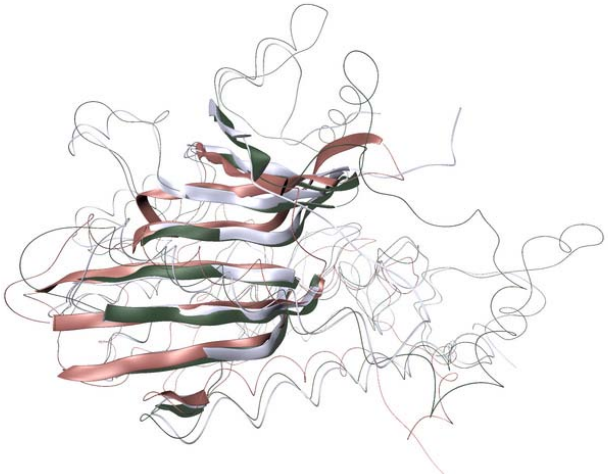
University of Illinois at Urbana-Champaign
Luthey-Schulten Group
Theoretical and Computational Biophysics Group
Summer School 2004 - University of Western Australia, Perth

Sequence Alignment Algorithms



Rommie Amaro

Felix Autenrieth

Brijeet Dhaliwal

Barry Israelewitz

Zaida Luthey-Schulten

Anurag Sethi

Taras Pogorelov

June 2004

A web version, in color, is available at
http://www.ks.uiuc.edu/Training/SumSchool03/Tutorials/bioinformatics

Contents

1 Introduction 3

2 Sequence Alignment Algorithms 5

2.1 Manually perform a Needleman-Wunsch alignment 5

2.2 Finding homologous pairs of ClassII tRNA synthetases 10

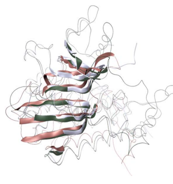
1 Introduction

The recent developments of projects such as the sequencing of the genome from several organisms, and high-throughput X-ray structure analysis, have brought a large amount of data about the sequences and structures of several thousand proteins to the scientific community. This information can be used effectively for medical and biological research only if one can extract functional insight from the sequence and structural data. To achieve this we need to understand how the proteins perform their functions. Two main computational techniques exist to reach this a goal: a *bioinformatics* approach, and atomistic *molecular dynamics* simulations. Bioinformatics uses the statistical analysis of protein sequences and structures to understand their function and predict structures when only sequence information is available. Molecular modeling and molecular dynamics simulations use principles from physics and physical chemistry to study the function and folding of proteins.

Bioinformatics methods are among the most powerful technologies available in life sciences today. They are used in fundamental research on theories of evolution and in more practical considerations of protein design. Algorithms and approaches used in these studies range from sequence and structure alignments, secondary structure prediction, functional classification of proteins, threading and modeling of distantly-related homologous proteins to modeling the progress of protein expression through a cell's life cycle.

In this tutorial you will use a classic global sequence alignment method, the Needleman-Wunsch algorithm, to align two small proteins. First you will align them by hand and perform your own dynamic programming; afterwards you will check your work against a computer program that we provide you. The Needleman-Wunsch alignment programs have been kindly provided by Anurag Sethi.

The entire tutorial takes about an hour to complete in its entirity.



Protein sequences vs. nucleotide sequences. A protein is a sequence of amino acids linked with peptide bonds to form a polypeptide chain. In this tutorial, the word *sequence* (unless otherwise specified) refers to the amino acid residue sequence of a protein; by convention these sequences are listed from the N-terminal to the C-terminal of the chain. Sequences can be written with full names, as in "Lysine, Arginine, Cysteine, ...", with 3-letter codes, "Lys, Arg, Cys, ...", or with 1-letter codes, "K, R, C, ..." . Proteins range in size from a few dozen to several thousand residues. The nucleotide sequences of DNA encodes protein sequence. Sections of genes in chromosomal DNA are copied to mRNA, which provides the guide for ribosome to assemble a protein. A nucleotide sequence may be written as "Cytosine, Adenine, Adenine, Guanine, ...", or "C, A, A, G, ...".

