UNIVERSITY OF OSLO

Department of Informatics

Developing an interactive web-based learning environment for bioinformatics

Master thesis

Daniel Løkken

Rustad

27th July 2005

Preface

This thesis is part of the Master Degree in Informatics at the Department of Informatics at the University of Oslo. The work with this thesis started in the spring of 2004 and was completed during the summer of 2005. I would like to thank my supervisors, Associate Professors Anja Bråthen Kristoffersen and Ole Christian Lingjærde at the Bioinformatics group of the Department of Informatics at the University of Oslo, for their patience, good advice, and encouragement during this period. I would also like to thank Jorun B. Holmberg, Associate Professor at the Department of Special Needs Education, Faculty of Education, University of Oslo, for her helpful explanations of some of the concepts and terms used in the field of pedagogics.

This thesis is intended for readers who are interested in the possibilities web-technology offers for creating interactive web-based learning environments for bioinformatics.

Abstract

This thesis explores some of the possibilities web-technology offers for creating an interactive learning environment for bioinformatics on the web. A set of implemented examples on how some of the basic algorithms in bioinformatics can be visualised with web-technology is presented and discussed. An implemented example on a web-based bioinformatics exercise and a set of hypothetical web-based exercises are also presented and discussed.

Table of Contents

Preface.....i

Abstract.....iii

Chapter 1

Why an interactive web-based learning environment? 1

1.1 Introduction..... 1
1.2 Visualisation of algorithms.....2
1.3 Biological databases and other tools.....3
1.4 Interactive exercises..... 3
1.6 Challenges..... 5
1.7 Document overview..... 5

Chapter 2

Background on biology and introduction to the Sourcer's Apprentice.....7

2.1 Introduction..... 7
2.2 DNA, genes, and gene expression.....8
2.2.1 DNA and genes..... 8
2.2.2 Gene expression..... 9
2.3 Mutations..... 12
2.4 Microarrays..... 13
2.5 The Sourcer's Apprentice.....17

2.5.2 Empirical results for the Sourcer's Apprentice..... 19

Chapter 3

Pairwise global sequence alignment.....21

3.1 Introduction..... 21

3.2 The Needleman-Wunsch alignment algorithm..... 24

Chapter 4

Clustering of microarray data..... 29

4.1 Introduction..... 29

4.2 Similarity between objects..... 30

4.2 Hierarchical clustering..... 31

4.3 K-means clustering..... 34

4.3.1 The batch variant..... 34

4.3.2 The online variant..... 36

4.4 Self organizing maps..... 37

Chapter 5

The BioTeach system.....39

5.1 Introduction..... 39

5.2 The portal..... 40

5.3 The Needleman-Wunsch visualisation..... 41

5.3.1 The simple visualisation..... 43

5.3.2 The advanced visualisation..... 44

5.3.3 Explanation of the visualisation..... 45

5.4 The Needleman-Wunsch exercise..... 54

5.5 The clustering visualisation..... 58

5.5.1 The hierarchical clustering visualisation..... 62

5.5.3 The k-means visualisation..... 71

5.5.5 The self organizing maps visualisation..... 75

Chapter 6
Hypothetical exercise examples....78
6.1 Introduction....78
6.3 A Blast example....79
6.4 Database and portal exercises....82
6.4.1 A hypothetical example....84

Chapter 7
Summary and Conclusions....89

References....93

Appendix A
Downloading, installing and
configuring the BioTeach-system....97

Appendix B
Technical Solution of the BioTeach-system....101
B.1 Applied technology....101
B.1.1 Java Applets....101
B.1.2 The JSP Model 2 architecture....102
B.1.3 The bio.jar library....105
B.1.4 Graham's Scan....107
B.2 Application organisation and program flow....108

B.2.1 The portal.....108

B.2.1.4 help.html.....110

B.2.1.5 Bottom.html.....110

B.2.1.6 Main.css.....110

B.2.2 The Alignment application.....111

B.2.3 The clustering application.....122

Chapter 1

Why an interactive web-based learning environment?

1.1 Introduction

The idea of this project is to explore the possibilities and challenges of creating an interactive web-based learning environment for bioinformatics based on discussions of theoretical and implemented examples of what such an environment could include.

The term interactive is used here to describe tools that allow experimentation with and manipulation of concepts and principles. In other words, an interactive web-based learning environment is an environment that allows the users (e.g. students of bioinformatics) to experiment with and manipulate the concepts and principles it discusses. The interactive web-based learning environment discussed in this document is meant to be a supplement to, and not a replacement of, the existing educational means in bioinformatics.

Web-technology offers a variety of possibilities for creating educational web-applications, and it would be impossible to explore all possibilities in a single

project. It was therefore necessary to focus on a few approaches: visualisation of algorithms and interactive exercises.

1.2 Visualisation of algorithms

Bioinformatics is concerned with the interpretation and analysis of biological data, and a wide array of algorithms have been developed for this purpose. These algorithms are often based on complex statistical and mathematical models. The traditional way of representing such algorithms in textbooks, is to present the reader with the mathematical or statistical formula accompanied by descriptive text and figures written in a highly mathematical or statistical language. These algorithms might be easier to understand if the traditional descriptions were supplemented with step-by-step visualisations or walk-throughs. Traditional textbooks are, however, limited to static presentation of information, and such visualisations and walk-throughs would therefore require an unreasonable amount of pages that probably would be uninspiring to read.

An interactive web-based learning environment would be a more suitable medium for visualisations and walk-throughs. Web-technology has reached a stage where it is possible to create dynamic presentations of information based on input from the user. It should therefore be possible to design web-based learning experiences (i.e. web applications) that not only provide presentations of how the algorithms work, but also allow the users to experiment with an algorithm's parameters to see how different parameters affect the results of the algorithms.

There are too many basic algorithms in bioinformatics to give a thorough discussion of how all of them could be visualised in an interactive web-based learning environment. Of the available algorithms, the Needleman-Wunsch alignment algorithm, and a selection of algorithms for clustering of microarray data were chosen as candidates for discussion and visualisation. These were chosen because sequence alignment and clustering of microarray data represent two central, yet entirely different, fields in bioinformatics.

The visualisation examples included in this project are implemented in a system called the BioTeach system. This system includes examples on how the Needleman-Wunsch alignment algorithm and three central clustering algorithms can be visualised in an interactive web-based learning

environment. The BioTeach system also includes an interactive Needleman-Wunsch exercise example. The system is discussed in detail in chapter 5.

1.3 Biological databases and other tools

Biological databases are another important tool in bioinformatics. Researchers use these databases to store their work and to share their work with other researchers. There are different databases for different types of biological information, and different databases may have different ways of presenting information. It is therefore necessary to know which information can be found in the different databases, and how to obtain that information. The development of portals or gateways that allow multiple databases to be searched simultaneously have simplified the information search, but it is still necessary to know how to perform a database search, which of the searched databases to access to find the information they need, and how to read the information.

A web-based learning environment could be used to discuss the principles of database searching, the portals, and the databases, but, as Wolfe (2001a, b) argues, the web is not the best medium for long and exhaustive discussions. Traditional textbooks would, for this reason, probably be better suited for a detailed discussion of search principles, the databases, and the portals.

However, both the databases and the portals are web-based, and a web-based learning environment can therefore provide direct access to the various databases and portals. A web-based learning environment could, for this reason, be a convenient medium for short and precise discussions that provide direct access to the various databases and portals through links.

1.4 Interactive exercises

Exercises are an important part in the learning process because they allow one's understanding of a curricular subject to be tested and compared to the goals of the curriculum (Anderson, 2001). Thus, exercises concerning databases and portals are a natural part of an interactive web-based learning environment for bioinformatics. The web provides direct access to the databases and portals, and a web-based learning environment should therefore

be a convenient medium for exercises which are concerned with databases and portals. Now, if these exercises were combined with the discussion mentioned above, it would not be necessary to consult one or more traditional textbooks to find the answers to the exercises.

Exercises in an interactive web-based learning environment offer opportunities to create information presentations based on user input. In terms of exercises, user input means answers, while information presentation refers to comments. It should, in other words, be possible to use web-technology to design web-based exercises that comment the answers that are given.

Traditional textbooks seldom provide any other comments to exercises than the correct answers. The correct answers are better than no comments, but providing the correct answers does not necessarily ensure that the subject an exercise is concerned about is fully understood. Providing the correct answers may not, for instance, be particularly helpful for students who are unable to arrive at the correct answer to an exercise, and who, at the same time, are unable to understand why the correct answer is correct. Such students are forced to consult a third person, e.g. a fellow student or a lecturer, in order to get help.

The fastest way for a student to get help is probably through fellow students, but as they are students, they may not have understood the subject well enough to provide a correct explanation, or they may disagree in their understanding of the subject, both of which may further confuse an already confused student. Lecturers, on the other hand, are often busy, and may therefore have little time to consult students, even through email. Web-based exercises that provide instant comments should therefore be a useful supplement for both students and lecturers.

The Sourcer's Apprentice system developed by Britt and Gabrys (2001) is an example on how interactive exercises could be designed, and, although this system is not concerned with bioinformatics, it represents one possible approach to interactive exercises in bioinformatics. The Sourcer's Apprentice system is discussed further in chapter 2.

1.5 Other interesting features of web technology

Other interesting features with web-technology and web-applications are that web-applications don't require any installation on the part of the user apart from a web-browser, which is usually preinstalled by the computer manufacturer along with the operating system; and that web-applications can

be accessed from any computer anywhere as long as the computer is connected to the web.

1.6 Challenges

There are, as with any other educational tool, challenges connected to use the web for educational purpose.

One of the elements that has to be taken into consideration when deciding whether or not to use the web for educational purposes is access to the web; that is, students obviously have to have access to the web to be able to use a web-based learning environment, and must therefore be supplied with equipment (e.g. computers) that provides them access to the web.

Access to equipment is not, however, the only consideration that has to be taken into account. Students obviously have to know how to use the equipment and the web to be able to take advantage of a web-based learning environment. Now, the web is a popular medium for distributing information to the students, and most students should have some experience with the use of both computers and the web. However, some students might feel uncomfortable with the use of computers and the web, and may thus dislike the notion of having to use the web as a learning environment. Such students might feel more comfortable with a web-based learning environment if they were presented with an environment that resembles a familiar environment. Most students should be familiar with traditional textbooks and how they are structured, and it might therefore be a good idea to design a web-based learning environment to resemble a traditional textbook, both in appearance and structure.

1.7 Document overview

The rest of this document is organised as follows:

- Chapter 2 discusses the biological background information that the algorithms implemented by the BioTeach system is based on. The chapter also includes a discussion of the Sourcer's Apprentice system. Chapter 3 contains an introduction to pairwise sequence alignment, a discussion of the Needleman-Wunsch algorithm.

- Chapter 4 contains a short introduction to clustering of microarray data, and a discussion of the three different clustering approaches implemented and visualised by the BioTeach system: hierarchical clustering, k-means clustering, and self organising maps.

- Chapter 5 presents the BioTeach system and discusses the design of the algorithm visualisations and the exercise.

- Chapter 6 discusses hypothetical examples on how different interactive exercises concerning the Blast algorithm, and biological databases and portals can be designed.

- Chapter 7 contains a summary and conclusions.

- The appendices are concerned with the implementation of the BioTeach system.

Chapter 2

Background on biology and introduction to the Sourcer's Apprentice

2.1 Introduction

One of the objectives of this project is to discuss and provide examples on how a selection of the algorithms used for interpretation and analysis of biological data could be visualised in an interactive web-based learning environment. Sections 2.2 through 2.4 are meant as an introduction to the biological principles the selected algorithms are based on, and the data these algorithms are designed to analyse.

Another objective of this project is to discuss how interactive exercises could be designed in an interactive web-based learning environment. The Sourcer's Apprentice system represents one possible approach to exercise design, and is discussed in section 2.5.

Sections 2.2 to 2.4 is largely based on Draghici (2003), Causton *et al.* (2003), and Berrar *et al.* (2003), while section 2.5 is based on Britt and Gabrys (2001).

2.2 DNA, genes, and gene expression

Most of the algorithms used in bioinformatics, the Needleman-Wunsch and microarray data clustering algorithms included, are in some way concerned with genes. Genes are segments of DNA that contain the instructions which are needed to build and maintain organisms, either as mechanisms that regulate cellular processes, or as recipes that allow a cell to produce a protein.

The remainder of this section provides a short description of DNA, and the process of transforming a gene into a protein, or the gene expression process. Readers who are familiar with these concepts can safely skip the rest of this section.

2.2.1 DNA and genes

DNA contains the hereditary material (i.e. the genes) of an organism, and is found in most of the cells that compose the organism. The DNA is organised in large molecules composed of two strands of nucleotides that form a double helix (fig. 2.1). Each of the nucleotides in each of the strands is composed of a sugar residue, a phosphate residue, and one of four bases: adenine (A), guanine (G), thymine (T), and cytosine (C). The sugar and phosphate residues are identical in each nucleotide and form the backbone of the DNA strands, while the bases vary from nucleotide to nucleotide. It is the order in which these bases occur that determines how a gene functions (e.g. which protein it produces), and it is common to refer to this order as a gene sequence.

The chemical nature of the bases is such that only adenine and thymine bond with each other, and only guanine and cytosine bond with each other. Adenine and thymine are therefore said to be each others complement, as are cytosine and guanine. The two DNA strands that compose a DNA molecule are, because of the bonding properties of the bases, base-by-base complements of each other in order for the two strands to bond and form a DNA molecule. This bonding process is called hybridisation, and is, as will be discussed in section 2.4, utilised by the microarray technology. The DNA of an organism

can be divided into two groups: coding and non-coding regions. Non-coding regions make up the majority of the DNA and have no known function. Coding regions are segments of DNA that contain the genes. The coding regions are distributed, seemingly at random, in the DNA separated by non-coding regions.

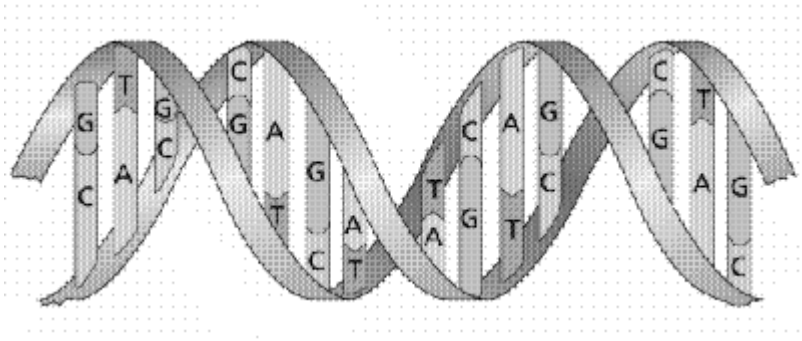


Fig. 2.1: A schematic description of the structure of a DNA molecule excerpt The two broad spiralling bands represent the sugar-phosphate backbone of the two DNA strands. The horizontal bars represent the complementary bases that bind the two strands together. (Illustration adapted from http://www.emc.maricopa.edu/faculty/farabee/BIOBK/BioBookDNAMOLGEN.html)

2.2.2 Gene expression

The process of transforming a gene into a protein is called gene expression, and a gene that is active in a cell (i.e. the cell produces the protein defined by the gene) is said to be expressed. Although the cells contain the same DNA and the same genes, different cell types differ in their pattern of gene expression (i.e. all genes are not expressed in all cells, nor do all cells produce the same amount of proteins). The gene expression levels of a cell may also change with the state of a cell, or as a cell develops. A cell that, for example, is diseased may produce different proteins or different amounts of proteins than a normal, healthy, cell. Thus, gene expression levels can indicate both the type of a cell and its state.

The two-step process of transforming a gene into a protein starts with the transcription of the DNA segment that makes up the gene into a complementary messenger RNA, or mRNA, molecule. RNA is very similar to

DNA except that RNA is a single stranded molecule, and that the thymine base is replaced by a base called uracil (U). Uracil does, however, possess the same hybridisation properties as thymine (i.e. it bonds with adenine, and vice versa).

The second step of the gene expression process is called translation. This part of the gene expression process translates the mRNA molecule into a protein. Proteins are chains of amino acid molecules, and there are 20 different amino acids that can be combined to form a protein. Amino acids are determined by triplets, or codons, of mRNA bases, and the mRNA molecule is therefore translated into a protein three bases at a time.

Once the gene expression process is completed (see fig. 2.2 for a schematic description of the gene expression process), the chain of amino acids is folded into a completed protein. The folding of the protein is partly determined by the order in which the amino acids occur in the protein sequence, and errors in the gene expression process that replaces, inserts or deletes amino acids in the protein sequence may therefore cause the folding (i.e. the structure) of the

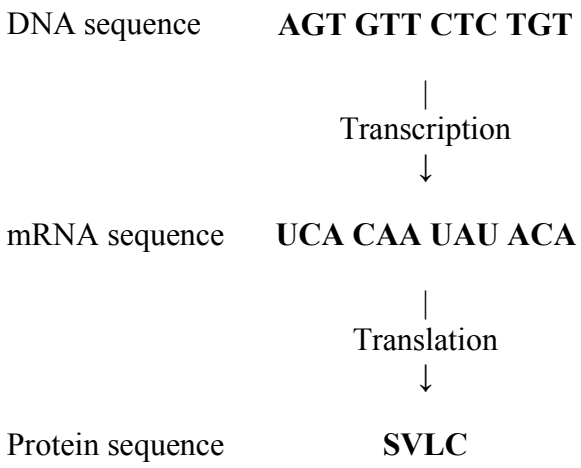


Fig. 2.2: A schematic description of the gene expression process. The DNA-sequence is transcribed into a complementary mRNA sequence, which in turn is translated, three bases at a time, into a sequence of amino acids (i.e. a protein)

protein to change. Changes in the protein structure may in turn cause the protein to malfunction. An error in the gene expression process does not, however, have to affect the protein structure if the error does not change any of the amino acids. For instance, an amino acid may be determined by more than one codon, and an error that changes a codon into another codon that determines the same amino acid will not have any effect on the protein. Thus, different mRNA molecules may produce the same protein even though the mRNA molecules are different. A codon, on the other hand, only translates into one specific amino acid, and an mRNA molecule can therefore only be translated into one specific protein. Thus, it is impossible to use a protein sequence to determine the gene sequence that produced a protein.

For instance, the DNA sequence (DNA sequences can be used to determine a protein sequence because of the complementary relationship between the DNA and mRNA sequences)

TTT TGC

produces a protein sequence composed of the amino acids phenylalanine, F, and cystein, C:

FC

Phenylalanine and cystein are, however, determined by two codons each:

Phenylalanine (F)	Cystein (C)
TTT	TGT
TTC	TGC

Thus, the protein sequence FC can be the product of these four DNA sequences:

TTT TGT

TTT TGC

TTC TGT

TTC TGC

The amount of protein produced by a gene is approximately the same as the amount of mRNA molecules that are transcribed from the DNA sequence of the gene. Since it is easier to measure the amount of mRNA transcripts present in a cell than it is to measure the amount of proteins, the amount of mRNA transcripts present in a cell is used to determine the gene expression levels of a cell.

2.3 Mutations

Mutations refer to changes in the DNA that may be transferred to offspring, and are a natural part in the development of genes and species.

There are different types of mutations, and a mutation may, or may not, affect an organism depending on whether or not the mutation occurs in a coding region, and the type of mutation. Some types of mutations can cause diseases such as cancer, while other types are harmless.

Mutations that occur in coding regions of the DNA (i.e. regions that contains genes) can severely affect an organism because they may change the gene product (i.e. the protein the gene produces) or the function of the gene. Some mutations may change the protein folding or the binding sites of a protein. Changes in protein folding alter the structure of a protein and may cause the protein to malfunction. Binding sites are locations in the protein where the protein binds to other molecules (e.g. proteins), and mutations that change these sites may cause a protein to malfunction. Malfunctioning proteins may cause entire cells to malfunction (e.g. cancer), which in turn may have severe effects on an organism. But again, some mutations may not affect the gene product at all.

