the MSA 2.0 paper [GKS95] offers a different description ?!

```
ASVLTQPPSVSGAPG--------QRVTISCTGSSSNIGAGHNV--KWYQQLPGTAPK---LLIFHNN--------
QSVLTQPPSASGTPG--------QRVTISCSGTSSNIGSS-TV--NWYQQLPGMAPK---LLIYRDAM--RPSGV
EVQLVQSGGGVVQPG--------RSLRLSCSSSGFIFSSY-AM--YWVRQAPGKGLEWVAIIWDDGSDQHYADSV
AVQLEQSGPGLVRPS--------QTLSLTCTVSGTSFDDY-YW--TWVRQPPGRGLEWIGYVFYTGTT-LLDPSL
------PSVFLFPPKPKDTLMISRTPEVTCVVVDVSHEDP-QVKFNWYVDGVQVHNA----KTKPREQ-------
-QPKAAPSVTLFPPSSEE--LQANKATLVCLISDFYPGAV-TV--AWKADGSPVKAG---VETTTPSK-------
-ASTKGPSVFPLAPT----------AALGCLVKDYFPEPV-TV--SW--NSGALTSG---VHTFPAVL-------
--QPREPQVYTLPPSREE--MTKNQVSLTCLVKGFYPSDI-AV--EW-ESNGQPENN---YKTTPPVL-------

-ARFS--VSKSGTSATLAITGLQAEDEADYYCQSYDRSL--------R--VFGGGTKLTVLR--
PDRFS--GSKSGASASLAIGGLQSEDETDYYCAAWDVSL--------NAYVFGTGTKVTVLGQ-
KGRFTISRNDSKNTLFLQMDSLRPEDTGVYFCARDGGHGFCSSASCFGPDYWGQGTPVTVSS--
RGRVTMLVNTSKNQFSLRLSSVTAADTAVYYCARNLIAG--------GIDVWGQGSLVTVSS--
-QYNS--TYRVVSVLTVLHQNWLDGK--EYKCKVSNKAL--------P---APIEKTISKAKG-
-QSNN--KYAASSYLSLTPEQWKSHK--SYSCQVTHEG-------------STVEKTVAPTSCS
-QSSG--LYSLSSVVTVPSSSLGTQ---TYICNVNHKPS--------N---TKVDKKVEPKSA-
-DSDG--SFFLYSKLTVDKSRWQQGN--VFSCSVMHEAL--------H---NHYTQKSLSL---
```

3.7 Appendix 2. "Optimal" MSA-Alignment in Fasta-Format.

For your convenience in solving some of the exercises, the following is the "Optimal" MSA-Alignment, in FASTA-Format.

```
>BS1, 7FAB light chain variable region
-----ASVLTQPPSVSGAPGQRVTISCTGSSSNIGAG-HNVKWYQQLPGTAPK--LLIFHNN----------ARF
SVSKSGTSAT--LAITGLQAEDEADYYC--QSYDR--------SLR--VFGGGTKLTVLR-
>BS2, 2FB4 light chain variable region
-----QSVLTQPPSASGTPGQRVTISCSGTSSNIGS--STVNWYQQLPGMAPK--LLIYRDA---MRPSGVPDRF
SGSKSGASAS--LAIGGLQSEDETDYYC--AAWDV--------SLNAYVFGTGTKVTVLGQ
>BS3, 2FB4 heavy chain variable region
-----EVQLVQSGGGVVQPGRSLRLSCSSSGFIFSS--YAMYWVRQAPGKGLEWVAIIWDDGSDQHYADSVKGRF
```

```
TISRNDSKNTLFLQMDSLRPEDTGVYFCARDGGHGFCSSASCFGPD--YWGQGTPVTVSS-
>BS4, 7FAB heavy chain variable region
-----AVQLEQSGPGLVRPSQTLSLTCTVSGTSFDD--YYWTWVRQPPGRGLEWIGYVFYTG-TTLLDPSLRGRV
TMLVNTSKNQFSLRLSSVTAADTAVYYCARNLIAG--------GID--VWGQGSLVTVSS-
>BS5, 1FC1 heavy chain constant region
-P--SVFLFPPKPKDTLMISRTPEVTCVVVDVSHEDPQVKFNWYVD--GVQVH--NAKTKPR----------EQQ
YNSTYRVVSV--LTVLHQNWLDGKEYKC--KVSNK--------ALP--APIEKTISKAKG-
>BS6, 7FAB light chain constant region
QPKAAPSVTLFPPSSEELQANKATLVCLISDFYPGA--VTVAWKADG-SPVKA--GVETTTP----------SKQ
SNNKYAASSY--LSLTPEQWKSHKSYSC--QVTHE--------GST----VEKTVAPTSCS
>BS7, 7FAB heavy chain constant region
--------ASTKGPSVFPLAPTAALGCLVKDYFPEP--VTVSWNSG--GALTS--GVHTFPA----------VLQ
SSGLYSLSSV--VTV-PSSSLGTQTYIC--NVNHK--------PSN--TKVDKKVEPKSA-
>BS8, 1FC1 heavy chain constant region
QPR-EPQVYTLPPSREEMTKNQVSLTCLVKGFYPSD--IAVEWESN--GQPEN--NYKTTPP----------VLD
SDGSFFLYSK--LTVDKSRWQQGNVFSC--SVMHE--------ALH--NHYTQKSLSL---
```

3.8 Acknowledgement.

This work was supported by the Association for the Promotion of Science and Humanities in Germany (Stifterverband für die Deutsche Wissenschaft). Peter Serocka from the Visualization Laboratory of the Research Center for Studies on Structure Formation has been very helpful with the preparation of several figures. Mandy Caird of the University of Colorado, and Chris Kiesewetter have offered invaluable technical assistance. I'd like to thank Hershel Safer, Rolf Engstrand, Gerard Pujadas, Robert Giegerich, Geoff Barton, Rebecca Parsons, Andrea Schafferhans, Peter Hjelmstrom, Jotun Hein, Jürgen Frey, Wolfram Altenhofen, Fredj Tekaia, Christian Büschking [more to follow] for valuable comments on the manuscript.