This tutorial assumes that the alignment programs we provide you have been correctly installed on the user's computer. Please ask a lab attendant for help if you have any trouble locating software or data files during the tutorial.

To set up the exercises...

You will make a copy of the files needed for these exercises in your home directory. Open a terminal window, and, if you don't already have one, make a `~/tbss.work` directory by typing at the Unix prompt:

```
>> mkdir ~/tbss.work
```

Make sure that you have a `~/tbss.work` directory:

```
>> ls ~/tbss.work
```

Copy the needed directory, but instead of typing *TOP_DIR*, type the location of the Summer School directory tree:

```
>> cp -rp TOP_DIR/sumschool03/tutorials/07-bioinformatics/files/Bioinformatics/ ~/tbss.work/
```

For instance, if the materials are located at `/mnt/cdrom`, you will type:

```
cp -rp /mnt/cdrom/sumschool03/tutorials/07-bioinformatics/files/Bioinformatics/ ~/tbss.work/
```

Check that you have the files in this directory by listing the contents:

```
>> cd ~/tbss.work/Bioinformatics
>> ls -lR
```

In this tutorial, when we refer to `~/tbss.work/Bioinformatics/` and its subdirectories, we are referring to the copy which you have just made in your own home directory.

2 Sequence Alignment Algorithms

In this section you will optimally align two short protein sequences using pen and paper, then search for homologous proteins by using a computer program to align several, much longer, sequences.

Dynamic programming algorithms are recursive algorithms modified to store intermediate results, which improves efficiency for certain problems. The Needleman–Wunsch algorithm uses a dynamic programming algorithm to find the optimal global alignment of two sequences — a and b. The alignment algorithm is based on finding the elements of a matrix H where the element $H_{i,j}$ is the optimal score for aligning the sequence (a_1,a_2,...,a_i) with (b_1,b_2,.....,b_j). Two similar amino acids (e.g. arginine and lysine) receive a high score, two dissimilar amino acids (e.g. arginine and glycine) receive a low score. The higher the score of a path through the matrix, the better the alignment. The matrix H is found by progressively finding the matrix elements, starting at $H_{1,1}$ and proceeding in the directions of increasing i and j. Each element is set according to:

$$H_{i,j} = \max \begin{cases} H_{i-1,j-1} + S_{i,j} \\ H_{i-1,j} - d \\ H_{i,j-1} - d \end{cases}$$

where $S_{i,j}$ is the similarity score of comparing amino acid a_i to amino acid b_j (obtained here from the BLOSUM40 similarity table) and d is the penalty for a single gap. The matrix is initialized with $H_{0,0} = 0$. When obtaining the local Smith-Waterman alignment, $H_{i,j}$ is modified:

$$H_{i,j} = \max \begin{cases} 0 \\ H_{i-1,j-1} + S_{i,j} \\ H_{i-1,j} - d \\ H_{i,j-1} - d \end{cases}$$

The gap penalty can be modified, for instance, d can be replaced by $(d \times k)$, where d is the penalty for a single gap and k is the number of consecutive gaps.

Once the optimal alignment score is found, the "traceback" through H along the optimal path is found, which corresponds to the the optimal sequence alignment for the score. In the next set of exercises you will manually implement the Needleman-Wunsch alignment for a pair of short sequences, then perform global sequence alignments with a computer program developed by Anurag Sethi, which is based on the Needleman-Wunsch algorithm with an affine gap penalty, $d + e(k - 1)$, where e is the extension gap penalty. The output file will be in the GCG format, one of the two standard formats in bioinformatics for storing sequence information (the other standard format is FASTA).

2.1 Manually perform a Needleman-Wunsch alignment

In the first exercise you will test the Needleman-Wunsch algorithm on a short sequence parts of hemoglobin (PDB code `1AOW`) and myoglobin 1 (PDB code `1AZI`).

Here you will align the sequence `HGSAQVKGHG` to the sequence `KTEAEMKASEDLKKHGT`.