Substitutions, or point mutations, are simple mutations in which a base in the DNA sequence is replaced by another. The effects a substitution may have on a gene depends on where in the gene the substitution occurs, whether or not the substitution changes the protein produced by the gene (i.e. replaces an amino acid with another), and if so, how different the replacement amino acid is from the original amino acid. Common types of substitutions are:

- Synonymous substitutions are mutations that replace a base with another without changing the protein (e.g. TGT and TGC both encode the amino acid cystein), and are thus harmless mutations.

- Substitutions that cause an amino acid to be replaced by another, or missense mutations. A substitution of the last base of the codon TGT with a G, for instance, would cause cystein to be replaced with the amino acid tryptophan. Effects of missense mutations may be positive, negative, or none depending on where in the gene it occurs and the difference between the original and the replacement amino acid

- Substitutions that cause an amino acid to be replaced by a termination signal (e.g. TGT → TGA), or nonsense mutations. A nonsense mutation causes a protein terminate prematurely, and the effects are almost always negative, as a part of the protein is removed.

Insertion and deletion are other mutations that can occur. Insertions cause one or more additional bases to be inserted in a gene sequence, while a deletion removes one or more bases from a gene sequence. This kind of mutations may have more dramatic effects on gene products, and may cause diseases like cancer.

2.4 Microarrays

Microarrays are used for various purposes in bioinformatics. A microarray is a glass or polymer slide onto which single stranded DNA, i.e. gene sequences or synthetic DNA, is attached at fixed locations. The DNA that is attached to the microarray has to be single stranded because the microarray technology utilises the hybridisation properties of DNA. The attached DNA is commonly referred to as DNA probes, while the fixed locations are referred to as spots (or features depending on the chosen technology).

A microarray may contain tens of thousands of spots, each of which can consist of millions of identical DNA probes. The DNA probes may be attached to the microarray in different ways depending on the technology that is used. Some technologies print probes of synthetic DNA onto the microarray (e.g. the Agilent technology), some use photolithographic techniques (e.g. the Affymetrix technology) or electrodes (e.g. the CombiMatrix technology) to attach synthetic DNA probes to the microarray, while other technologies

deposit small amounts of solutions that contain DNA, i.e. gene sequences or synthetic DNA, onto the microarray (e.g. the cDNA microarray technology).

One of the applications of microarrays is in gene expression level experiments, that is, experiments that, for example, examine which genes are and are not expressed in a cell type (e.g. a skin cell, a liver cell, a muscle cell, etc.), or which genes are and are not expressed in a cell type under different conditions (e.g. different disease stages or developmental stages). The information gene expression level experiments produce can, for instance, contribute to the development of gene expression profiles (or signatures) for diseased cells such as cancer cells. Gene expression profiles can, for example, be a useful tool for identifying disease in patients and determining how the disease should be treated.

There are different approaches to examine gene expression levels with microarrays. The general approach is to extract mRNA from a tissue sample, label the extract and hybridise the labelled extract with the DNA probes on the microarray. The microarray is then placed in a scanner that detects the amounts of mRNA extract that have hybridised with the probes of each spot, and creates an image of the microarray (fig. 2.3).

The image is then processed and converted into a gene expression data matrix that contain the numerical representation of the expression levels of the genes represented by the spots on the microarray.

One approach is to use cDNA microarrays to measure the gene expression levels in different samples (e.g. liver cells in different disease states) relative to the gene expression levels of a reference sample (e.g. healthy liver cells).

The first step in such an experiment is to prepare the microarrays, one for each sample not including the reference sample. The preparation process involves selecting the DNA probes, i.e. the genes whose expressions are to be measured, and depositing the DNA probes onto the microarrays.

The next step is to extract mRNA transcripts from the different samples, and the reference sample. The mRNA extracts of the different samples are commonly referred to as targets, while the mRNA extracted from the reference sample is referred to as a reference. The targets are then labelled with a fluorescent red dye, while the reference is labelled with a fluorescent green dye. Each target is then mixed with reference in equal amounts into target-reference mixtures, one mixture for each target. The target-reference mixtures are washed over one microarray each to allow the mRNA of the mixtures to hybridise with the probes of the microarrays.

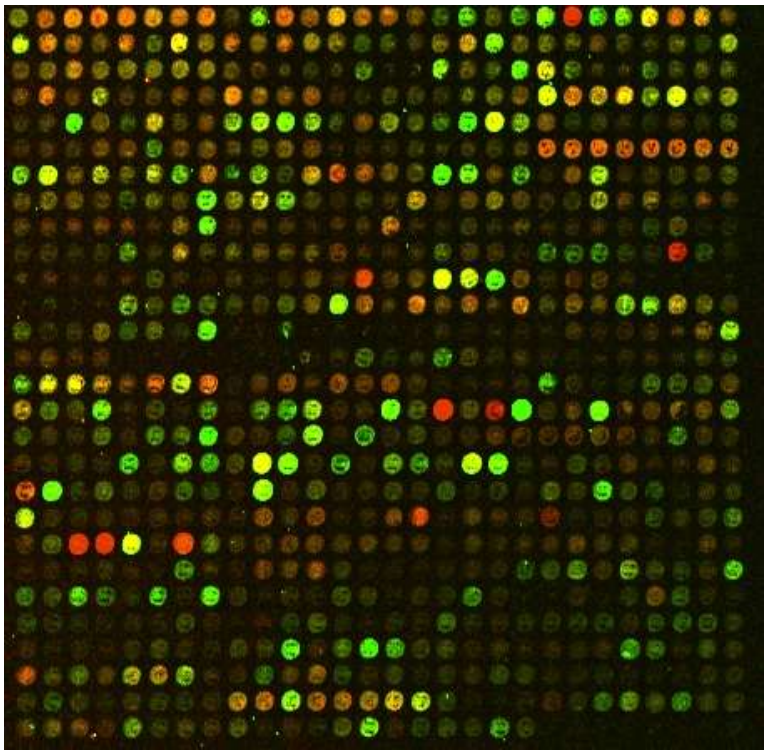


Fig. 2.3: An image of a microarray after hybridisation. The red circles represent genes that are over-represented in the target, the green circles represent genes that are under-represented in the target, yellow circles represents genes that are equally represented in the target and the reference, while black circles represent genes that is not represented in the target or the reference. The colour intensities of the circles represent the amount of mRNA-transcripts present in the target and/or the reference, i.e. a high green intensity indicates that a high amount of the mRNA-transcript of a gene is present in the reference, a high red intensity indicates that a high amount of the mRNA-transcript of a gene is present in the target, while a high yellow intensity indicates that a high amount of the mRNA-transcript of a gene is present in both the target and the reference. (Illustration adapted from http://smf.stanford.edu/cgi-bin/data/clickable.pl?fullID=16954GENEPIX16954)

The mixtures are allowed to hybridise with the probes of the microarrays for a certain period of time before the excess mixture, i.e. the mRNA that has not hybridised, is removed from the microarrays. The microarrays are then, one by one, placed in a scanner that creates an image of each microarrays based on the fluorescence of the spots of each microarray. The images depict the spots of each microarrays as coloured or black circles (fig. 2.3). A black circle means that no mRNA transcripts have hybridised with the probes of a spot, indicating that the gene represented by the spot represents is unexpressed in both the reference and the target. A green circle means that more mRNA transcripts from the reference than from the target have hybridised with the probes of a spot, indicating that the gene that the spot represents is underexpressed (i.e. less active) in the target. A red circle is the opposite of a green circle, meaning that more mRNA transcripts from the target than form the reference has hybridised with the probes of the spot, and the gene is therefore underexpressed in the reference. A yellow circle means that approximately the same amount of mRNA transcripts from the reference and the target has hybridised with the probes of the spot, indicating that the gene is more or less equally expressed in the reference and the target.

The next step is to transform each image into a spot quantitation matrix where the colour intensities of the spots in each image are translated into a numerical quantity. This process translates underexpression in the target (green spots) into negative values, underexpression in the reference (red spots) into positive values, and equal expression (yellow spots) into values around 1.

The last step before the data generated by a microarray experiment can be analysed is to convert the data of the spot quantitation matrices into a gene expression data matrix. A gene expression data matrix is an n x m matrix in which each column represents a sample (i.e. a target), while each row represents a gene. The values in a row therefore represent the expression levels of a particular gene in the different samples, while the values in a column represent the expression levels of the different genes in a particular sample. Hence, the rows of a gene expression data matrix are referred to as gene expression profiles, while the columns are referred to as sample expression profiles. The gene expression data matrix is the basis for all analysis of gene expression. Figure 2.4 shows a gene expression data matrix in combination with a heat map.

A heat map is a gene expression data matrix presentation form that presents the values of the gene expression data matrix as coloured squares; red squares for positive expression, green for negative expression, and black for equal expression.

Note that none of the steps involved in a microarray experiment is trivial. The description given above should therefore be regarded as a simplification.

Fig. 2.4: A combined gene expression data matrix and heat map that shows the gene expression levels of 7 genes in two samples (x and y) as represented by the BioTeach system. A green background indicates under-representation of a gene in the target (i.e. negative expression level), a red background indicates over-representation of a gene in the target (i.e. positive expression level), while a black background indicates equal representation.

2.5 The Sourcer's Apprentice

The Sourcer's Apprentice (SA) developed by Britt & Gabrys (2001) is meant to teach high school students the skill of sourcing and corroboration, two literacy skills that are required to use multiple sources of information correctly. The SA-system is a web-application in which the students are to identify crucial source information about the document excerpts they are presented with. Students are then to answer questions regarding this information, and write essays in which they use the information they have found.

First-time users of the Sourcer's Apprentice receive instructions on how to use the application through an interactive skills tutorial. This tutorial describes the information that is important to identify when using multiple sources of information, how this information can be found, how the information can be used when writing papers, and how the information can be used to evaluate the trustworthiness of the excerpts. The information is broken down into components (e.g. the author, when the document was written, the publisher etc.). Each component is discussed separately, and each discussion is followed

by two control questions that are meant to ensure that the students have understood both how to find the information component and how to use it.

Students that have been through the tutorial can enter the practice environment (fig. 6.1) in which they are to practice on finding the information components discussed in the tutorial in a set of excerpts. The excerpts the students are to read are represented by a shelf of books, and the students have to click on the books to access the excerpts. Each book contains four scrollable pages; a table of contents, a page about the author (contains name, credentials etc.), a page about the document (contains publisher, when the document was written etc.), and a page with the actual excerpt. Each book is also accompanied by a structured note card that lists the information components the students are to find and insert into the note card.

An information component is inserted into a note card by selecting the text-phrase that contains the information and dropping the text into a bucket corresponding to the information component. A correct answer is awarded with points and the component is displayed in the note card. An incorrect answer produces a hint. The hints become clearer each time an incorrect answer is dropped into a bucket, and each hint received reduces the points the correct answer is awarded with. After a certain number of incorrect attempts the hints more or less become instructions for finding the correct text-phrase. Thus, all students should eventually be able to fill in the note cards correctly.

The Sourcer's Apprentice also offers help if any student should need a reminder of what kind of information the different buckets accept. Clicking on a bucket produces the same instructions that were used in the tutorial. Britt & Gabrys regarded three of the information components, the author's motive for writing the text, the main point of the text, and comments, to be too awkward to answer by the drag and drop technique. Instead of using the excerpts to fill in the note cards, students have to construct an answer to the three mentioned components from the respective help texts. These answers are not evaluated by the application, but are graded by the teacher.

When the note cards are filled in, the students are required to write an essay about the excerpts they have read supported by the note cards, which are available during writing. After finishing the essay, the students have to answer several questions that are meant to ensure that the students have acquired the literacy skills interspersed with questions about the contents of the texts. These questions are answered in the same way as the note cards are filled in, that is, by dragging the correct answer from one of the books into an answer bucket. The scoring system is also the same; if the answer is correct, the students receive a number of points, if it is incorrect, the number of points is decreased for each hint the student receives.

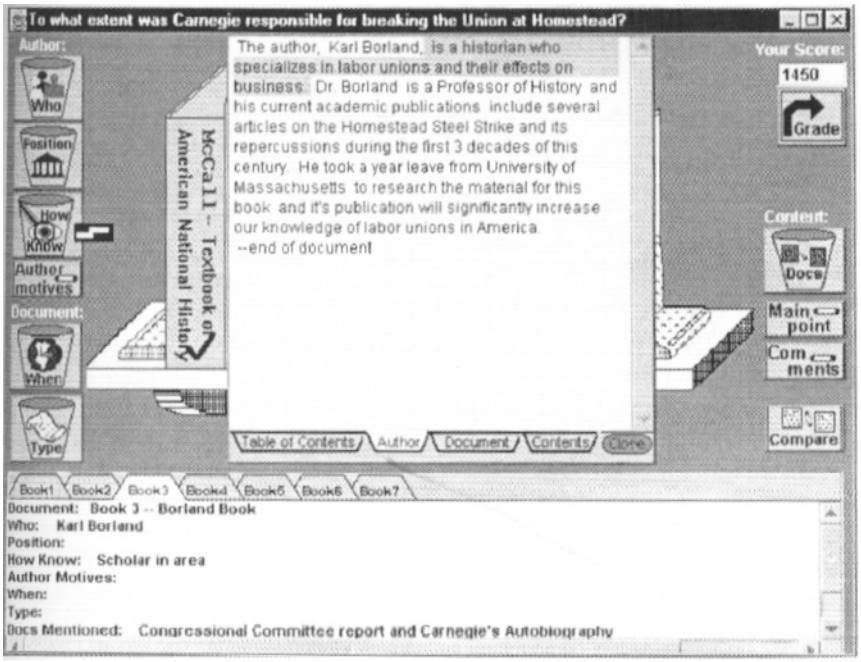


Fig. 6.1: The practice environment of the Sourcer's Apprentice with the historical text at the centre, the structured note cards at the bottom, and the buckets along each side. The selected text has been dropped into the How Know bucket, and appears in the How Know slot in the structured note card at the bottom.

2.5.2 Empirical results for the Sourcer's Apprentice

Britt & Gabrys (2001) performed two studies of the effects of the use of the Sourcer's Apprentice. Both studies used a pretest-posttest methodology, and both studies showed that students that had been exposed to the Sourcer's Apprentice showed better sourcing skills than students who had not been exposed.

Given the good empirical results it could be useful to base web-based exercises in bioinformatics on the principles and the features of this system.

Chapter 3

Pairwise global sequence alignment

3.1 Introduction

Pairwise global sequence alignment (Durbin *et al.*, 1998; Baxevanis, 2005) is a technique that is used to examine whether or not genes are related, that is, whether or not genes have evolved from the same gene through different sets of mutations. This technique works on pairs of gene sequences, and determines the relationship between a pair of sequences by aligning (i.e. comparing) the two sequences residue by residue (e.g. amino acid by amino acid). Related genes are interesting for various reasons, and pairwise sequence alignment is therefore used for various purposes.

Related genes tend to function in the same, or at least in a similar, way. Thus it is possible to use pairwise sequence alignment to gain insight into the function of a gene with unknown functionality by examining the gene's relationship to genes with known functionality.

Another application of pairwise sequence alignment is in studies that examine the evolution of species. Related genes have, as mentioned, evolved from the same gene through different sets of mutations, and by comparing related genes from different species (e.g. humans and gorillas), it is possible to estimate the amount of time that has passed since the species separated from their common ancestor and became separate species.

Pairwise global sequence alignment can be used on pairs of DNA sequences and pairs of RNA sequences, as well as on pairs of protein sequences. The alignment visualisation of the BioTeach system is implemented to align protein sequences, and protein sequences are therefore used in the examples of this chapter.

A global alignment of two protein sequences S1 and S2, here WGQMNSFS and AMNESFQS, might look something like this:

```
W G Q M N - S F - S
- A - M N E S F Q S
```

The letters in the alignment represent the amino acids of the sequences, while the character '-' represents insertions or deletions that may have occurred during the evolution of either of the genes. The '-' character is commonly referred to as a gap, or as an indel symbol.

When and where to insert gaps in an alignment depends both on the algorithm that is used to align the two sequences and the algorithm parameters that are used. There are, however, certain rules that any global alignment algorithm has to follow:

1) An amino acid in one sequence can be aligned with an amino acid or a gap in the other sequence. A gap cannot be aligned with a gap.

2) The aligned sequences (each of which contain zero, one or more gap symbols) should have equal length, so that the i-th residue of one sequence is aligned with the i-th residue of the other sequence.

Inserting a gap in one sequence (e.g. column 3) means that an amino acid has been deleted inserted in the other sequence, or that an amino acid has been deleted from the sequence in which the gap is inserted. It is, however, impossible to know which of the two alternatives has occurred. It is not possible to know in which of the sequences a substitution (e.g. column 2) has occurred either.The previous alignment example showed an alignment of two sequences of equal length (i.e. with the same number of residues). It is, however, possible to align two sequences of different lengths:

```
A C Q K M W F S
- - Q R - W - S
```

As this example shows, gaps are always inserted in such a manner that the lengths of the sequences become identical in the alignment.

One of the problems with pairwise global sequence alignments is that gaps make it possible to align two sequences in a number of ways. A global alignment of a pair of sequences of two residues, for instance, has 13 possible alignments (see fig. 3.1), and the number of possible alignments increases rapidly as the length of the sequences increases. The problem with multiple possible alignments is simply to determine which of the alignments are optimal, that is, to determine which of the alignments represents the mutation sets that are most likely to have occurred naturally.

Another problem is to determine whether or not an global alignment of one pair of sequences is more optimal than a global alignment of a different pair of sequences. There are different ways of solving these two problems, of which the Needleman-Wunsch alignment algorithm is one.

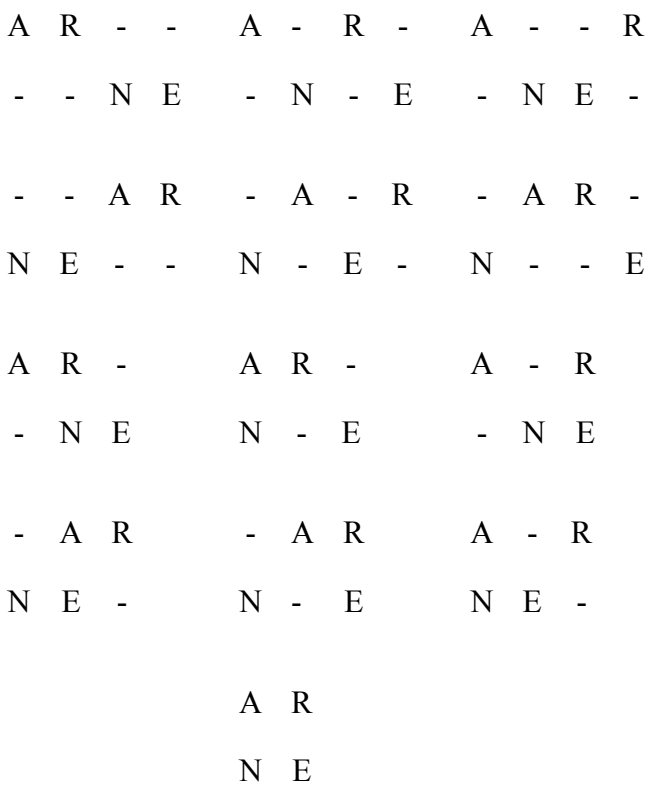


Fig. 3.1: The 13 possible alignments of the protein sequences AR and NE

3.2 The Needleman-Wunsch alignment algorithm

The Needleman-Wunsch alignment algorithm is one of the algorithms that is used to find the optimal alignment of a pair of sequences, and is the algorithm that is visualised by the BioTeach system. It belongs to a group of algorithms called dynamic programming algorithms, and is used for finding optimal global alignments; global meaning that the algorithm finds the optimal alignment of complete sequences, not parts of them as is the purpose of local alignment algorithms.