References

[Alt89] S. F. Altschul, Gap Costs for Multiple Sequence Alignment. J Theoretical Biol 1989;138:297-309

[AlL89] S. F. Altschul, D. J. Lipman, Trees, Stars, and Multiple Biological Sequence Alignment. SIAM J Appl Math 1989;49(1):197-209

[BCHM94] P. Baldi, Y. Chauvin, T. Hunkapiller, M. A. McClure, Hidden Markov models of biological primary sequence information. Proc Nat Acad Sci USA 1994;91:1059-1063

[BaS87] G. J. Barton, M. E. J. Sternberg, A Strategy for the Rapid Multiple Alignment of Protein Sequences. Confidence Levels from Tertiary Structure Comparisons. J Mol Biol 1987;198:327-337

[Bar95] G. J. Barton, Protein Sequence Alignment and Database Scanning. in: Protein Structure prediction, a practical approach, Edited by M.J.E. Sternberg, to appear.

[CaL88] H. Carrillo, D. Lipman, The Multiple Sequence Alignment Problem in Biology. SIAM J Appl Math 1988;48(5):1073-1082

[CWC92] S. C. Chan, A. K. C. Wong, A Survey of Multiple Sequence Comparison Methods. Bull Math Biol 1992;54(4):563-598

[Edd95] S. Eddy, Multiple Alignment using Hidden Markov Models. Proc. Intelligent Systems for Molecular Biology (Edited by: C. Rawlings et al.) AAAI Press 1995;0:114-120

[FeD87] D. F. Feng, R. F. Doolittle, Progressive Sequence Alignment as a Prerequisite to Correct Phylogenetic Trees. J Mol Evol 1987;25:351-360

[GKS95] S. K. Gupta, J. Kececioglu, A. A. Schäffer, Improving the Practical Space and Time Efficiency of the Shortest-Paths Approach to Sum-of-Pairs Multiple Sequence Alignment. J Computational Biology 1995;2(3):459-472

[Gus93] D. Gusfield, Efficient Methods for Multiple Sequence Alignment with Guaranteed Error Bounds. Bull Math Biol 1993;55(1):141-154

[Kec95] J. Kececioglu *et al*, The MSA algorithm. (Transcript of a Discussion via Electronic Conferencing.) VSNS BioComputing Division WWW Server 1995; http://www.techfak.uni-bielefeld.de/bcd/Lectures/kececioglu.html

[Kec93] J. Kececioglu, The Maximum Weight Trace Problem in Multiple Sequence Alignment. Lecture Notes in Comput Sci 1993;684:106-119

[KBM+94] A. Krogh, M. Brown, I. S. Mian, K. Sjölander, D. Haussler, Hidden Markov Models in Computational Biology: Applications to Protein Modeling, J Mol Biol 1994;235:1501-1531

[LAB+93] C. E. Lawrence, S. F. Altschul, M. S. Boguski, J. S. Liu, A. F. Neuwald, J. C. Wootton, Detecting Subtle Sequence Signals: A Gibbs Sampling Strategy for Multiple Alignment. Science 1993;262:208-214

[LAK89] D. J. Lipman, S. F. Altschul, J. Kececioglu, A Tool for Multiple Sequence Alignment. Proc Nat Acad Sci USA 1989;86:4412-4415

[MVF94] M. A. McClure, T. K. Vasi, W. M. Fitch, Comparative Analysis of Multiple Protein-Sequence Alignment Methods. Mol Biol Evol 1994;11(4):571-592

[Mye91] E. W. Myers, An Overview of Sequence Comparison Algorithms in Molecular Biology. Technical Report 91-29, Department of Computer Science, The University of Arizona, Tucson, AZ 1991;0:1-23

[Tay86] W. R. Taylor, Identification of Protein Sequence Homology by Consensus Template Alignment. J Mol Biol 1986;188:233-258

[Tay87] W. R. Taylor, Multiple Sequence Alignment by a Pairwise Algorithm. Comput Appl Biosci 1987;3(2):81-87

[THG94] J. Thompson, D. G. Higgins, T. J. Gibson, Clustal W: improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice. Nucleic Acids Res 1994;22:4673-4680

[ViA91] M. Vingron, P. Argos, Motif Recognition and Alignment for Many Sequences by Comparison of Dot-matrices. J Mol Biol 1991;218:33-43

[Vin91] M. Vingron, Multiple Sequence Alignment and Applications in Molecular Biology. Preprint 91-21, IWR Interdisziplinaeres Zentrum fuer Wissenschaftliches Rechnen der Universitaet Heidelberg, Heidelberg, Germany, 1991;0:1-69

[Wat89] M. S. Waterman, Sequence Alignments. *in* Mathematical Methods for DNA Sequences, M. S. Waterman (ed.) CRC Press, Boca Raton, FL, 1989;0:53-92

[Wat95] M. S. Waterman, Introduction to Computational Biology, Chapter 10: Multiple Sequence Alignment. Chapman and Hall, London, 1995;0:233-252

[WaP84] M. S. Waterman, M. D. Perlwitz, Line Geometries for Sequence Comparisons. Bull Math Biol 1984;46(4):567-577

[Wil87] A. Williams, A year in the life of the immunoglobulin superfamily. Immunology Today 1987;8:298-303