The two sequences are arranged in a matrix in Table 1. The sequences start at the upper right corner, the initial gap penalties are listed at each offset starting position. With each move from the start position, the initial penalty increase by our single gap penalty of 8.

		H	G	S	A	Q	V	K	G	H	G
	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
K	-8										
T	-16										
E	-24										
A	-32										
E	-40										
M	-48										
K	-56										
A	-64										
S	-72										
E	-80										
D	-88										
L	-96										
K	-104										
K	-112										
H	-120										
G	-128										
T	-136										

Table 1: The empty matrix with initial gap penalties.

1 The first step is to fill in the similarity scores $S_{i,j}$ from looking up the matches in the BLOSUM40 table, shown here labeled with 1-letter amino acid codes:

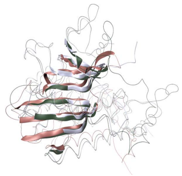
```
A  5
R -2  9
N -1  0  8
D -1 -1  2  9
C -2 -3 -2 -2 16
Q  0  2  1 -1 -4  8
E -1 -1 -1  2 -2  2  7
G  1 -3  0 -2 -3 -2 -3  8
H -2  0  1  0 -4  0  0 -2 13
I -1 -3 -2 -4 -4 -3 -4 -4 -3  6
L -2 -2 -3 -3 -2 -2 -2 -4 -2  2  6
K -1  3  0  0 -3  1  1 -2 -1 -3 -2  6
M -1 -1 -2 -3 -3 -1 -2 -2  1  1  3 -1  7
F -3 -2 -3 -4 -2 -4 -3 -3 -2  1  2 -3  0  9
P -2 -3 -2 -2 -5 -2  0 -1 -2 -2 -4 -1 -2 -4 11
S  1 -1  1  0 -1  1  0  0 -1 -2 -3  0 -2 -2 -1  5
T  0 -2  0 -1 -1 -1 -1 -2 -2 -1 -1  0 -1 -1  0  2  6
W -3 -2 -4 -5 -6 -1 -2 -2 -5 -3 -1 -2 -2  1 -4 -5 -4 19
Y -2 -1 -2 -3 -4 -1 -2 -3  2  0  0 -1  1  4 -3 -2 -1  3  9
V  0 -2 -3 -3 -2 -3 -3 -4 -4  4  2 -2  1  0 -3 -1  1 -3 -1  5
B -1 -1  4  6 -2  0  1 -1  0 -3 -3  0 -3 -3 -2  0  0 -4 -3 -3  5
Z -1  0  0  1 -3  4  5 -2  0 -4 -2  1 -2 -4 -1  0 -1 -2 -2 -3  2  5
X  0 -1 -1 -1 -2 -1 -1 -1 -1 -1 -1 -1  0 -1 -2  0  0 -2 -1 -1 -1 -1 -1
   A  R  N  D  C  Q  E  G  H  I  L  K  M  F  P  S  T  W  Y  V  B  Z  X
```

2 We fill in the BLOSUM40 similarity scores for you in Table 2.

3 To turn this S matrix intro the dynamic programming H matrix requires calculation of the contents of all 170 boxes. We've calculated the first 4 here, and encourage you to calculate the contents of at least 4 more. The practice will come in handy in the next steps. As described above, a matrix square cannot be filled with its dynamic programming value until the squares above, to the left, and to the above-left diagonal are computed. The value of a square is,

$$H_{i,j} = \max \begin{cases} H_{i-1,j-1} + S_{i,j} \\ H_{i-1,j} - d \\ H_{i,j-1} - d \end{cases}$$

using the convention that H values appear in the top part of a square in large print, and S values appear in the bottom part of a square in small print. Our gap penalty d is 8.