The Needleman-Wunsch algorithm finds the optimal alignment of a pair of sequences by optimising a score function, that is, each possible alignment is scored according to a score function, and the alignment that yields the highest score is the optimal alignment of pair of sequence. If the score of more than one of the possible alignments equals the highest score, there is more than one optimal alignment of the pair of sequences.

The score of an alignment is computed by assigning each pair of aligned residues and each residue-gap pair with a score term, and summing these terms. The sum of these terms is then the score of an alignment. In order for the score of an alignment to provide any information about the optimality of the alignment, it is important that the assigned score terms reflect how mutations occur in nature. Insertions and deletions of residues are observed less frequently than substitutions and conservations of residues. Alignments that include insertions and deletions, i.e. gaps, should therefore yield a lower score than alignments that do not include gaps. Introducing gaps into an alignment is therefore penalised by assigning residue-gap pairs with a negative score term, a gap penalty. Some substitutions are observed more frequently than others, and the score terms of substitutions may therefore be both positive and negative. Conservations of residues are observed more frequently than both insertions/deletions and substitutions, and the score terms for conservations are always positive. Some conservations are, however, observed more frequently than other, and the score terms of conservations may therefore vary. Score terms that corresponds to the observed frequencies of the different substitutions and conservations have been computed, and are organised in substitution matrices.

A substitution matrix (fig. 3.2) is a two-dimensional matrix in which each row and each column contain the score terms for each of the possible substitutions of a residue, and the score term for conservation of the residue. A score term for a pair of aligned residues is given in the intersection of the row that represents the first residue in the pair and the column that represents the other residue in the pair.

It is, however, due to the numerous possible alignments of a pair of sequences, inefficient to separately compute the score of each possible alignment and then compare these scores to find the optimal alignment (or alignments) of a pair of sequences. Instead, the Needleman-Wunsch algorithm treats the optimal global alignment as a construction composed from the optimal alignments of pairs of subsequences (i.e. shorter parts of the sequences).

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

Fig. 3.2: The Blosum62 substitution matrix. Each value in a row represents the score of aligning the amino acid at the head of the row with each of the other amino acids. The score of aligning a pair of amino acids is found where the row of the first amino acid and the column of the second amino acid (or vice versa) intersect. B and Z are ambiguity codes and the values in those rows (or columns) represent the score of aspargine or aspartic acid (B) or glutamine or glutamic acid (Z) with the other amino acids. X represents any amino acid and a value in this row (or column) represents aligning the amino acid at the head of a column with any amino acid. The * represents the lowest possible score in a row or column.

With this approach it is possible to view the alignment of the sequences WGQMNSFS and AMNESFQS given as an example in the previous section

W	G	Q	M	N	-	S	F	-	S
-	A	-	M	N	E	S	F	Q	S

as being composed from the optimal alignment of the subsequence pair WGQMNSF and AMNESFQ, which in turn is composed from the optimal alignment of the subsequence pair WGQMNSF and AMNESF, which in turn is composed from the optimal alignment of the subsequence pair WGQMNS and AMNES, and so forth. Thus, the problem of finding the optimal alignment is decomposed into finding the optimal subsequence alignments that compose the optimal global alignment of a pair of sequences. This problem is solved by constructing a score matrix, F (fig. 3.3), composed of the optimal scores of the optimal subsequence alignments, and following the optimal path from the last cell of the score matrix to the first cell.

	-	P	W	R
-	0	-5	-10	-15
W	-5	-4	6	1
Q	-10	-6	1	7
N	-15	-11	-4	2
S	-20	-16	-9	-3

Fig.3.3: The score matrix of the alignment of the sequences PWR and WQNS using a gap penalty of -5 and the Blosum62 matrix as represented by the BioTeach-system. Each cell represents the optimal alignment of the *i* (i.e. the column number) first residues of the first sequence, here PWR, and the *j* first sequences of the second sequence, here WQNS. The arrows represents the pointers of the cells, and indicate through which of the neighbouring cells the path to the first cell, cell *F(0, 0)*, is.

A score matrix for the alignment of a sequence A of *i* residues and a sequence B of *j* residues is represented as a matrix of *(i +1)* rows and *(j + 1)* columns excluding the header row and column. The residues of sequence A is given in the cells of the header row preceded by an indel symbol, while the residues of sequence B is given in the cells of the header column, also preceded by an indel symbol. The columns of the score matrix are numbered from 0 to *i*, while the rows are numbered from 0 to *j*. Hence, a cell in a score matrix is referred to as *F(i, j)*. The construction of the score matrix starts with initialising the first cell of matrix, cell *F(0, 0)*, to zero. The scores of the remaining cells of the first row are then computed with the expression

$$F(i, 0) = i * \text{gap penalty}$$

Each of these cells represents the alignment of the *i* first residues of sequence A and a gap, as well as the score of each of these alignments.

A pointer to the preceding cell is also included in each of these cells to indicate that the path from each cell to the first cell of the matrix is through the preceding cell.

The next step is to compute the scores of the remaining cells of the first column. The scores of these cells are computed with the expression

$$F(0, j) = j * \text{gap penalty}$$

Each of these cells represents the alignment of the j first residues of sequence B and a gap, as well as the score of each of these alignments. A pointer to the preceding cell is included in these cells for the same reason as a pointer was included in each of the cells of the first row.

The scores of each of the remaining cells, $F(i, j)$, are then determined, row by row, by extending the alignments represented by the cells $F(i-1, j)$, $F(i-1, j-1)$, and $F(i, j-1)$ with

1) the alignment of the i-th residue of the sequence A and a gap

2) the alignment of the i-th residue of sequence A and the j-th residue of sequence B

3) the alignment of the j-th residue of sequence B and a gap

respectively. The extension that yields the highest score according to the score function is the optimal alignment of the i first residues of sequence A and the j first residues of sequence B. A pointer to the cell that represents the alignment that was extended is therefore added to cell $F(i, j)$ to indicate that the path from $F(i, j)$ to $F(0, 0)$ is through the cell that represents the alignment that was extended. If the score of more than one extension equals the highest score, there are more than one optimal alignment of sequence A and B. A pointer to each of the cells that represents an alignments that was extended therefore has to be added to cell $F(i, j)$.

Because of the order in which the scores of the cells of the matrix are computed, the scores of cells $F(i-1, j)$, $F(i-1, j-1)$, and $F(i, j-1)$ have already been computed. Thus, the score of each of the extensions can be computed by

1) adding the gap penalty to the score of cell $F(i-1, j)$

2) adding the substitution matrix score term for aligning the i-th residue of sequence A and the j-th residue of sequence B to the score of cell $F(i-1, j-1)$

adding the gap penalty to the score of cell $F(i, j-1)$

The more formal representation of the calculation of the optimal score of a cell is:

$$F(i,j)=\begin{cases} F(i-1,j)+gappenalty \\ F(i-1,j-1)+s(i,j) \\ F(i,j-1)+gappenalty \end{cases}$$

where *s(i, j)* represents the substitution matrix score term for aligning the i-th residue of sequence A with the j-th residue of sequence B.

The optimal alignment of a pair of sequences can be found once the score matrix is completed. The optimal alignment is found by following the path of pointers from the last cell of the score matrix, cell *F(i, j)*, to the first cell, cell *F(0, 0)*. Note that the alignment is built in reverse. The directions of the pointers determine how the sequences are aligned:

1) A pointer from *F(i, j)* up towards *F(i, j-1)* means aligning the j-th residue of sequence B with a gap

2) A pointer from *F(i, j)* diagonally towards *F(i-1, j-1)* means aligning the i-th residue of sequence A with the j-th residue of sequence B

3) A pointer from *F(i, j)* left towards *F(i-1, j)* means aligning the i-th residue of sequence A with a gap

The last cell of the score matrix represents the optimal score of an alignment, and it is this score that is used to determine whether or not an alignment of one pair of sequences is more optimal than an alignment of a different pair of sequences.

Chapter 4

Clustering of microarray data

4.1 Introduction

Clustering (Draghici, 2003; Causton *et al.*, 2003; Berrar *et al.*, 2003) is one of the most popular approaches to analyzing gene expression data. The purpose of this approach is to group genes or samples with similar expression profiles (see fig. 4.1), that is, to group genes with similar expression levels in the different samples, or to group samples in which the different genes are expressed similarly. This approach to microarray data analysis can, for instance, be useful for identifying different types of behaviour, and for reducing the dimensions of the data.

Clustering is a well-established field and a number of algorithms have been developed. Which algorithm performs best depends on the dataset, and complex algorithms are not necessarily better than simple algorithms. In general there are two groups of clustering algorithms: hierarchical and flat.

Only the three algorithms implemented in the clustering application of the BioTeach system will be discussed in this chapter.

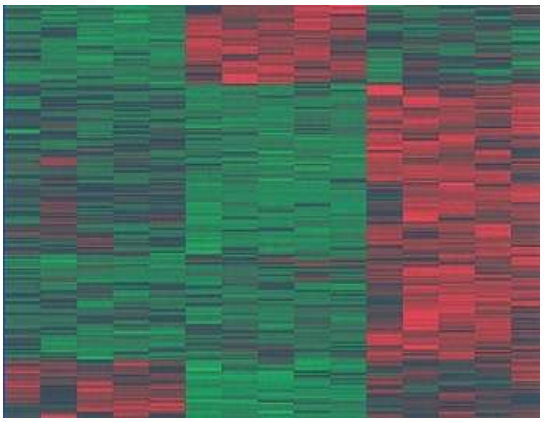


Fig. 4.1: A heat map in which the samples (i.e. the columns) have been grouped in three distinct clusters, while the genes (i.e. the rows) have been grouped in 4 clusters.

4.2 Similarity between objects

The first step in a clustering process would be to define similarity between the objects (i.e. genes or samples) that are to be clustered. By treating the gene expression data matrix as a matrix of m samples (or columns) and n genes (or rows), it is possible to treat a row (i.e. the expression levels of a gene in the m samples) as an m-dimensional vector, and a column (i.e. the expression levels of the n genes in a sample) as an n-dimensional vector.

As it is possible to calculate the distance between vectors of the same dimension, it is possible to treat similarity between genes or samples as the distance between their vectors. There are, however, different ways of calculating the distance between vectors, and which distance measurement to use depends on the purpose of the clustering.

The clustering application of the BioTeach system (chapter 5) uses Euclidean distance as the similarity measurement. The Euclidean distance, d_E, between two n-dimensional vectors $x = (x_1, x_2, \ldots, x_n)$ and $y = (y_1, y_2, \ldots, y_n)$ is

$$d_E = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}$$

This distance measurement corresponds to the distance between two points we humans interpret as the length of a straight line between the two points. The purpose of the clustering application of the BioTeach system is to demonstrate the workings of different clustering algorithms by visualisation, and it would be confusing if the data were clustered according to some distance measure that does not correspond to the distances that are seen in the data on the screen.

Another distance measurement is the Pearson correlation coefficient. The Pearson correlation coefficient compares the components of two n-dimensional vectors to determine whether or not the components vary in the same way. Vectors whose components vary in the same, or similar, way produces a high correlation coefficient and are said to be similar, while vectors with components that do not vary in the same way produce a low correlation coefficient and are said to be dissimilar.

Please refer to chapter 11.2 of Draghici (2003) for information on the different distance measurements.

4.2 Hierarchical clustering

The hierarchical clustering algorithm that is implemented in the BioTeach system is agglomerative. A hierarchical agglomerative clustering starts by treating each entity (i.e. row/gene or column/sample) of the dataset as separate, or singleton, clusters. The two most similar, i.e. the closest, clusters are then grouped together, reducing the total number of clusters by one. This process is then repeated until all entities are grouped in one large cluster.

The hierarchical clustering algorithm can be divided into the following steps (Quackenbush, 2005):

1. Calculate the pairwise distance matrix for all of the singleton clusters to be clustered, that is, to calculate and organise the distances between the vectors representing the genes or samples into a matrix.

2. Search the distance matrix for the two most similar clusters (i.e. the clusters with the least distance between them). This is the true first stage in the clustering process. If several pairs share the same similarity, a predefined rule is used to decide between alternatives.

3. The two selected clusters are merged to produce a new cluster that now contains two or more objects.

4. The distances are calculated between this new cluster and all other clusters. There is no need to calculate all distances because only those involving the new cluster have changed

5. Steps 2 through 4 are repeated until all objects are grouped in one cluster.

In order to make this algorithm work, one has to know the distance between all the clusters, which is rather easy when there only are singleton clusters. When non-singleton clusters are grouped with other clusters, singleton or non-singleton, the distance between all the entities in one cluster and all the entities in the other cluster has to be known, and there are different ways to define the least distance. The three approaches implemented by the BioTeach system are single linkage, complete linkage, and average linkage (see figure 4.2).

Single linkage

Single linkage, or nearest neighbour linkage, group the two clusters with the least minimum distance between two of their respective entities.

Complete linkage

Complete linkage, or furthest neighbour linkage, groups the two clusters with least maximum distance between two of their respective entities.

Average linkage

Average linkage groups the two clusters with least average distance. Average distance is here the average of the distances between all the entities in one cluster and all the entities in the other cluster.

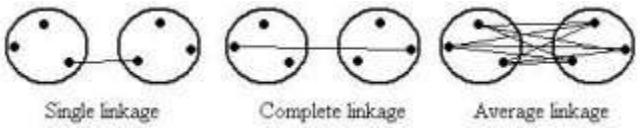


Fig. 4.2: The three different least distance measurements implementet by the BioTeach system. The connected objects represents the objects that define the least distance between two clusters.

The result of the hierarchical clustering approach is represented as a hierarchical tree or dendrogram in which the leaf nodes represent the entities of the dataset. The clusters are found by drawing a horizontal line across the dendrogram, and look at the subtrees below the line.

Figure 4.3 shows the dendrogram of a clustering of 13 genes using complete linkage, in which the horizontal line defines three different clusters. The leaf nodes in the dendrogram represent the entities of the dataset (in figure 4.3, genes).

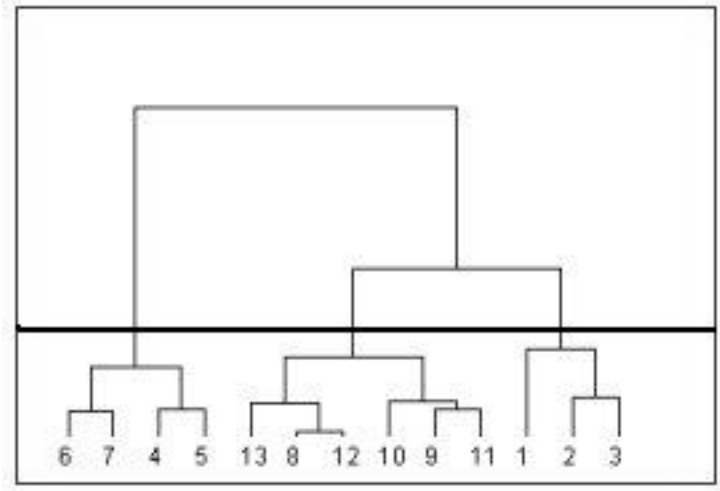


Fig. 4.3: A dendrogram as represented by the BioTeach system. It shows the clustering of 13 genes using complete linkage. The horizontal line defines the different clusters. Here it defines the three clusters A, B, and C: A = 6, 7, 4, 5; B = 13, 8, 12, 10, 9, 11; and C = 1, 2, 3. Different cluster definitions can be found by moving the horizontal line up or down.

It is also possible to implement the algorithm to represent similarity through the length of the branches. This option is implemented in the BioTeach clustering application, and the two most similar genes of figure 4.3 are gene 8 and gene 12.

4.3 K-means clustering

K-means clustering is one of the simplest and fastest algorithms, and is therefore widely used. It is a non-hierarchical algorithm that starts by defining k points as cluster centres, or centroids in the input space (i.e. the n-dimensional space defined by the number of genes, or the m-dimensional space defined by the number of samples).

The algorithm clusters the objects (e.g. genes/rows or samples/columns) of a dataset by iterating over the objects, assigning each object to one of the centroids, and moving each centroid towards the centre of a cluster. This process is repeated until some termination criterion is reached. When this criterion is reached, each centroid is located at a cluster centre, and the objects that are assigned to a particular centroid form a cluster. Thus, the number of centroids determines the number of possible clusters.

Hence, the number of centroids affects the results of the algorithm; a clustering using five centroids would obviously produce different results than a clustering using four centroids. The results are further affected by the initial positions of the centroids; different initial positions may cause an object to be assigned to a different centroid, and the algorithm may therefore yield a different set of clusters. Thus, the number of centroids and their position has to be chosen carefully. There are different ways of implementing this algorithm. The BioTeach system implements two variants: the batch variant and the online variant (The names are taken from Ripley, 1996).

4.3.1 The batch variant

The batch variant of the k-means algorithm can be divided into two steps: object assignment and centroid relocation. The first step, object assignment, starts once the centroids have been placed in the input space. In this step the algorithm iterates over the objects in the dataset and assigns each object to the closest centroid. The next step, centroid relocation, moves each centroid to the

position in the input space that corresponds to the average position of the vectors representing the objects assigned to each centroid.

As the centroids are moved, some of the objects may now be closer to different centroids than the ones they initially were assigned to, requiring the object assignment step to be repeated. As the assignments of the objects are revaluated, some centroids may receive additional objects, while others may have some objects removed. The average position of the vectors representing the objects assigned to a centroid may thus shift, requiring the centroids to be relocated again.

The cycle of object assignment and centroid relocation is repeated until the clusters stabilise (i.e. the objects assigned to the centroids remain the same), or until a predefined maximum number of cycles (e.g. 20.000 to 100.000) has been reached.

The batch variant implemented in the BioTeach system uses the former termination criterion.

Thus, the steps involved in a k-means batch clustering are:

1) iterate over the set of objects, and for each object in the set
    a. find the closest centroid
    b. assign the object to the closest centroid
2) iterate over the centroids, and for each centroid
    a. calculate the average vector of the objects that are assigned to the centroid
    b. relocate the centroid at the position of the average vector of the objects that are assigned to the centroid
3) repeat steps 1 and 2 until the centroids no longer have to be relocated, or until the predefined number of cycles is reached.

4.3.2 The online variant

This variant of the k-means algorithm uses the same approach as the batch variant, that is, it can be divided into the same two steps as the batch variant. The two variants do, however, differ in their execution of the two steps. While the batch variant iterates over the objects of the whole dataset before the centroids are relocated, the online variant moves a centroid at each step of the iteration, that is, each object of the dataset pulls the nearest centroid a certain distance towards itself. In the BioTeach system this distance is 1% of the distance between the object and the centroid. This approach is similar to that of Self Organizing Maps (which are discussed in the next section), and the result is that the centroids appear to be gliding rather than jumping towards the cluster centres of the dataset.

The online variant also uses a different termination criterion than the batch variant. The centroids are only moved a slight distance each time, and the objects assigned to a centroid could therefore appear to be stable for a while, but, as the centroid moves towards the cluster centre, it could move in such a way that it become the closest centroid to objects that are assigned to other centroids. Thus, it is possible for a centroid to "steal" objects from other centriods and change an object assignment that seemed to be stable. The termination criterion used in the batch variant would, in such cases, cause the algorithm to terminate prematurely. One way of ensuring convergence, and to avoid premature termination, is to reduce the distance a centroid is moved gradually over a number of iterations. The termination criterion implemented in the BioTeach system will be discussed in the next chapter.

The steps involved in a k-means online clustering are thus:

1. iterate over the set of objects, and for each object
    a. find the closes centroid
    b. move the closest centroid a certain distance towards the object
2. repeat step 1 until termination criterion is reached.

4.4 Self organizing maps

Self organizing maps, or self organizing feature maps (hence referred to as SOM), is a neural network technique developed by Teuvo Kohonen that can be used to cluster microarray data.

A SOM is a grid (e.g. a line, an array, a cube, a parallelepiped) of units that represent the clusters. The units are organised in neighbourhoods, and the size of the neighbourhoods depends on which grid is used to represent the SOM; the units of a line grid, for instance, would, with the exception of the first and the last unit, have two neighbouring units each, one behind and one in front.

Clustering a dataset with a SOM is achieved by stretching the grid to fit the dataset. This is accomplished by iterating over the entities of the dataset, and for each entity, move the nearest unit a certain distance towards the entity. Because each unit is a part of a neighbourhood, the neighbours of a unit are also moved. The neighbours are only moved a portion of the distance a unit is moved, but the neighbours are moved in such a way that the distances between a unit and its neighbours mirror the similarity between the cluster a unit represents and the clusters the neighbours represent.

This process is repeated until some convergence criterion is reached. A common way of ensuring convergence is to reduce the distance the units and its neighbours are moved, and/or to reduce the size of the neighbourhoods gradually until the units stops moving. When convergence is reached, the grid that represents the SOM has been stretched to fit the dataset in such a way that the organisation of the units represents the relationships which are found between the objects in the dataset.

The steps involved in a clustering with a SOM are then:

1. iterate over the set of objects, and for each object
    a. find the closest unit
    b. move the closest unit a certain distance towards the object
    c. move the neighbouring units a certain distance towards the object
2. repeat step 1 until convergence is achieved

Chapter 5

The BioTeach system

5.1 Introduction

The purpose of developing the BioTeach system is to explore how the Needleman-Wunsch alignment algorithm discussed in chapter 3.2, and the clustering algorithms discussed in chapter 4 (i.e. hierarchical clustering, k-means clustering, and self organising maps) can be visualised in a web-based learning environment for bioinformatics. These algorithms were chosen because they represent two central, yet entirely different, fields in bioinformatics.

The BioTeach system is composed of a Java-based Needleman-Wunsch alignment application, a Java-based clustering application, and a HTML-based portal. The alignment application implements a visualisation of the Needleman-Wunsch algorithm and a Needleman-Wunsch exercise, while the clustering application implements a visualisation of the three clustering algorithms.

This chapter does not discuss the details of the technical solution of the BioTeach system. Readers who are interested in technical details about the implementation, installation, and configuration of the system are advised to refer to the appendices and the source code. The source code can be

downloaded by selecting the download-link from the visualisation menu of the system.

5.2 The portal

The idea with the portal (fig. 5.1) is, apart from providing a single access point to the two applications, to create a uniform and easily navigable environment for the applications.

The portal is composed of a visualisation menu that provides access to either web applications, a help menu that provides explanations of the Needleman-Wunsch and clustering algorithms, a header, and a footer. The layout of the portal is such that these four elements constitute the four edges of a frame: the header is the top edge, the footer the bottom edge, the visualisation menu the left edge, and the help menu the right edge. The space within the frame is reserved for the two web applications, that is, once either of the applications is chosen from the visualisation menu, the chosen application opens within the frame. Thus, the frame remains the same for both applications, creating a uniform environment that at all times provides access to the visualisation menu and the help menu.

The help menu is designed to open the explanations of the algorithms in a separate browser window in order to allow an application to be run and an explanation to be viewed simultaneously.

The portal and the web applications can be accessed at:

http://www.ifi.uio.no/bioinf/Projects/BioTeach

Selecting the alignment application from the visualisation menu opens the main page of the alignment application. This page provides links to the Needleman-Wunsch visualisation and the exercise. Selecting the clustering application from the menu opens the introduction page of the clustering visualisation. This page explains the components of the clustering visualisation interface, and provides a link that starts the visualisation application.

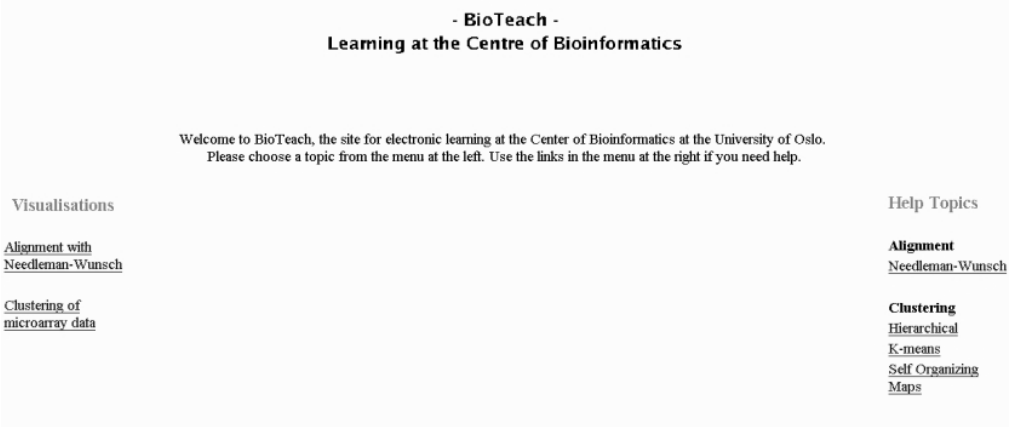


Fig. 5.1: The BioTeach portal with the implemented visualisations to the left, and the help menu to the right.

5.3 The Needleman-Wunsch visualisation

The Needleman-Wunsch visualisation is designed to teach how the Needleman-Wunsch algorithm aligns protein sequences by visualising the steps that are involved in protein sequence alignments. The key to understanding the algorithm is to understand how the score matrix is computed. The focus of the visualisation is therefore on demonstrating how a score matrix is computed, and how the optimal alignment is found in the score matrix.

The score matrix (fig. 5.2) of the Needleman-Wunsch visualisation is based on the principles that were discussed in chapter 3. The amino acids that compose the first protein sequence, sequence A, of an alignment are given, preceded by an indel symbol, in the header row, while the amino acids that compose the second protein sequence, sequence B, are given, also preceded by an indel symbol, in the header column. The rows of the matrix are numbered from 0 to *i*, while the columns are numbered from 0 to *j*. The cells of the matrix are assigned with a set of coordinates on the form *(i, j)*, where *i* refers to the column and *j* the row in which the cell can be found. Each cell, except the cells in the first row and column, represents the optimal alignment of the i first amino acids of sequence A and the j first amino acids of sequence B, and presents the optimal score of these optimal alignments. Each cell also includes pointers that indicate through which of the neighbouring cells the path to the

first cell continues. The cells in the first row represent aligning each of the i first amino acids of sequence A with a gap, while the cells in the first column represent aligning each of the j first amino acids of sequence B with a gap. The matrix is referred to as *F*, and a cell is referred to as *F(i, j)* The upper left cell is the first cell of the matrix, and is referred to as cell
F(0, 0).

The score matrix is supplemented with a computation table (fig. 5.2) that presents the expressions and the calculations that are involved in the computation of the score matrix. The computation table also indicates how the directions of the pointers are determined. The idea with providing this information is to show how the expressions are used during the computation of a score matrix. The computation table also shows which cell is being processed to make it possible to verify the calculations presented by the computation table without having to count columns and rows.

The visualisation also provides a link (fig. 5.2) to the substitution matrix that is used in an alignment. The purpose of providing this link is to allow the calculations that include values from the substitution matrix to be verified. The substitution matrix opens in a separate browser window in order to allow the substitution matrix and the visualisation to be viewed simultaneously.

The visualisation is rounded off with a set of navigation buttons that controls the progress of the visualisation. There is a next-button that displays the next step in an alignment, a previous-button that displays the previous step in an alignment, and a finish-button that completes an alignment and displays the completed score matrix and the optimal alignments. There is also a button that aborts a visualisation, and a button that allows a new alignment visualisation to be configured.

The Needleman-Wunsch visualisation is implemented with two types of visualisations: a simple visualisation and an advanced visualisation.

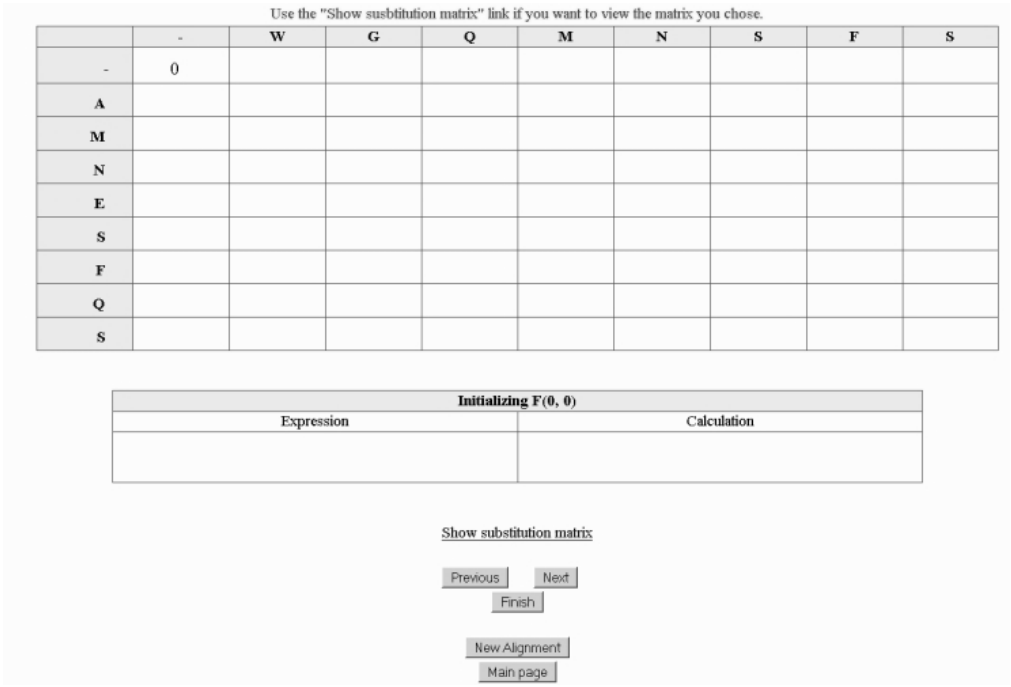


Fig. 5.2: The visualisation interface with the score matrix, the computation table, the link to the substitution matrix, and the buttons.

5.3.1 The simple visualisation

The simple visualisation is meant to be an introduction to the Needleman-Wuncsh algorithm, and is intended for first-time users who have limited experience with the algorithm. The simple visualisation demonstrates how the Needleman-Wunsch algorithm aligns the sequence pair LWA and RSP, using a gap penalty of -5 and the Blosum62 substitution matrix. This visualisation begins with a page that describes the input parameters that are used in the visualisation, and a button that starts the visualisation. Since this visualisation demonstrates the alignment of a specific pair of sequences using a specific gap penalty and substitution matrix, the button that allows a new alignment visualisation to be configured is excluded from this visualisation.

5.3.2 The advanced visualisation

The advanced visualisation allows the users to define the input parameter of the alignment that shall be visualised, and is intended for users who have some experience with the Needleman-Wunsch algorithm.

The idea with this visualisation is to provide an environment in which it is possible to experiment with different input parameters, and learn how the algorithm works by observing how different input parameters affect an alignment. An advanced visualisation is configured by entering a valid pair of sequences and a valid gap penalty into a form, and by selecting one of the six implemented substitution matrices from a list in the form. The six implemented substitution matrices are: Blosum80, Blosum62, Blosum45, Pam250, Pam120, and Pam30.

The advanced visualisation begins with a page that contains a listing of the valid amino acids, a recommendation regarding a sensible gap penalty (negative or zero), the configuration form, and a button that starts the visualisation (fig 5.3). Entering invalid sequences or an invalid gap penalty and pressing the start-button produces an error message instead of starting the visualisation of the alignment.

A valid sequence is composed of at least one, and maximum eight, of the following amino acids:

A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V, B, Z, X.

The limit of eight amino acids was set because sequences of greater length produce a score matrix that is too wide to fit within the frame of the portal, and it would then be impossible to view the complete score matrix.

A valid gap penalty is any integer in the range

[-9.999.999, 99.999.999]

This range was chosen because gap penalties longer than eight digits, or seven digits for negative numbers, produce scores that are too long to fit on a single line in the score matrix, resulting in a score matrix that is difficult to read.

The fields of the configuration form are implemented such that it is impossible to enter sequences or gap penalties that exceed the limit of eight amino acids or eight digits.

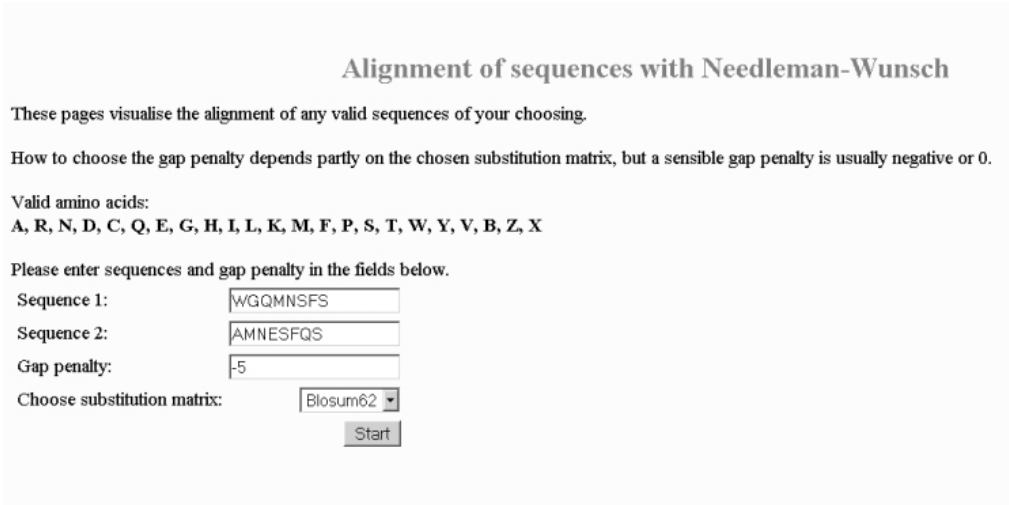


Fig 5.3: The configuration form of the advanced visualisation. The form configures here a visualisation of the alignment of the sequences WGQMNSFS and AMNESFQS using a gap penalty of -5 and the Blosum62 substitution matrix.

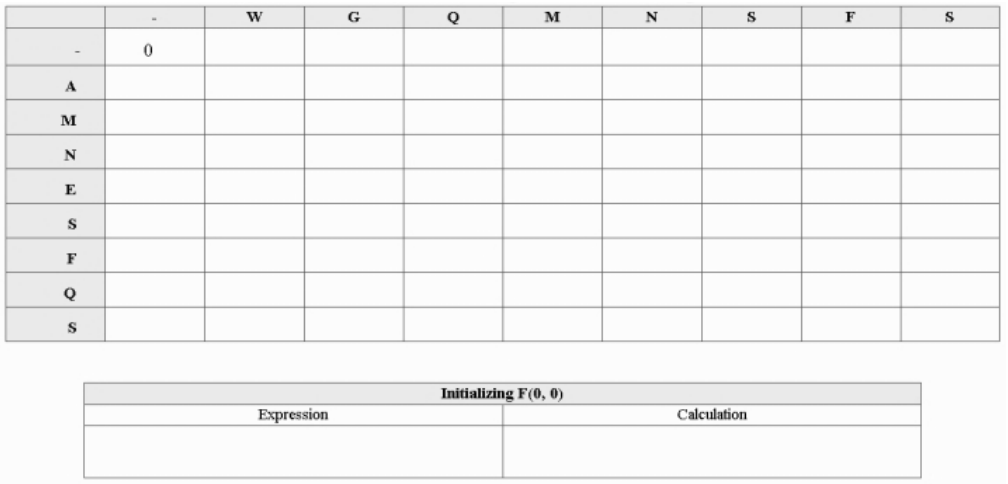
5.3.3 Explanation of the visualisation

All sequence alignments that are conducted with the Needleman-Wunsch algorithm starts with the initialisation of cell *F(0, 0)*, the first cell of the first row, of the score matrix. The score of this cell is predefined to zero. Thus, both the simple and the advanced visualisation starts with a score matrix in which cell *F(0, 0)* has been initialised to zero (fig. 5.4).

Since the score of cell *F(0, 0)* is predefined, there are therefore no expressions or calculations to display in the computation table.

The visualisation continues with the initialisation of the first row and column once cell *F(0, 0)* has been initialised. The initialisation process includes computing the scores of the cells of the first row and column, and setting the direction of the pointers of the cells.

The initialisation process starts with the initialisation of the first row. Thus, pressing the next-button after the initialisation of cell *F(0, 0)* produces a score matrix in which the two first cells of the first row, cell *F(0, 0)* and cell *F(1, 0)*, have been initialised (fig. 5.5). Cell *F(0, 0)* precedes cell *F(1, 0)*, and a pointer



	-	W	G	Q	M	N	S	F	S
-	0								
A									
M									
N									
E									
S									
F									
Q									
S									

Initializing F(0, 0)	
Expression	Calculation

Fig 5.4: An example of the visualisation interface the users are presented with when a visualisation is started. The score matrix at the top shows that the score of cell *F(0, 0)*, the first cell of the first row is zero. The computation table at the bottom shows that the current step in the alignment is the initialisation of cell *F(0, 0)*.

to cell *F(0, 0)* is therefore included in cell *F(1, 0)* to indicate that the path from cell *F(1, 0)* to cell *F(0, 0)* is towards the left.

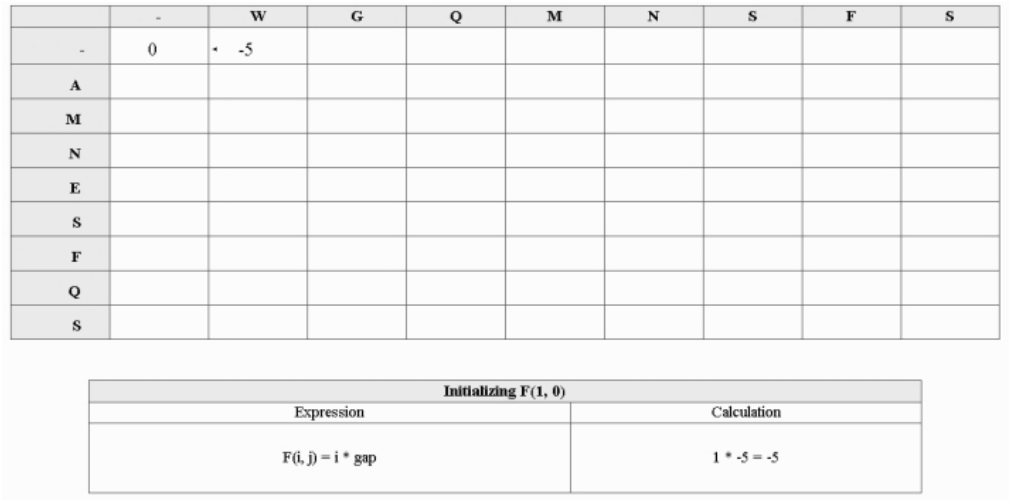
The accompanying computation table now shows that the current step in the alignment is the initialisation of cell *F(1, 0),* and displays the expression and the calculation with which the score of cell *F(1, 0)* were computed (fig. 5.5).

The expression that is used to compute the score of the cell *F(1, 0)* and the other cells of the first row is

*i * gap penalty*

in which i denotes the column that the cell is located in. The computation table also displays that the current step of the alignment is the initialisation of F(1, 0).