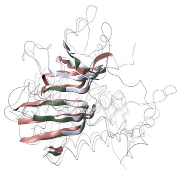


Example:. In the upper left square in Table 2, square (1,1), the similarity score $S_{1,1}$ is -1, the number in small type at the bottom of the box. The value to assign as $H_{1,1}$ will be the greatest ("max") of these three values: $(H_{0,0} + S_{1,1}), (H_{0,1} - d), (H_{1,0} - d))$. That is, the greatest of: $(0 + -1), (-8 - 8), (-8 - 8)$ which just means the greatest of: -1, -16, and -16. This is -1, so we write -1 as the value of $H_{1,1}$ (the larger number in the top part of the box). The same reasoning in square (2,1) leads us to set $H_{2,1}$ as -9, and so on. *Note: we consider $H_{0,0}$ to be the "predecessor" of $H_{1,1}$, since it helped decided $H_{1,1}$'s value. Later, predecessors will qualify to be on the traceback path.*

4 Again, just fill in 4 or 5 boxes in Table 2 until you get a feel for gap penalties and similarity scores S vs. alignment scores H. In the next step, we provide the matrix with all values filled in as Table 2.1. Check that your 4 or 5 calculations match.

5 Now we move to Table 2.1, with all 170 $H_{i,j}$ values are shown, to do the "alignment traceback". To find the alignment requires one to trace the path through from the end of the sequence (the lower right box) to the start of the sequence (the upper left box). This job looks complicated, but should only take about 5 –7 minutes.

6 We are tracing a path in Table 2.1, from the lower right box to the upper left box. You can only move to a square if it could have been a "predecessor" of your current square – that is, when the matrix was being filled with $H_{i,j}$ values, the move from the predecessor square to your current square would have followed the mathematical rules we used to find H$_{i,j}$ above. Circle each square you move to along your path.



Example:. we start at the lower right square (10,17), where $H_{10,17}$ is -21 and $S_{10,17}$ is -2. We need to test for 3 possible directions of movement: diagonal (up + left), up, and left. The condition for diagonal movement given above is: $H_{i,j} = H_{i-1,j-1} + S_{i,j}$, so for the diagonal box (9,16) to have contributed to (10,17), $H_{9,16} + S_{10,17}$ would have to equal the H value of our box, -21. Since (-29 + -2) does not equal -21, the diagonal box is not a "predecessor", so we can't move in that direction. We try the rule for the box to the left: $H_{i,j} = H_{i-1,j} - d$ Since -37 - 8 does not equal -21, we also can't move left. Our last chance is moving up. We test $H_{i,j} = H_{i,j-1} - d$. Since -21 = (-13 - 8) we can move up! Draw an arrow from the lower right box, ($H_{10,17} = -21, S_{10,17} = -2$) to the box just above it, ($H_{10,16} = -13, S_{10,16} = 8$).

7 Continue moving squares, drawing arrows, and circling each new square you land on, until you have reached the upper right corner of the matrix If the path branches, follow both branches.

8 Write down the alignment(s) that corresponds to your path(s) by writing the the letter codes on the margins of each position along your circled path. Aligned pairs are at the boxes at which the path exits via the upper-left corner. When there are horizontal or vertical movements movements along your path, there will be a gap (write as a dash, "-") in your sequence.

9 Now to check your results against a computer program. We have prepared a pairwise Needleman-Wunsch alignment program, `pair`, which you will apply to the same sequences which you have just manually aligned.

10 Change your directory by typing at the Unix prompt:

```
cd ~/tbss.work/Bioinformatics/pairData
```

then start the pair alignment executable by typing:

```
pair targlist
```

. All alignments will be carried out using the BLOSUM40 matrix, with a gap penalty of 8. The paths to the input files and the BLOSUM40 matrix used are defined in the file `targlist`; the BLOSUM40 matrix is the first 25 lines of the file `blosum40`. (Other substitution matrices can be found at the NCBI/Blast website.)

Note: In some installations, the `pair` *executable is in* `~/tbss.work/Bioinformatics/pairData` *and here you must type* `./pair targlist` *to run it.*
If you cannot access the `pair` *executable at all, you can see the output from this step in* `~/tbss.work/Bioinformatics/pairData/example_output/`

11 After executing the program you will generate three output files namely `align`, `scorematrix` and `stats`. View the alignment in GCG format by typing `less align`. The file `scorematrix` is the 17x10 *H* matrix. If there are multiple paths along the traceback matrix, the program `pair` will choose only one path, by following this precedence rule for existing potential traceback directions, listed in decreasing precedence: diagonal (left and up), up, left. In the file `stats` you will find the optimal alignment score and the percent identity of the alignment.