The next step in the alignment is the initialisation of the second cell of the first row, cell *F(2, 0)*. Thus, pressing the next-button a second time produces a score matrix in which cells *F(0, 0)*, *F(1, 0)* and the third cell of the first row, *F (2, 0)*, have been initialised (fig. 5.6). The path from cell *F(2, 0)* to cell *F(0, 0)* is through cell *F(1, 0)*, and hence a pointer to cell *F(1, 0)* is added to cell *F(2, 0)*.



	-	W	G	Q	M	N	S	F	S
-	0	◂ -5							
A									
M									
N									
E									
S									
F									
Q									
S									

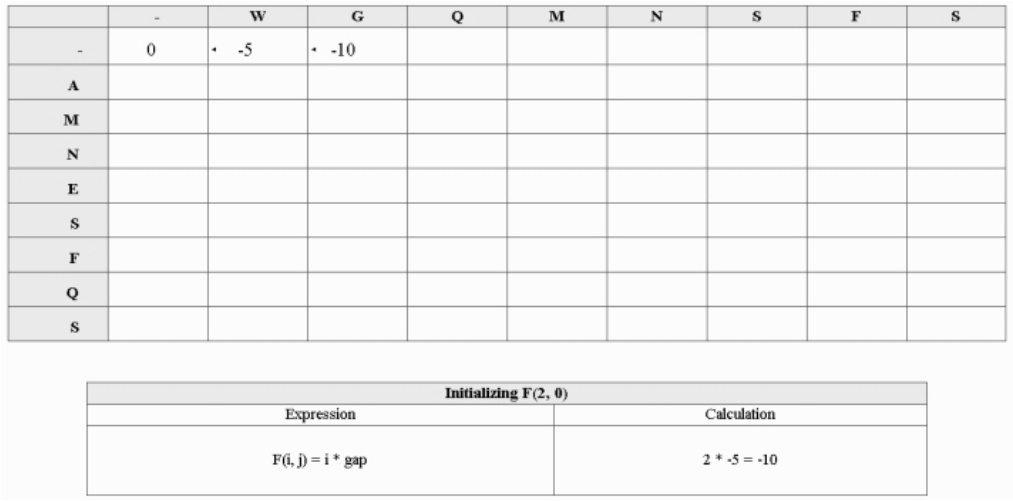
Initializing F(1, 0)	
Expression	Calculation
F(i, j) = i * gap	1 * -5 = -5

Fig. 5.5: An example of the score matrix and the computation table the users are presented with in the first step of the initialisation of the first row of the score matrix. The pointer in the second cell in the first row, cell *F(1, 0)* of the score matrix indicates that the path from cell *F(1, 0)* to cell F*(0, 0)* is to the left. The computation table displays that the current step in the alignment is the initialisation of *F(1, 0)*, the expression that is used to compute the score, and the calculation of the score.

The accompanying computation table now shows that the current step of the alignment is the initialisation of cell *F(2, 0)*, and displays the calculation of the score of cell *F(2, 0)* (fig. 5.6).

This pattern of presentation is repeated for each of the remaining cells of the first row. That is, each time the next button is pressed, a score matrix is presented in which the next un-initialised cell, cell *F(x, 0)*, has been initialised with a score and a pointer to the preceding cell. The accompanying computation table shows that the current step of the alignment is the initialisation of cell *F(x, 0)*, and displays the calculation with which the score of cell F*(x, 0)* was computed.

Once the initialisation of the first row is completed, the next step in the initialisation process is the initialisation of the first column. Thus, pressing the next-button after the initialisation of the first row is completed produces a score matrix in which the cells of the first row, cells *F(0, 0)* to *F(i, 0)*,



	-	W	G	Q	M	N	S	F	S
-	0	← -5	← -10						
A									
M									
N									
E									
S									
F									
Q									
S									

Initializing F(2, 0)	
Expression	Calculation
F(i, j) = i * gap	2 * -5 = -10

Fig. 5.6: An example of the score matrix and computation table the users are presented with at the second step in the initialisation of the first row of the score matrix. The pointer in the third cell in the first row, cell *F(2, 0)*, indicates that the path from cell *F(2, 0)* to cell *F(0, 0)* is through cell *F(1, 0)*. The computation table shows that the current step in the alignment is the initialisation of cell *F(2, 0)*, the expression used to compute the score of cell *F(2, 0)*, and the calculation of the score.

and the second cell of the first column, cell *F(0, 1)*, have been initialised (fig. 5.7). Cell *F(0, 0)* precedes cell *F(0, 1)*, and a pointer to cell *F(0, 0)* is therefore added to cell *F(0, 1)* to indicate that the path from cell *F(0, 1)* to cell *F(0, 0)* is upwards.

The score of cell *F(0, 1)* and the rest of the cells in the first column is computed with the expression

$$j * \text{gap penalty}$$

where *j* denotes the row in which the cell is located. Thus, the computation table (fig. 5.7) that accompanies the score matrix now displays this expression and the calculation of this expression for cell *F(0, 1)*.

The computation table also shows that the current step of the alignment is the initialisation of cell *F(0, 1)*.

The initialisation of the remaining un-initialised cells of the first column is visualised in the same manner. That is, each time the next-button is pressed, a score matrix is presented in which the next un-initialised cell, cell *F(0, y)*, has been initialised with a score and a pointer to the preceding cell.

	-	W	G	Q	M	N	S	F	S
-	0	-5	-10	-15	-20	-25	-30	-35	-40
A	-5								
M									
N									
E									
S									
F									
Q									
S									

Initializing F(0, 1)	
Expression	Calculation
F(i, j) = j * gap	1 * -5 = -5

Fig 5.7: An example of a score matrix and computation table the users are presented with in the first step of the initialisation of the first column of the score matrix. The pointer of the cell in the first row and first column, cell *F(0, 1)*, indicates that the path from cell *F(0, 1)* to from cell *F(0, 0)* is upwards. The computation table shows that the current step in the alignment is the initialisation of cell *F(0, 1)*, the expression that is used to compute the score of cell *F(0, 1)*, and the calculation of the score.

The accompanying computation table shows that the current step of the alignment is the initialisation of cell *F(0, y)*, and displays the calculation with which the score of cell *F(0, y)* was computed.

Once the initialisation process is completed, the next step is to compute the scores and determine the directions of the pointers of the remaining cells, starting with the second cell in the second row and column, cell *F(1, 1)*. Thus, pressing the next-button after the initialisation process is completed produces a score matrix in which the scores and pointers of the cells of the first row, the first column, and cell *F(1, 1)* have been determined (fig. 5.8).

The general expression that is used to compute the score of cell *F(1, 1)* and the other remaining cells is

$$F(i,j) = \max\begin{cases} F(i,j-1) + gap\,penalty \\ F(i-1,j-1) + s(i,j) \\ F(i-1,j) + gap\,penalty \end{cases}$$

	-	W	G	Q	M	N	S	F	S
-	0	-5	-10	-15	-20	-25	-30	-35	-40
A	-5	-3							
M	-10								
N	-15								
E	-20								
S	-25								
F	-30								
Q	-35								
S	-40								

Computing F(1, 1)		s(i, j) = alignment of (W, A)	
Expression		Calculation	Pointer direction
F(i, j) = max {	F(i, j-1) + gap penalty	-5 + -5 = -10	up
	F(i-1, j-1) + s(i, j)	0 + -3 = -3	diagonal
	F(i-1, j) + gap penalty	-5 + -5 = -10	left

Fig. 5.8: An example of a score matrix and computation table the users are presented with when the initialisation of the first row and column is completed. The pointer in the second cell of the second row, cell *F(1, 1)*, indicates that the path from cell *F(1, 1)* to cell *F(0, 0)* is up and to the left. The computation table shows that the current step in the alignment is the computation of cell *F(1, 1)*, that the expression *s(i, j)* represent the score of aligning the first amino acid of the first and second sequence, the expression that is used to compute the score of cell *F(1, 1)*, the calculations that are used to determine the score of cell *F(1, 1)*, and which pointer directions the calculations represent.

where *i* denotes the column in which the cell whose score is being computed is located, and *j* the column. The term *s(i, j)* refers to the substitution matrix score of aligning the i-th amino acid in sequence A with the j-th amino acid in the sequence B.

The score of *F(1, 1)* and the other remaining cells is found, as the above expression shows, by computing the three expressions to the right of the brace (hence referred to as the sub-expressions) for *F(1, 1)* and each of the remaining cells, and for each cell compare the sums the sub-expression yields to find the highest sum. The highest sum is then the score of a cell.

The sub-expressions also represent the three rules an alignment has to obey:

1) the j-th amino acid of sequence B can aligned with a gap (represented by the first sub-expression)

2) the i-th amino acid of sequence A can aligned with the j-th amino acid of sequence B (represented by the second sub-expression)

3) the i-th amino acid of sequence A can be aligned with a gap (represented by the third sub-expression)

The sub-expression that yields the highest score therefore determines the direction of the pointer of a cell, cell *F(x, y)*. Thus:

1) if the first sub-expression yields the highest score, the optimal alignment represented by cell *F(x, y)* is an extension of the optimal alignment represented by cell *F(x, y-1)*, and a pointer to cell *F(x, y-1)* is added to cell *F(x, y)*.

2) if the second sub-expression yields the highest score, the optimal alignment represented by cell *F(x, y)* is an extension of the optimal alignment represented by cell *F(x-1, j-1)*, and a pointer to cell *F(x-1, y-1)* is added to cell *F(x, y)*.

3) if the third sub-expression yields the highest score, the optimal alignment represented by cell *F(x, y)* is an extension of the optimal alignment represented by cell *F(x-1, y)*, and a pointer to cell *F(x-1, y)* is added to cell *F(x, y)*.

It is, however, possible for two, or all three, sub-expressions to yield the same score, and there may therefore be up to three pointers in cell *F(1, 1)* and the remaining cells. A cell with more than one pointer signifies that there is more than one optimal alignment of sequence A and sequence B.

For *F(1, 1)* the sub-expressions are

1) *F(1, 0)* + gap penalty

2) *F(0, 0)* + *s(1, 1)*

3) *F(0, 1)* + gap penalty

Thus, the score of cell *F(1, 1)* is found by

1) adding the gap penalty to the score of cell *F(1, 0)*, the cell above cell *F (1, 1)*

2) adding the score of aligning the first amino acid of the first sequence with the first amino acid of the second sequence to the score of cell *F (0, 0)*, the cell above and to the left of cell *F(1, 1)*

3) adding the gap penalty to the score of cell *F(0, 1)*, the cell to the left of cell *F(1, 1)*

and then comparing these sums to determine which is highest. The pointer (or pointers) is then added to cell *F(1, 1)* according to the rules described above.

The computation table that accompanies the score matrix now displays, apart from that the current step is the computation of *F(1, 1)*, which alignment the term *s(i, j)* represents, the general expression that is used in the computation of cell *F(1, 1)* and the remaining cells, the calculation of the sub-expressions, and the pointer direction each of the sub-expressions represent (fig. 5.9). The calculation (or calculations) that yields the highest sum is, along with the pointer direction it represents, marked with red to make it easier to see the connection between the sub-expression that yields the highest score and the score and pointer (or pointers) that is found in cell *F(1, 1)*.

Computing F(1, 1)		**s(i, j) = alignment of (W, A)**	
Expression		Calculation	Pointer direction
F(i, j) = max {	F(i, j-1) + gap penalty	-5 + -5 = -10	up
	F(i-1, j-1) + s(i, j)	0 + -3 = -3	diagonal
	F(i-1, j) + gap penalty	-5 + -5 = -10	left

Fig. 5.9: An example of a computation table the users are presented with once the initialisation of the first row and column of the score matrix is completed. The computation table shows the current step in the alignment (here the computation of the score of cell *F(1, 1)*), which alignment the term *s(i, j)* represents (here the alignment of the amino acids W and A), the expression that is used to compute the score of a cell, the calculations that are made to compute the score, and the pointer direction each of the calculations represent. The calculation(s) that yield the highest score and the corresponding pointer direction(s) is marked with red to make it easier for the users to connect the score and pointer direction of the cell with the calculations.

The computation of the remaining cells are visualised in the same manner, row by row, until the last cell, *F(i, j)*, has been computed. Thus, pressing the next-button after the computation of the score of cell *F(1, 1)* is completed produces a score matrix in which the scores and pointers of the cells in the first row, the first column, and cells *F(1, 1)* and *F(2, 1)* have been set.

The accompanying computation table shows that the current step in the alignment is the computation of cell *F(2, 1)*, and displays the calculations of the sub-expressions for cell *F(2, 1)*, which of the calculations yield the highest score, and which alignment the term *s(i, j*) represents.

Once the second row, i.e. cells *F(1, 1)* to *F(i, 1)*, is completed, the visualisation continues with the computation of the third row, cells *F(1, 2)* to *F(i, 2)*, followed by the computation of the fourth row, cells *F(1, 3)* to *F(i, 3)*, and so forth until the last row, cells *F(1, j)* to *F(i, j)*, is completed. Each step is accompanied by a computation table similar to that which accompanies the computation of cell *F(1, 1)* and *F(2, 1)* updated to show the current step, the calculations and so forth.

As the last row is completed, the next-button is replaced by a show-optimal-alignments-button. Pressing this button presents the optimal alignment or set of alignments for a pair of sequences. The optimal alignment is found by following the path of pointers from the last cell, cell *F(i, j)* to the first cell, cell *F(0, 0)*, and if there is more than one path from the last cell to the first cell, then there is more than one optimal alignment. The optimal alignment(s) are supplemented with a score matrix in which the optimal path (or paths) has been marked with red to make it easier to verify the optimal alignment(s) (fig. 5.10).

A visualisation of an alignment can, at any time, be completed by pressing the finish-button. Pressing the finish-button produces the same score matrix as was described in the above paragraph, and a listing of the optimal alignment (or alignments).

A visualisation can also, at any time, be aborted by pressing the main page-button, which returns the users to the main page of the alignment application.

	-	W	G	Q	M	N	S	F	S
-	0	-5	-10	-15	-20	-25	-30	-35	-40
A	-5	-3	-5	-10	-15	-20	-24	-29	-34
M	-10	-6	-6	-5	-5	-10	-15	-20	-25
N	-15	-11	-6	-6	-7	1	-4	-9	-14
E	-20	-16	-11	-4	-8	-4	1	-4	-9
S	-25	-21	-16	-9	-5	-7	0	-1	0
F	-30	-24	-21	-14	-9	-8	-5	6	1
Q	-35	-29	-26	-16	-14	-9	-8	1	6
S	-40	-34	-29	-21	-17	-13	-5	-4	5

Optimal alignment:

WGQMN-SF-S
-A-MNESFQS

Fig 5.10: An example of a completed score matrix and the optimal alignments. The optimal path through the score matrix is marked with red to make it easier for the users to verify the optimal alignment(s).

5.4 The Needleman-Wunsch exercise

The Needleman-Wunsch exercise is meant as an example on how an exercise can be implemented in a web-based learning environment, and as a supplement to the Needleman-Wunsch visualisation. The visualisation only demonstrates how the Needleman-Wunsch algorithm works, and, although the visualisation allows the users to define the alignment that is visualised and to control the progress of the visualisation, it does only allow the users to participate in an alignment as observers. The purpose of the Needleman-Wuncsh exercise is to provide a learning environment in which the users can participate actively by defining an alignment and completing the score matrix of the alignment.

Selecting the exercise from the main page of the alignment application opens a page that is identical to the configuration page of the advanced visualisation.

This page contains the same listing of the valid amino acids, the same recommendation regarding the gap penalty, the same configuration form, and a button that starts the exercise (fig 5.11). The limitations on the length of the sequences and the gap penalty apply to this form for the same reason as they were applied to the configuration form of the visualisation.

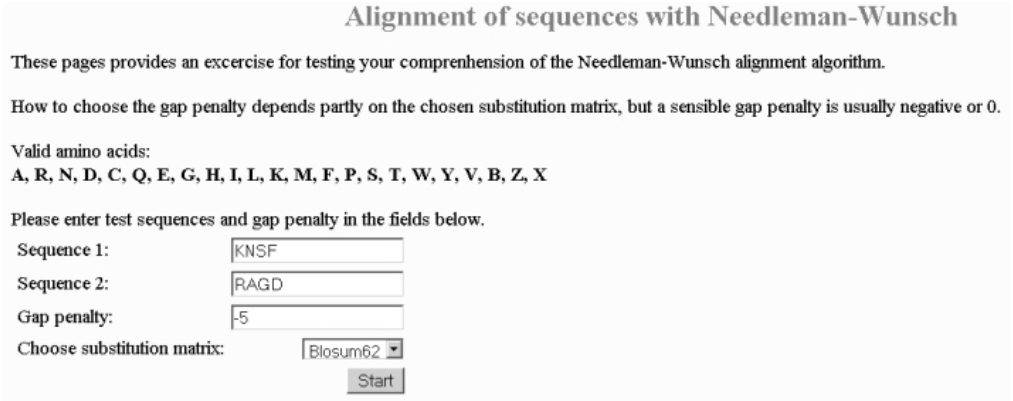


Fig 5.11: An example on a configuration form for the alignment exercise. Similar to the configuration form of the alignment visualisation. This example will generate an exercise in which the users are to complete the score matrix of the alignment of the sequences KNSF and RAGD using a gap penalty of -5, and the Blosum62 substitution matrix.

Pressing the start button opens, given that the sequences and gap penalty are valid, a page (fig. 5.12) that contains the empty score matrix of the alignment defined by the parameters entered into the configuration form, a link to the chosen substitution matrix, a submit button that checks if the scores and direction entered into the score matrix are correct, a button that allows a new exercise to be configured, and a button that exits the exercise and opens the main page of the alignment application.

The cells of the score matrix each contain a text field, in which the correct score is to be entered, and three check boxes. The check boxes are arranged to correspond to the three possible directions the pointers of a cell can have (i.e. left, upwards to the left, and up). A pointer is added to a cell by ticking off a check box, and it is possible to add up to three pointers in each cell.

It may be argued that having to choose between three check boxes in the first cell of the first row, which has no real pointer option, or in the remaining cells of the first row and the first column, which only have one real pointer option each, may be annoying and/or confusing. Using one check box for the cells that only have one real pointer option, and none in the first cell of the first row, would be the obvious alternative to using three check boxes. This alternative could, however, be equally annoying because being forced to select an alternative when there is only one, mandatory, alternative would seem meaningless.

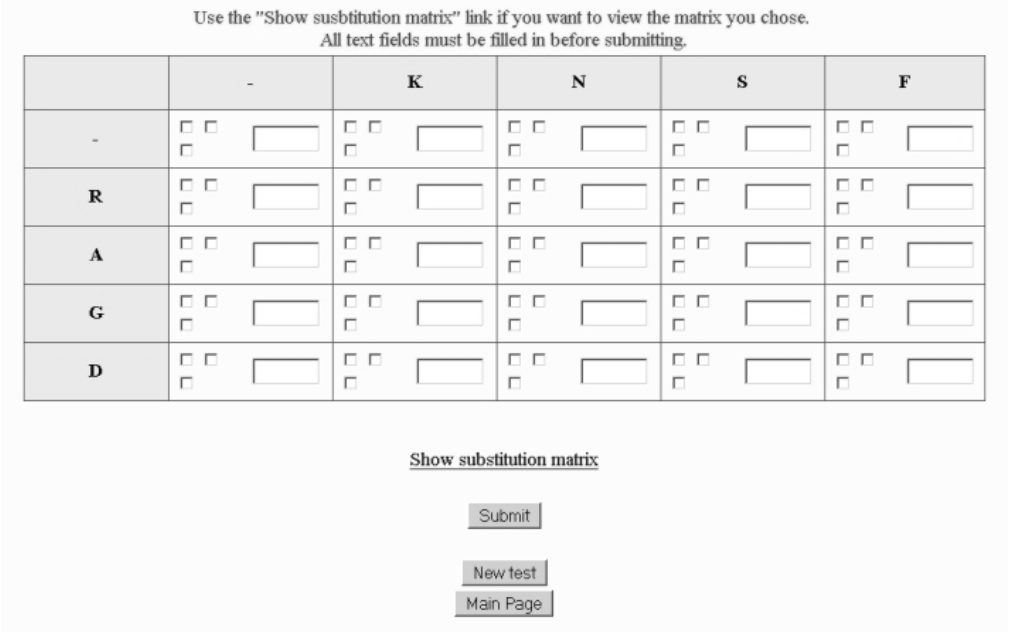
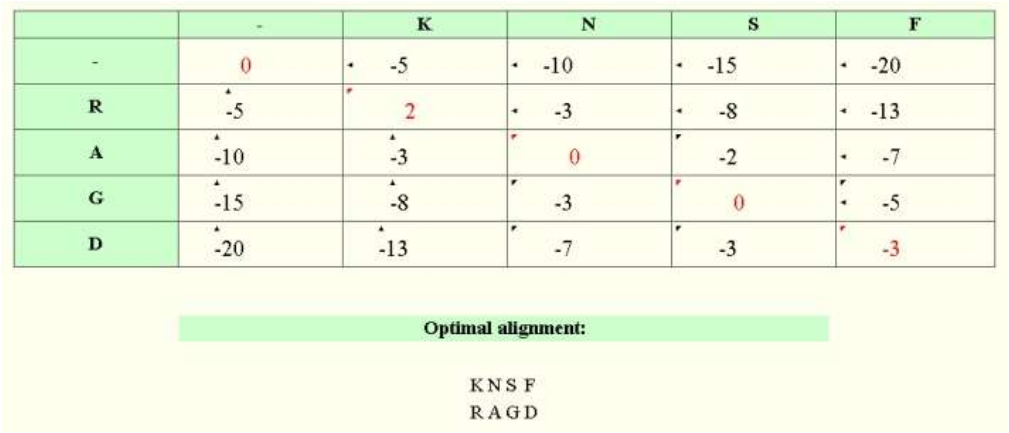


Fig. 5.12: An example of an alignment exercise. This exercise requires the users to complete the score matrix of the alignment of the sequences KNSF and RAGD.

Removing the checkboxes from the cells of the first row and column is another option. This alternative could, however, be even more confusing than using one or three check boxes because it does not correspond to how the algorithm works. At worst, this alternative could give the impression that the pointers of the first row and column are insignificant when a score matrix is computed. Thus, the only alternatives are one or three check boxes.

The reason for choosing three check boxes over one check box for these cells is that three check boxes requires some understanding of the purpose of the pointers before the correct pointer can be set without guessing. With one check box it is possible to set the pointers of these cells without giving the reason for why the pointers are set any thought. Pressing the submit-button once the score matrix is completed produces, given that the submitted score matrix is correct, a page that displays the correct score matrix and the optimal alignment or alignments (fig. 5.13). The optimal path (or paths) through the score matrix is marked with red to make it easier verify the optimal alignment (or alignments).



	-	K	N	S	F
-	0	-5	-10	-15	-20
R	-5	2	-3	-8	-13
A	-10	-3	0	-2	-7
G	-15	-8	-3	0	-5
D	-20	-13	-7	-3	-3

Optimal alignment:

K N S F
R A G D

Fig. 5.13: An example of a completed exercise. Once the users submit a correct score matrix, they are presented with the completed score matrix in which the pointers and the optimal path are included. The optimal alignment(s) is also presented.

Submitting an incorrect score matrix produces an error message that explains what is wrong and where the error is (fig. 5.14), e.g. “The score of cell F(2, 1) is incorrect” or “The direction of cell F(4, 3) is incorrect”

Submitting an incomplete score matrix, i.e. a score matrix in which one or more scores is missing, or an invalid score matrix, i.e. a score matrix that contains a score that is not an integer, produces an error message that states that an invalid score has been entered.

Pressing the new test-button aborts an exercise and opens the exercise configuration page, while pressing the main page-button aborts an exercise and opens the main page of the alignment application.

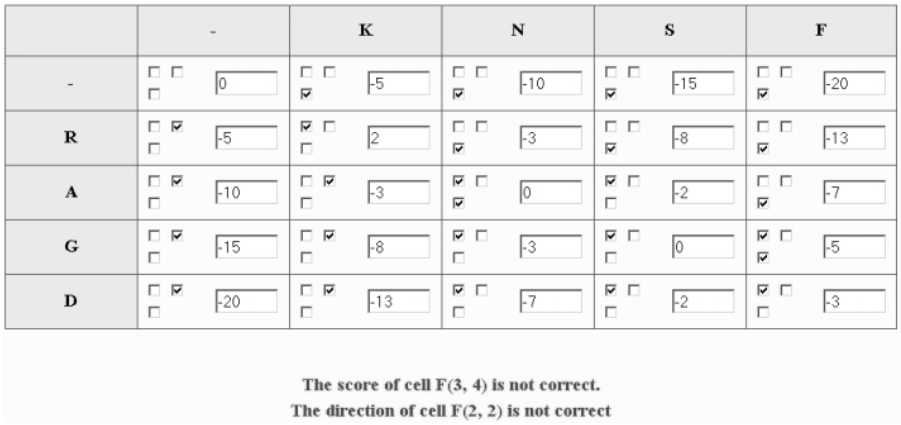


Fig. 5.14: An example of the error messages the users receive id they submit an incorrect score matrix.

5.5 The clustering visualisation

The clustering visualisation is designed to teach how the implemented clustering algorithms cluster gene expression data by providing a graphical presentation of how the different clustering algorithms cluster the genes (i.e. the rows) of a user-defined gene expression data matrix.

The clustering application is started through the link in the introduction page of the clustering application. The interface of the clustering application (fig. 5.15) is composed of a two-dimensional coordinate system, two data tables, a menu, a canvas, and a set of buttons.

The coordinate system has two roles in the clustering visualisation. Firstly, it is where the gene expression data matrix is defined. A row in the gene expression data matrix is defined by placing the mouse pointer within the coordinate system and pressing the left mouse button. Once the left mouse button is pressed within the coordinate system, a numbered point is deposited in the position of the mouse pointer. The deposited point represents a gene, where the number denotes the number of the gene, and the coordinates of the

point represents the gene expression levels of the gene in two samples: the x-coordinate of the point represents the gene expression level of the gene in sample x, while the y-coordinate represents the gene expression level of the gene in sample y. Thus, the gene expression data matrix is defined by depositing a set of points/genes in the coordinate system.

Secondly, the coordinate system is used to present how the algorithms cluster a set of gene expression vectors. Exactly how the coordinate system is used to present a clustering depends on which of the implemented clustering algorithms are chosen to be visualised, and the usage of the coordinate system will therefore be discussed in the sections that describe the visualisation of each of the implemented clustering algorithms.

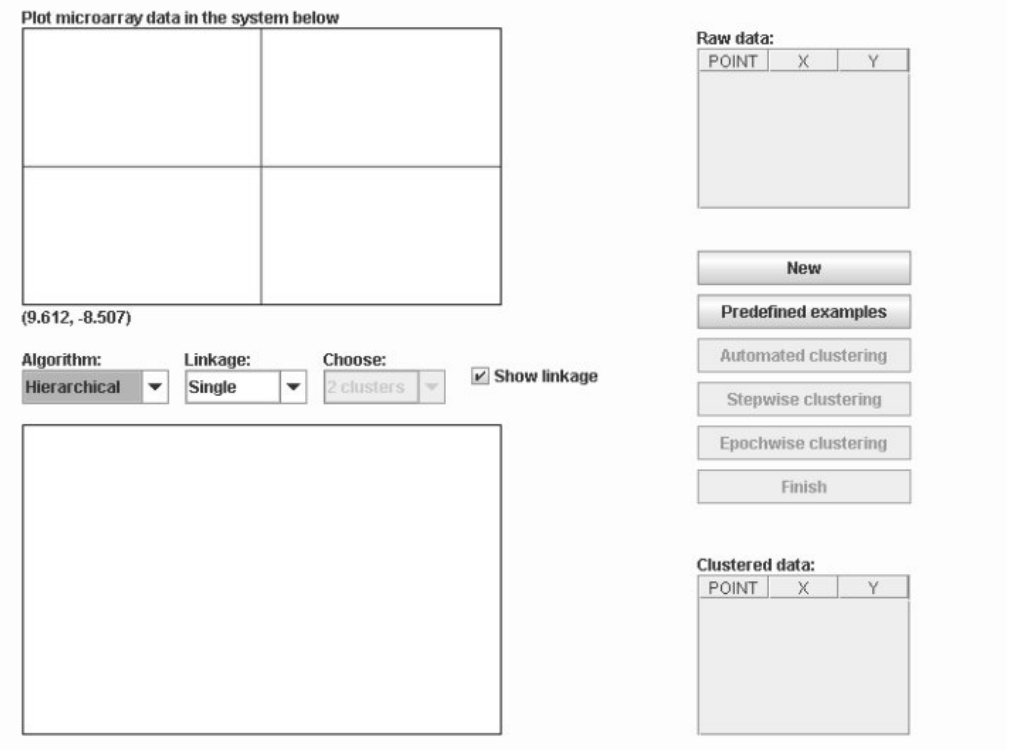


Fig. 5.15: The visualisation interface the users initially are presented with. The coordinate system at the upper left, the canvas at the lower left with the menu in the middle, the raw data table to the upper right, the clustered data table to the lower right with the buttons in the middle.

The two data tables represent the gene expression data matrix whose rows (i.e. genes) the algorithms are to cluster. As the gene expression data matrix is defined by the users, no gene expression data is available when the clustering application is started. Both tables are therefore empty when the application is started. The first table, named the raw data table, represents the gene expression data matrix as it is before it is clustered, and a row is added to this table each time a point is deposited in the coordinate system. The second table, named the clustered data table, represents the gene expression data matrix during and after the clustering process, and therefore remains empty until a visualisation is started. The rows of each of these tables are composed of three columns where the first column displays the number of the gene (i.e. the number of the point in the coordinate system), the second column displays the gene's expression level in sample x (i.e. the x-coordinate of the point), and the third column displays the gene's expression level in sample y (i.e. the y-coordinate of the point). A gene expression data matrix can be presented as a heat map, in which the gene expression levels are represented as coloured squares instead of by the numerical values. The intensity of the colours of these squares corresponds to the expression levels of each gene in the different samples where green intensity signifies negative expression, red intensity positive expression, and no intensity (i.e. black) signifies zero expression. The same colour scheme is used in the second and third column of the data tables to emphasise that the deposited points represent gene expression data, and that the data tables are gene expression data matrices, not merely tables of the coordinates of the deposited points.

It is also possible to define the gene expression data matrix without depositing the points manually. The clustering application is implemented with three predefined sets of genes in which the genes are distributed in 2, 3, and 4 distinct clusters respectively (fig. 5.16). Each cluster is composed of 50 genes each. Pressing the predefined examples-button activates the list from which the examples can be chosen. Selecting an example from this list generates the selected set of genes, and displays the genes both as points in the coordinate system, and as rows in the raw data table (i.e. the gene expression data matrix). The points in the coordinate system are unnumbered because the points are distributed too densely for the numbers to be read.

Since gene expression data normally are n-dimensional, some may argue that a two-dimensional representation of gene expression data might be a too simplified representation to have any educational value. The reason for choosing a two-dimensional representation is simply that the computer screen is two-dimensional, and it is therefore difficult to represent data properly in more than two dimensions. Besides, understanding how multi-dimensional

gene expression data is clustered should not be problematic once clustering of two-dimensional data is understood.

Some may also argue that plotting the gene expression data in a coordinate system may be misleading since gene expression data normally are presented as gene expression data matrices, not as plots. Thus, the correct way of defining a gene expression data matrix would have been to provide an empty gene expression data matrix, in which the gene expression data could have been entered manually. Although this approach would be correct, it might have caused the visualisation to be too cumbersome to be interesting to use.

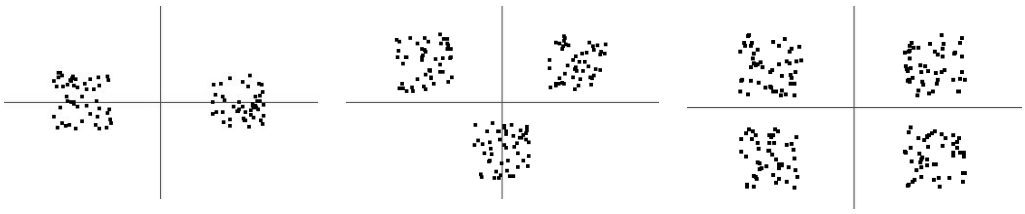


Fig. 5.16: Three examples on the three possible predefined sets of genes. To the left the set of genes is distributed in two clusters, in the middle the set is distributed in tree clusters, and to the right the set is distributed in four clusters.

The clustering application requires at least three genes to be defined in the gene expression data matrix before any visualisation can be started. The first step of starting a clustering visualisation is to choose one of the implemented clustering algorithms from the menu. The menu is composed of two lists, one that contains the algorithms, and one that contains the variants of the algorithms. The algorithm has to be chosen before the variant can be chosen. The next step of starting a visualisation depends on the chosen algorithm. A started visualisation can, at any time, be completed by pressing the finish-button.

Because the defined set of genes remains unchanged until either additional points are plotted in the coordinate system or the new-button is pressed, it is possible to use a set of genes in multiple visualisations. Pressing the new-button resets the clustering interface to the state it was in when the clustering application was started.

5.5.1 The hierarchical clustering visualisation

The hierarchical clustering visualisation demonstrates how the hierarchical clustering algorithm clusters gene expression data. The key to understanding how this algorithm clusters genes is to understand how the dendrogram (i.e. the hierarchical tree) is constructed. Hence, the visualisation focuses on demonstrating how the dendrogram is built. The algorithm that is implemented in the BioTeach system is agglomerative, that is, it starts by treating the set of genes as a set of clusters where each gene is a separate cluster, a singleton cluster, and constructs the dendrogram by grouping two and two clusters together into a larger cluster until all clusters are grouped together in one large cluster. The first pair of clusters that are grouped together in an agglomerative clustering is the pair of singleton clusters (i.e. the pair of genes) that is most similar. The similarity measurement that is used by the BioTeach system is, in all the implemented clustering algorithms, the Euclidean distance between the clusters, that is, the length of a straight line between two clusters. Thus, the Euclidean distance between all the singleton clusters have to be calculated in order to find the pair of clusters that is most similar.

The next step is to group the next pair of clusters that is most similar. The pair of clusters that was grouped together in the first step is now treated as one cluster composed of two singleton clusters, and the distances between the clusters in the set of clusters therefore have to be recalculated.

The distance between a non-singleton cluster (i.e. a cluster composed of two or more singleton clusters) and other clusters, singleton or non-singleton, can, as discussed in chapter 4, be defined in different ways. The options that are implemented by the BioTeach system are single linkage, complete linkage, and average linkage. Single linkage defines the distance between a non-singleton cluster and other clusters as the distance between the pair of singleton clusters in the non-singleton cluster and the other cluster that is nearest to each other. If the other cluster is a singleton cluster, the distance is defined as the distance between the singleton cluster in the non-singleton cluster that is nearest the other singleton cluster. Complete linkage defines the distance as the distance between pair of singleton clusters in the non-singleton cluster and the other cluster that is furthest apart. Average linkage defines the distance as the average distance between all the singleton clusters in the non-singleton cluster and the other cluster.

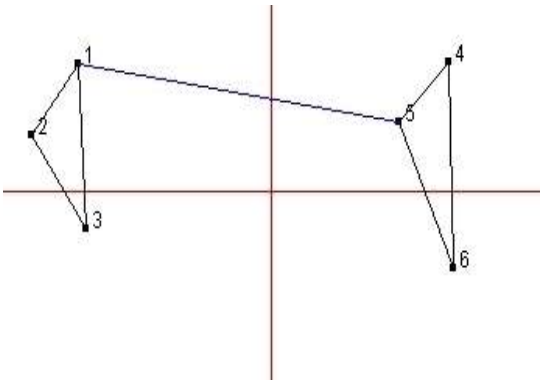
The hierarchical clustering continues to group the most similar pair of clusters until all clusters have been grouped together in one single cluster.

A hierarchical clustering is started by choosing hierarchical clustering and linkage from the lists in the menu, and pressing either the automated clustering-button, or the stepwise clustering-button. The automated clustering starts a fully automated presentation of the hierarchical clustering algorithm, while the stepwise clustering allows the progress of the visualisation to be controlled by the users.

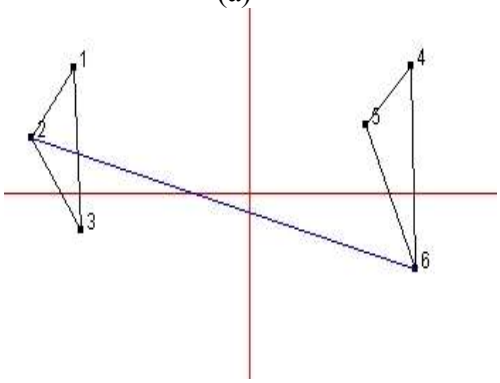
It is possible to choose whether or not the visualisation is to display the singleton clusters that define the least distance in each step of the visualisation by ticking off in the show-linkage box. A visualisation with this option enabled starts each step in the clustering with the drawing of a blue line in the coordinate system between the two singleton clusters that define the least distance according to the chosen linkage (fig. 5.17). The blue line is temporary, and is only displayed for one second before it is removed. Since linkage is not used by any of the other implemented algorithm, the show-linkage box is deactivated in the visualisations of other implemented algorithms. The canvas is also only used by the hierarchical clustering visualisation, so the canvas is removed from the visualisation interface if any of the other algorithms is selected.

The automated and the stepwise visualisation start with the grouping of the two singleton clusters that are closest to each other. This step is visualised (fig. 5.18) by drawing a line between the pair of points in the coordinate system that is closest to each other. The canvas is simultaneously updated to show the grouped pair of clusters as a dendrogram with two leaf nodes, while the remaining ungrouped clusters are presented as points in the canvas. The clustered data table is also updated to display the genes in the order they are grouped, that is, the ungrouped genes are displayed first, while the grouped pair is found in the last two rows of the table.

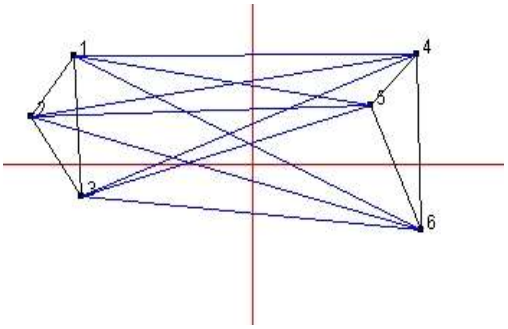
In an automated clustering visualisation the next step in the clustering is presented after a pause of one second, while in the stepwise clustering visualisation the next step is presented after the next-button has been pressed. The next step can be visualised in two different ways depending on whether the next pair of clusters to be grouped is a pair of singleton clusters, or a grouping of the non-singleton cluster from the previous step and a singleton cluster. In the first case (fig. 5.19), a line is drawn in the coordinate system between the two singleton clusters that are nearest each other. The canvas displays, in this case, two separate dendrograms of two leaf nodes each, while the order of the genes in the clustered data table is rearranged so that the grouped pair of genes is found in the two last rows.