Questions. Compare your manual alignment to the the output of the pair program. Do the alignments match?

2.2 Finding homologous pairs of ClassII tRNA synthetases

Homologous proteins are proteins derived from a common ancestral gene. In this exercise with the Needleman-Wunsch algorithm you will study the sequence identity of several class II tRNA synthetases, which are either from Eucarya, Eubacteria or Archaea or differ in the kind of aminoacylation reaction which they catalyze. Table 4 summarizes the reaction type, the organism and the PDB accession code and chain name of the employed Class II tRNA synthetase domains.

We have have prepared a computer program `multiple` which will align multiple pairs of proteins.

1 Change your directory by typing at the Unix prompt:

```
cd ~/tbss.work/Bioinformatics/multipleData
```

then start the alignment executable by typing:

```
multiple targlist
```

Note: In some installations, the `multiple` *executable is in* `~/tbss.work/Bioinformatics/multipleData` *and here you must type* `./multiple targlist` *to run it.*
If you cannot access the `multiple` *executable at all, you can see the output from this step in* `~/tbss.work/Bioinformatics/multipleData/example_output/`

2 In the `align` and `stats` files you will find all combinatorial possible pairs of the provided sequences. On a piece of paper, write the names of the the proteins, grouped by ther domain of life, as listed in Table 4. Compare sequence identities of aligned proteins from the same domain of a life, and of aligned proteins from different domains of life, to help answer the questions below.

Questions. What criteria do you use in order to determine if two proteins are homologous? Can you find a pattern when you evaluate percent identities between the pairs of class II tRNA synthetases? Which is the most evolutionarily related pair, and which is the most evolutionarily divergent pair according to the sequence identity?

		H	**G**	**S**	**A**	**Q**	**V**	**K**	**G**	**H**	**G**
	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
K	-8	**−1** −1	**−9** −2	0	−1	1	−2	6	−2	−1	−2
T	-16	**−9** −2	**−3** −2	2	0	−1	1	0	−2	−2	−2
E	-24	0	−3	0	−1	2	−3	1	−3	0	−3
A	-32	−2	1	1	5	0	0	−1	1	−2	1
E	-40	0	−3	0	−1	2	−3	1	−3	0	−3
M	-48	1	−2	−2	−1	−1	1	−1	−2	1	−2
K	-56	−1	−2	0	−1	1	−2	6	−2	−1	−2
A	-64	−2	1	1	5	0	0	−1	1	−2	1
S	-72	−1	0	5	1	1	−1	0	0	−1	0
E	-80	0	−3	0	−1	2	−3	1	−3	0	−3
D	-88	0	−2	0	−1	−1	−3	0	−2	0	−2
L	-96	−2	−4	−3	−2	−2	2	−2	−4	−2	−4
K	-104	−1	−2	0	−1	1	−2	6	−2	−1	−2
K	-112	−1	−2	0	−1	1	−2	6	−2	−1	−2
H	-120	13	−2	−1	−2	0	−4	−1	−2	13	−2
G	-128	−2	8	0	1	−2	−4	−2	8	−2	8
T	-136	−2	−2	2	0	−1	1	0	−2	−2	−2

Table 2: Alignment score worksheet. In all alignment boxes, the similarity score $S_{i,j}$ from the BLOSUM40 matrix lookup is supplied (small text, bottom of square). Four alignment scores are provided as examples (large text, top of square), try and calculate at least four more, following the direction provided in the text for calculating $H_{i,j}$.