(a)



(b)



(c)

Fig 5.17: The visualisation of (a) single linkage in which the two object closest to each other define the least distance, (b) complete linkage in which the two objects furthest apart define the least distance, and (c) average linkage in which the average distance between all objects define the least distance.

In the second case (fig. 5.20), a line is drawn between the singleton cluster and each of the singleton clusters in the pair of clusters that was grouped in the first step of the visualisation. Thus, the three singleton clusters are connected to each other, and the lines forms a triangle that signifies that the three singleton clusters belong to the same larger cluster. In this case, a branch is added to the dendrogram that was drawn on the canvas in the first step so thatthe tree becomes a tree composed of three leaf nodes. The genes of the clustered data table are also rearranged so that the three grouped genes are found in the three last rows of the table.

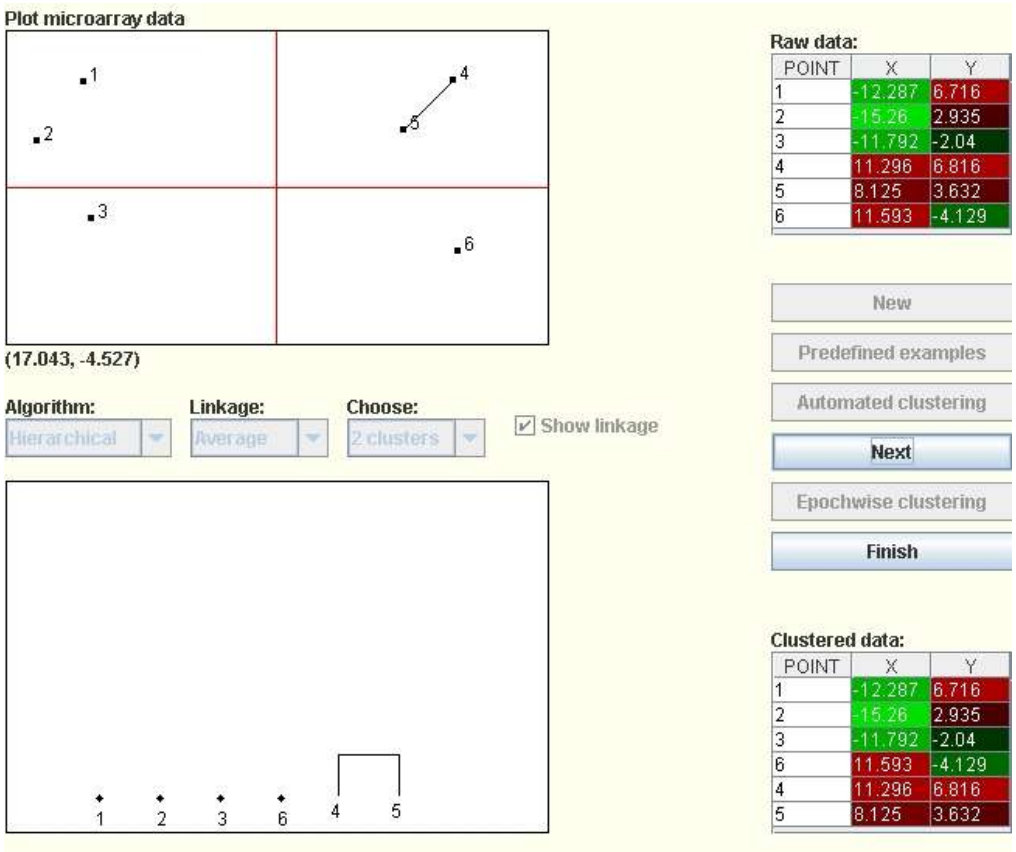


Fig 5.18: The visualisation of the first step of a hierarchical clustering. The coordinate system shows that gene 4 and 5 are most similar, and are therefore clustered in the dendrogram. The clustered data table has also been updated to list gene 4 and 5 in the two last rows.

The remaining steps in the clustering can be visualised in several different ways depending on if the most similar pair of clusters are a pair of singleton clusters, a non-singleton cluster and a singleton cluster, or a pair of non-singleton clusters.

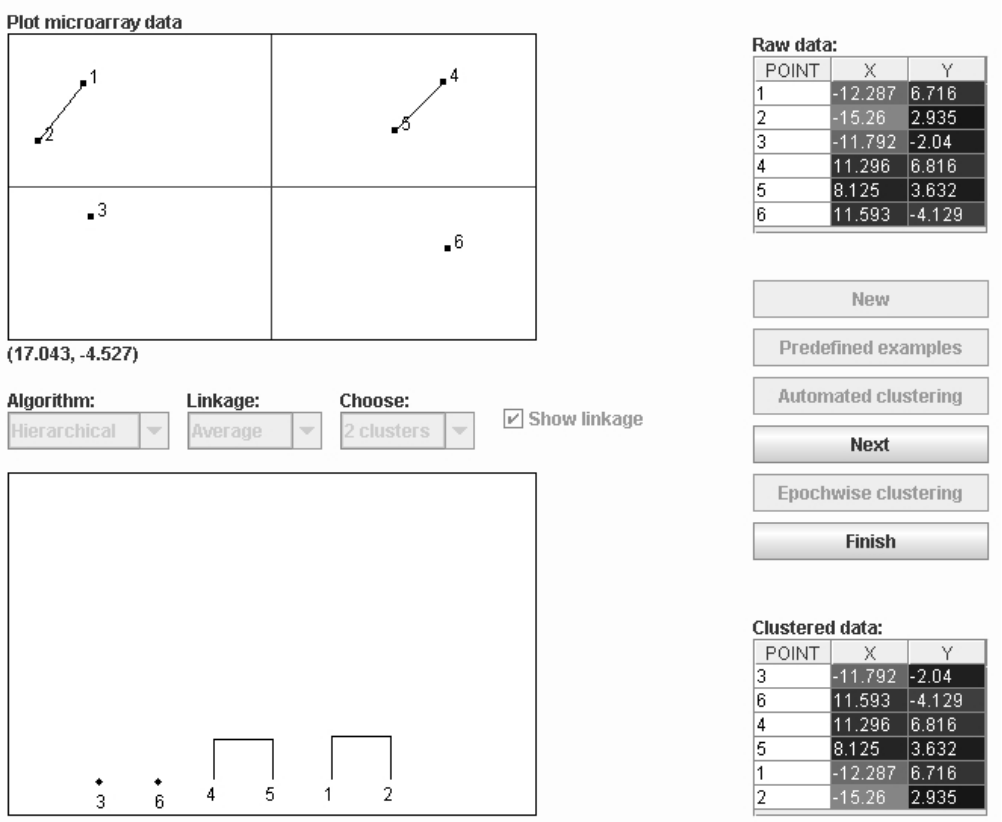


Fig. 5.19: The second step in the visualisation in which genes 4 and 5, and 1 and 2 have been clustered.

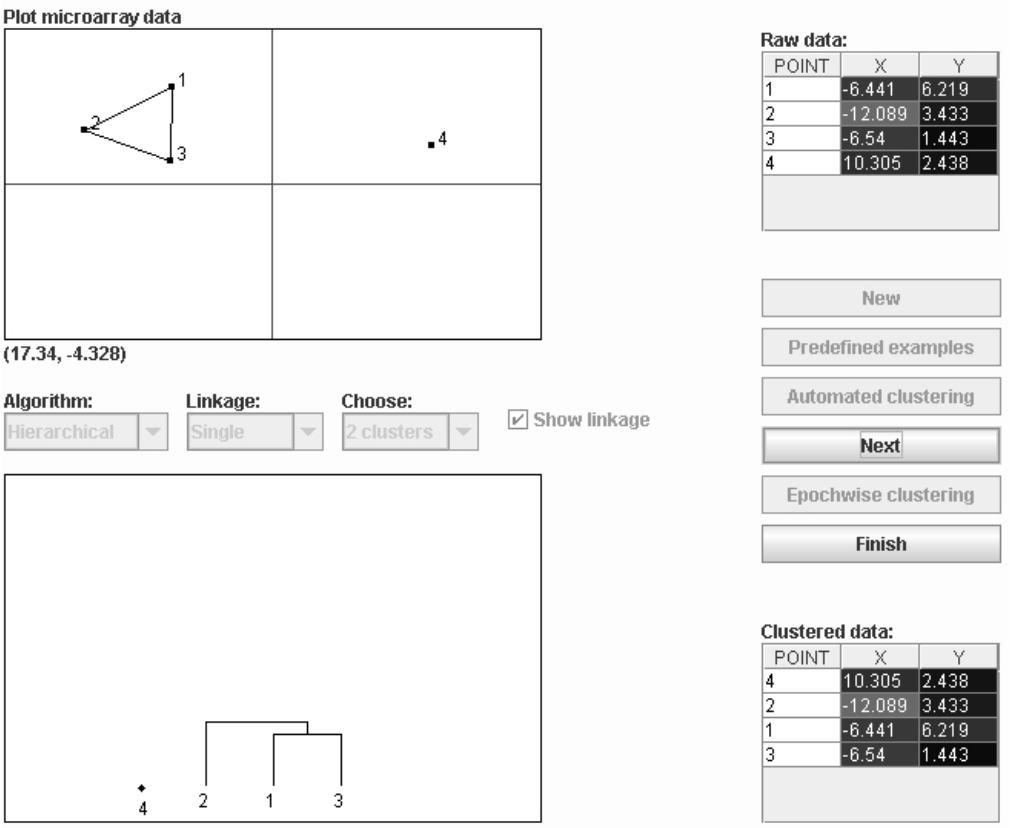


Fig. 5.20: A second step in the visualisation where genes 1, 2, and 3 have been clustered.

The steps that involve the grouping of two singleton clusters are visualised in the same way as the first step (fig. 5.17), while the steps that involves grouping a non-singleton cluster with a singleton cluster, or grouping a pair of non-singleton clusters can be visualised in different ways.

A step that involves grouping a non-singleton cluster of two singleton clusters with another singleton cluster is visualised in the same way as the second possibility of the second step (fig. 5.20).

A step that involves grouping a non-singleton cluster composed of three singleton clusters with another singleton cluster, or grouping two non-singleton clusters composed of two singleton clusters each, is visualised by treating the four singleton clusters as the four corners of a quadrangle, and by

drawing the quadrangle in the coordinate system to signify that the four singleton clusters belongs to the same larger cluster (fig. 5.21).

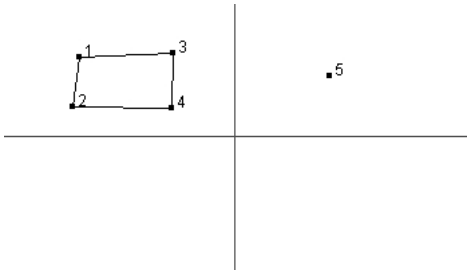


Fig. 5.21: The quadrangle. Signifies that the four genes/clusters belong to the same, larger, cluster.

The dendrogram that the canvas now displays is, in the first case, a dendrogram in which a fourth branch has been added to the tree of the non-singleton cluster (fig. 5.22).

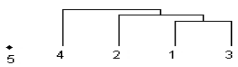


Fig. 5.22: A dendrogram in which gene 4 has been clustered with an earlier cluster of genes 1, 2, and 3.

In the second case, the dendrogram that the canvas displays is composed of the two dendrograms of the non-singleton clusters linked together (fig. 5.23).

A step that involves grouping a non-singleton cluster composed of four or more singleton clusters with another singleton cluster, or grouping a pair of non-singleton clusters that has five or more singleton clusters between them, is visualised by drawing the convex hull of the set of singleton clusters that composes the grouped pair of clusters. This is to signify that the singleton clusters that compose the grouped pair of clusters belong to the same larger cluster.

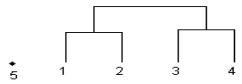


Fig. 5.23: A dendrogram in which two clusters separate clusters of two genes, 3 and 4, and 1 and 2, have been clustered.

The first scenario produces a dendrogram that is similar to that of fig. 5.22, only with 5 or more branches instead of 4. The second scenario produces a dendrogram similar to that of fig. 2.23, except that the linked dendrograms in this case have 5 or more branches between them instead of 4.

A completed visualisation of the hierarchical clustering algorithm (fig. 5.24) presents a coordinate in which the convex hull of the complete set of genes is drawn. This is to signify that the complete set of genes have been grouped together in one large cluster. The accompanying dendrogram is presented as a single tree in which each gene is a leaf node, while the clustered data table presents the genes in the order that they are clustered.

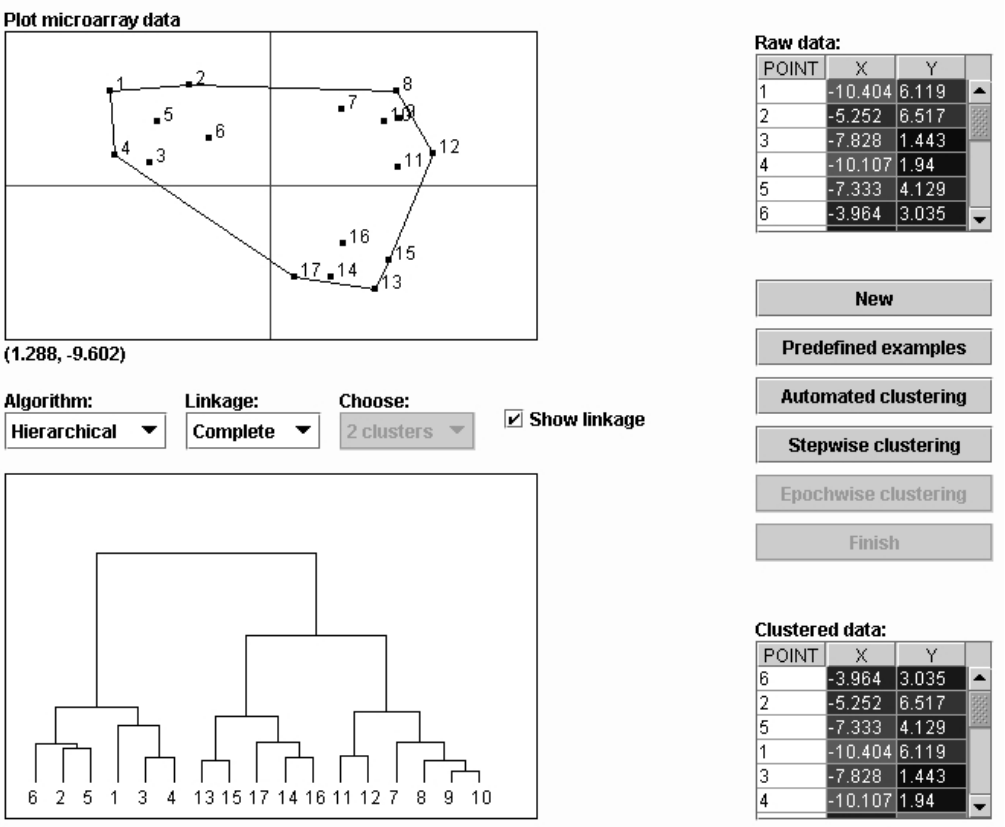


Fig. 5.24: A completed visualisation with the convex hull of the whole set of genes at the top, the completed dendrogram at the bottom, and the genes in the clustered order in the clustered data table at the lower right.

5.5.3 The k-means visualisation

The k-means visualisation demonstrates how the batch variant and the online variant of the k-means algorithm cluster a set of genes. The k-means algorithm clusters a set of genes by placing n centroids in the set of genes, and recalculating their position until they are located in the n cluster centres. Thus, the number of centroids predefines how many clusters there are in a set of genes. The number of centroids therefore has to be chosen with care. Normally it is the users who define the number of the centroids, while it is the implementation of the algorithm that determines their initial position. However, as the initial positions of the centroids affect the results of the algorithm, the number of clusters and their initial positions are user-defined in the clustering application to allow for as much experimentation with the algorithm parameters as possible.

When the algorithm is finished, that is, when all centroids have been repositioned at the cluster centres, all genes that are assigned (i.e. nearest) to a centroid belong to the same cluster. A k-means visualisation is started by choosing the k-means algorithm and either the batch variant or the online variant from the lists in the menu.

The first step in the k-means visualisation is to place the centroids in the coordinate area. The centroids are placed in the same way as the genes are plotted. However, in order to place centroids and not more genes, the place centroids-button has to be pressed. Attempting place centroids in the coordinate area without pressing this button, will only add more genes to the gene set. Once the place centroids-button is pressed, an indefinite number of centroids can be placed in the coordinate system. Each centroid is represented in the coordinate system as a coloured point labelled C1, C2, C3, and so forth (fig. 5.25). A k-means clustering visualisation can be started once at least two centroids have been placed. As with the hierarchical clustering visualisation, it is possible to choose between an automated visualisation and a stepwise visualisation. The progress of the stepwise visualisation is controlled with the next-button, while the automated visualisation presents each step in the clustering after a delay of one second.

Independently of whether the automated visualisation or the stepwise visualisation is chosen, the batch variant of the k-means visualisation starts by iterating over the set of genes and assigning each gene to its nearest centroid. The points that represent the genes are simultaneously coloured with the same

colour as the nearest centroid to make it easier to discern which genes belongs to which centroid.

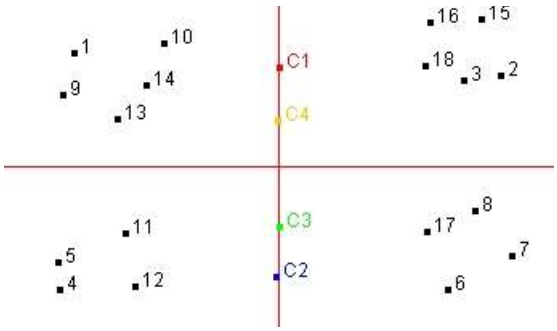


Fig 5.25: The start of visualisation of a k-means clustering with four centroids.

The next step is to reposition the centroids. The goal is to place each centroid in a position where the distance from the centroid to each of its assigned genes is equal. This position is found by calculating the average of the positions of the genes that are assigned to each centroid (fig. 5.26).

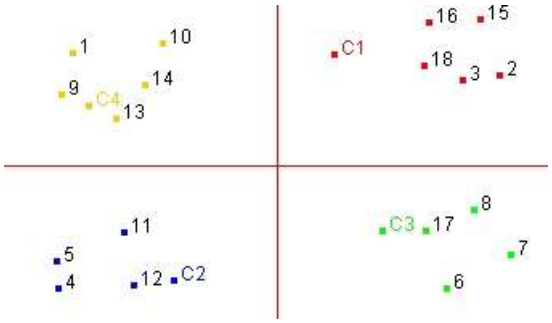


Fig 5.26: The second step of a k-means visualisation in which the centroids have been relocated for the second time.

Since the clusters have been repositioned it is necessary to control whether any of the genes are nearer to a different centroid than the one they currently are assigned to. The first step is therefore repeated to reassign each gene to its closest centroid. The algorithm is considered to be finished if none of the genes are reassigned. Otherwise, it is necessary to reposition the centroids again; those centroids that loose genes or receive genes are no longer positioned such that the distances from the centroids to their assigned genes are equal. The positions of these centroids are therefore recalculated, and the centroids repositioned.

The process of reassigning genes and repositioning the centroids is repeated until no genes are reassigned, and, hence, no centroids have to be repositioned. The clustered data table remains empty during this process.