		H	**G**	**S**	**A**	**Q**	**V**	**K**	**G**	**H**	**G**
	0	-8	-16	-24	-32	-40	-48	-56	-64	-72	-80
K	-8	**−1** −1	**−9** −2	**−16** 0	**−24** −1	**−31** 1	**−39** −2	**−42** 6	**−50** −2	**−58** −1	**−66** −2
T	-16	**−9** −2	**−3** −2	**−7** 2	**−15** 0	**−23** −1	**−30** 1	**−38** 0	**−44** −2	**−52** −2	**−60** −2
E	-24	**−16** 0	**−11** −3	**−3** 0	**−8** −1	**−13** 2	**−21** −3	**−29** 1	**−37** −3	**−44** 0	**−52** −3
A	-32	**−24** −2	**−15** 1	**−10** 1	**2** 5	**−6** 0	**−13** 0	**−21** −1	**−28** 1	**−36** −2	**−43** 1
E	-40	**−32** 0	**−23** −3	**−15** 0	**−6** −1	**4** 2	**−4** −3	**−12** 1	**−20** −3	**−28** 0	**−36** −3
M	-48	**−39** 1	**−31** −2	**−23** −2	**−14** −1	**−4** −1	**5** 1	**−3** −1	**−11** −2	**−19** 1	**−27** −2
K	-56	**−47** −1	**−39** −2	**−31** 0	**−22** −1	**−12** 1	**−3** −2	**11** 6	**3** −2	**−5** −1	**−13** −2
A	-64	**−55** −2	**−46** 1	**−38** 1	**−26** 5	**−20** 0	**−11** 0	**3** −1	**12** 1	**4** −2	**−4** 1
S	-72	**−63** −1	**−54** 0	**−41** 5	**−34** 1	**−25** 1	**−19** −1	**−5** 0	**4** 0	**11** −1	**4** 0
E	-80	**−71** 0	**−62** −3	**−49** 0	**−42** −1	**−32** 2	**−27** −3	**−13** 1	**−4** −3	**4** 0	**8** −3
D	-88	**−79** 0	**−70** −2	**−57** 0	**−50** −1	**−40** −1	**−35** −3	**−21** 0	**−12** −2	**−4** 0	**2** −2
L	-96	**−87** −2	**−78** −4	**−65** −3	**−58** −2	**−48** −2	**−38** 2	**−29** −2	**−20** −4	**−12** −2	**−6** −4
K	-104	**−95** −1	**−86** −2	**−73** 0	**−66** −1	**−56** 1	**−46** −2	**−32** 6	**−28** −2	**−20** −1	**−14** −2
K	-112	**−103** −1	**−94** −2	**−81** 0	**−74** −1	**−64** 1	**−54** −2	**−40** 6	**−34** −2	**−28** −1	**−22** −2
H	-120	**−99** 13	**−102** −2	**−89** −1	**−82** −2	**−72** 0	**−62** −4	**−48** −1	**−42** −2	**−21** 13	**−29** −2
G	-128	**−107** −2	**−91** 8	**−97** 0	**−88** 1	**−80** −2	**−70** −4	**−56** −2	**−40** 8	**−29** −2	**−13** 8
T	-136	**−115** −2	**−99** −2	**−89** 2	**−96** 0	**−88** −1	**−78** 1	**−64** 0	**−48** −2	**−37** −2	**−21** −2

Table 3: Traceback worksheet. The completed alignment score matrix H (large text, top of each square) with the BLOSUM40 lookup scores $\mathrm{S}_{i,j}$ (small text, bottom of each square). To find the alignment, trace back starting from the lower right (T vs G, score -21) and proceed diagonally (to the left and up), left, or up. Only proceed, however, if the square in that direction could have been a predecessor, according to the conditions described in the text.

Specificity	Organism	PDB code:chain	ASTRAL catalytic domain
Aspartyl	Eubacteria	`1EQR:B`	`d1eqrb3`
Aspartyl	Archaea	`1B8A:A`	`d1b8aa2`
Aspartyl	Eukarya	`1ASZ:A`	`d1asza2`
Glycl	Archaea	`1ATI:A`	`d1atia2`
Histidyl	Eubacteria	`1ADJ:C`	`d1adjc2`
Lysl	Eubacteria	`1BBW:A`	`d1bbwa2`
Aspartyl	Eubacteria	`1EFW:A`	`d1efwa3`

Table 4: Domain types, origins, and accession codes