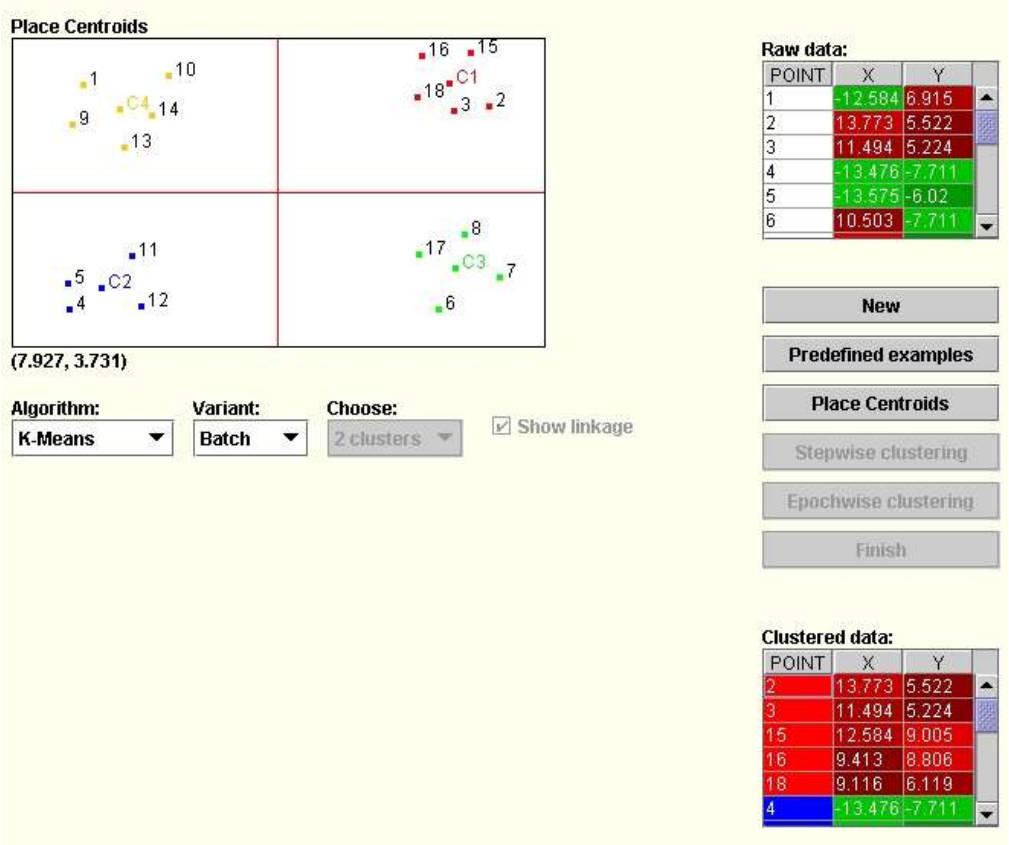


Fig 5.27: A completed k-means visualisation. The centroids have been relocated at the four cluster centres, and the clustered data table shows that genes 2, 3, 15, 16, and 18 are clustered together around C1, the red centroid.

Once the process is finished, the clustered data table is rearranged so that the order corresponds to the order in which the genes are clustered. The first column in each row is also coloured with the same colour as the centroid the gene belongs to (fig. 5.27).

The online variant uses a different approach to position the centroids at the cluster centres. Instead of assigning each gene to the nearest centroid and then repositioning the centroids, the online variant iterates over the set of genes and assigns one gene at a time to the nearest centroid. At the same time, each gene moves the nearest centroid closer to itself. In the BioTeach implementation of this k-means variant, each gene moves the nearest centroid 1% of the distance between the gene and the centroid closer to itself. Thus, the centroids appear to be gliding towards the cluster centres.

One of the challenges with the online variant of the k-means algorithm is to find a good criterion for when the algorithm is finished. Using the termination criterion of the batch variant would cause the algorithm to terminate prematurely; the centroids only move a slight distance every time they are moved, and the set of genes that is assigned to each centroid could therefore be the same over a number of iterations. The termination criterion of the batch variant could therefore regard the algorithm as finished as early as after the first iteration. The termination criterion that is implemented in the online variant of the k-means algorithm stops the visualisation when the set of genes that is assigned to each centroid is unchanged

a) the centroids have moved less than 0.4 pixels in each of the five last movements

b) the average distance of the last five movements is less than 0.2 pixels

These criterions prevent the algorithm from terminating too early when the genes are distributed in well defined, but small, clusters. At the same time, these criterions prevent the algorithm from terminating when the genes are distributed in clusters that are large in circumference, or not well defined. In the latter case, the visualisation has to be terminated manually by pressing the finish-button.

The purpose of using these criterions is to illustrate that the online variant is not as straightforward to use or implement as the batch variant, and that the results of the online k-means variant may depend on how the termination criterion is defined.

5.5.5 The self organizing maps visualisation

The self organizing maps visualisation demonstrates how self organising maps (SOM) cluster genes. SOMs cluster a set of genes by stretching a grid of units (e.g. a line, an array, a cube, a parallelepiped) to fit the gene set. The grid is fitted to the set of genes in such a way that the relationship between the units corresponds to the relationships in the set of genes. Thus, the grid becomes a map over the set of genes. To achieve this, the units of the grid is organised in neighbourhoods, and when a unit is moved, the unit pulls the neighbouring units with it.

The BioTeach implementation of SOMs allows the users to choose between a line grid and an array grid. The line grid is composed of ⅓ as many units as there are genes in the set of genes. Each unit in the line grid have a neighbour in front and one behind, except the first, which only has one neighbour in front, and the last, which only has one neighbour behind. The array grid is initially composed of ⅓ as many units as there are genes in the gene set, but the number is adjusted so that the array always is a full rectangle or square.

Each unit in the array grid, except the corner units and the units along the side of the square or rectangle, has four neighbours, the unit above, below, to the left, and to the right. The corner units only have two neighbours, the unit below or above, and the unit to the left or to the right. The units along the horizontal edges have three neighbours each: the unit above or below, and the units to the left and the right. The units along the vertical edges also have three neighbouring units each: the unit above and below, and the unit to the left or to the right.

During the stretching process, each gene moves the closest unit 5% of the distance between the gene and the unit, while the neighbours of a unit are moved 2% of that distance.

A visualisation of how SOMs work is started by selecting SOM and either the line option, or the grid option (i.e. the array grid) from the lists in the menu, and then pressing the automated clustering-button, the stepwise clustering-button, or the epochwise clustering button. The epochwise clustering presents the visualisation in epochs, that is, a full iteration over the set of genes is performed before any of the units are moved. In comparison, the stepwise clustering moves a unit at each step in the iteration. It is not recommended to select the stepwise clustering because the visualisation is implemented to do 3000 full iterations before it is finished (if there are 50 genes in the dataset, the

next-button has to be pressed 50 x 3000 = 15000 times to complete the visualisation.). Thus, the epochwise visualisation is the better alternative if the users want to step through the visualisation.

The visualisation of both the line grid variant and the array grid variant starts with the units of the grid distributed randomly around the origin of the coordinate system (fig. 5.28).

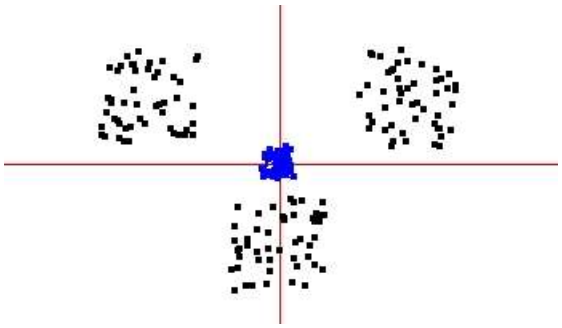


Fig. 5.28: The beginning of a SOM visualisation in which the units of the array grid has been distributed around the origin of the coordinate system.

The visualisation proceeds to iterate over the set of genes and stretch the grid to gradually fit the set of genes (fig. 5.29).

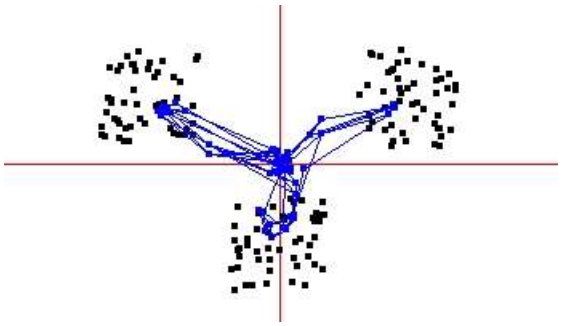


Fig. 5.29: The array grid during the stretching process.

After 3000 iterations over the set of genes, the visualisation is completed and the grid has been fitted to the set of genes (fig. 5.30).

The SOMs work best on large datasets (e.g. thousands of genes), and since the coordinate system of the BioTeach system is too small to accommodate such a number of genes, it is difficult to provide an exact visualisation of how SOMs are constructed. The best results are produced by a visualisation that uses the predefined example with the genes distributed in four clusters. Although the visualisation is not perfect, it should provide a presentation that is good enough to enable the users to understand how a SOM is constructed.

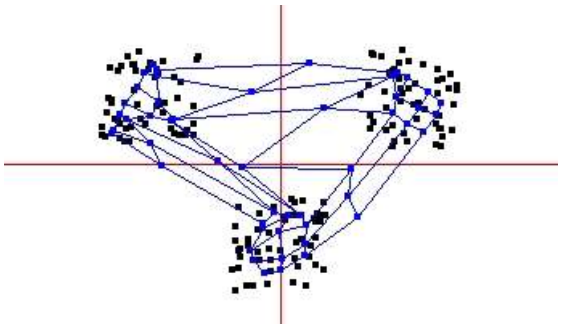


Fig. 5.30: The array grid after the stretching process is completed.

Chapter 6

Hypothetical exercise examples

6.1 Introduction

Although web-technology offers possibilities for designing exercises that comment the answers that are given, there are obvious limitations to such exercises. Firstly, as it is difficult to design computer programs which are able to assess the quality of answers that are based on creative thinking, such exercises should have a definite answer. Exercises that require discussion and creative thinking could, of course, be included in a web-based learning environment, but the answers to such exercises should be forwarded to the lecturer, or to his or hers assistants, for proper evaluation instead of being evaluated by a computer program. Secondly, there should be a limited number of possible incorrect answers to such exercises. The reason is more or less the same as with exercises that has an indefinite answer; it is difficult to anticipate all possible answers, and therefore difficult to provide appropriate comments.

One approach that meets these limitations is multiple choice exercises. The Needleman-Wunsch exercise discussed in the previous chapter is another possible approach, and a third possible approach is the Sourcer's Apprentice approach discussed in chapter 2.

This chapter discusses how multiple choice exercises and the Sourcer's Apprentice approach could be used in a web-based learning environment for bioinformatics.

6.3 A Blast example

The Blast algorithm (Higgs & Attwood, 2005) is a popular tool for searching sequence databases (i.e. databases of gene sequences), and it is a tool that students of bioinformatics are required to master. The algorithm uses a form of pairwise sequence alignment to search a database for gene sequences that are similar to a query sequence. An important part of mastering the Blast search tool is to interpret the results the search produce.

A typical Blast result (fig. 6.2) lists the similar sequences sorted by similarity in descending order. Each row in the list represents a gene sequence, and each row is composed of four columns. The first column provide information about the different database ids of a sequence (a gene sequence can be stored in multiple databases) and the name of the gene sequence, the second column provides a short description of the gene sequence, the third column displays the score of aligning a sequence with the query sequence, and the fourth displays the e-value of the alignment. The score and the e-value are used to determine whether or not a sequence is similar to the query sequence. A high score and a low e-value (e.g. $e < 0.01$) indicate that a sequence is similar to the query sequence. An exercise that aims at teaching how to interpret Blast results could be designed like the Sourcer's apprentice (see fig. 6.3 for a sketch).

Sequences producing significant alignments:	Score (Bits)	E Value
gi|1705762|sp|P13569|CFTR_HUMAN Cystic fibrosis transmembrane...	2804	0.0
gi|62510444|sp|Q7JII8|CFTR_MACFA Cystic fibrosis transmembran...	2755	0.0
gi|2506121|sp|Q00555|CFTR_SHEEP Cystic fibrosis transmembrane...	2549	0.0
gi|1705763|sp|Q00554|CFTR_RABIT Cystic fibrosis transmembrane...	2532	0.0
gi|461719|sp|P35071|CFTR_BOVIN Cystic fibrosis transmembrane ...	2531	0.0
gi|20141218|sp|P26361|CFTR_MOUSE Cystic fibrosis transmembran...	2231	0.0
gi|116142|sp|P26363|CFTR_XENLA Cystic fibrosis transmembrane ...	2163	0.0
gi|116141|sp|P26362|CFTR_SQUAC Cystic fibrosis transmembrane ...	2001	0.0
gi|21431744|sp||P34158_2 [Segment 2 of 2] Cystic fibrosis tra...	755	0.0
gi|50402226|sp|Q8VI47|MRP2_MOUSE Canalicular multispecific or...	469	2e-137
gi|3219024|sp|Q63120|MRP2_RAT Canalicular multispecific organ...	464	6e-136
gi|8928546|sp|O15439|MRP4_HUMAN Multidrug resistance-associat...	399	4e-110
gi|33112350|sp|P91660|L259_DROME Probable multidrug resistanc...	331	1e-89
gi|461721|sp|Q00553|CFTR_MACMU Cystic fibrosis transmembrane ...	323	2e-87
gi|31340495|sp|Q9P5ND|YM85_SCHPO Probable ATP-dependent permease	293	3e-78
gi|172325S|sp|Q10185|YAWB_SCHPO Probable ATP-dependent permease	287	2e-76

Fig. 6.2: The 15 best matches for a blast search for the protein sequence of the gene CFTR_HUMAN in the Swiss-Prot database.

One way of designing a Blast exercise based on the Sourcer's Apprentice could be to replace the different historical texts with different Blast results in which the column headers have been removed. The exercise could then be to identify specific information about a selection of the sequences that are listed in each Blast result. Each sequence in the selection should have a note card in which the information are to be entered. The specific information to be identified for each sequence could include

- the different database ids
- the name of the gene the sequence represent
- the score of aligning the sequence with the query sequence
- the e-value of the alignment

The exercise should be implemented with a drag and drop system similar to the bucket system of the Sourcer's Apprentice because it limits the possible wrong answers, and, thus, makes it possible to provide comments if incorrect information is attempted to be entered into the note cards (allowing information to be typed into the note cards manually would allow information that is impossible to comment to be entered into the note cards). If the information suggested above was to be identified, there should be separate

buckets for each of the possible database ids in the selection (a sequence can be stored in different databases), for the name of the gene, the description of the gene, the alignment score, and the e-value.

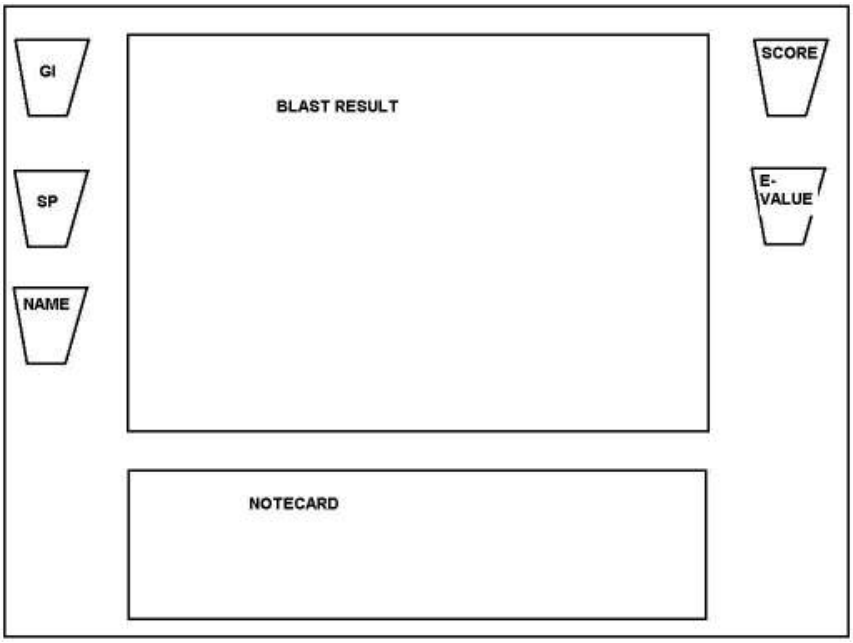


Fig. 6.3: A sketch of a Blast exercise based on the Sourcer's Apprentice system. The Blast results can be presented in the upper frame, while the structured notecards for each set of results can be presented at the bottom. This example would require that the gi-id and the sp-id for a selection of genes are found along with the name of the genes, the score and the e-value.

The exercise should also require the scores and e-values of a selection of the sequences to be discussed in order to ensure that the significance of the scores and e-values has been understood. It would, as discussed above, be difficult to design a computer program that could evaluate these discussions, and the discussions should therefore be sent to the lecturer, his (or hers) assistants for evaluation. These discussions could also, for instance, be included in a mandatory exercise.

An exercise like this should be accompanied by the information necessary to understand the algorithm and the results. Otherwise it would be necessary to consult textbooks or other web sites to understand how to complete an exercise. The exercise should also be preceded by an introduction that explains the features of the application as well as the purpose of the exercise.

6.4 Database and portal exercises

Mastery of web-based biological databases and portals are an essential part of the bioinformatics curriculum. There are different databases for different types of biological information, and there are different portals that provide access to these databases. The different portals also have different ways of presenting database entries (e.g. fig 6.4), and they provide access to a varying selection of the available databases. Mastery of these tools requires knowledge about the information the different databases provides, which portals to use to gain access to them, and how to read the database entry presentations of the different portals.

Mastery of the databases and the portals also requires knowledge about the techniques that are used to find information in the databases, i.e. database searching. The various portals provide search engines that allow the different databases to be queried. Some portals also allow multiple databases to be queried simultaneously. The different portals do, however, have different ways of presenting the search results (e.g. fig 6.5). Thus, it is necessary to know how to read these presentations in order to find the information that the databases was queried for.

The Sourcer's Apprentice approach could be used to design an exercise that teaches the use of the different portals and databases, but it would require the different features and presentation forms of the different databases and portals to be integrated into a realistic interface. A less complicated approach would be to exploit that the portals that provide access to the different databases are web-based.

☐ 1: NM_000660. Reports Homo sapiens tran...[gi:63025221]

```
LOCUS       NM_000660               2346 bp    mRNA    linear   PRI 04-JUL-2005
DEFINITION  Homo sapiens transforming growth factor, beta 1 (Camurati-Engelmann
            disease) (TGFB1), mRNA.
ACCESSION   NM_000660
VERSION     NM_000660.3  GI:63025221
KEYWORDS    .
SOURCE      Homo sapiens (human)
  ORGANISM  Homo sapiens
            Eukaryota; Metazoa; Chordata; Craniata; Vertebrata; Euteleostomi;
            Mammalia; Eutheria; Euarchontoglires; Primates; Catarrhini;
            Hominidae; Homo.
REFERENCE   1  (bases 1 to 2346)
  AUTHORS   Zhu,C., Ying,D., Zhou,D., Mi,J., Zhang,W., Chang,Q. and Li,L.
  TITLE     Expression of TGF-beta1 in smooth muscle cells regulates
            endothelial progenitor cells migration and differentiation
  JOURNAL   J. Surg. Res. 125 (2), 151-156 (2005)
   PUBMED   15854667
  REMARK    GeneRIF: TGFbeta1 transfected smooth muscle cells conditional
            medium produced significantly more endothelial colonies (P<0.05).
            The adhesion force between endothelial progenitor cells and smooth
            muscle cells in TGFbeta1 group was higher than control.
REFERENCE   2  (bases 1 to 2346)
  AUTHORS   Cross,N.A., Chandrasekharan,S., Jokonya,N., Fowles,A., Hamdy,F.C.,
            Buttle,D.J. and Eaton,C.L.
  TITLE     The expression and regulation of ADAMTS-1, -4, -5, -9, and -15, and
            TIMP-3 by TGFbeta1 in prostate cells: relevance to the accumulation
            of versican
  JOURNAL   Prostate 63 (3), 269-275 (2005)
   PUBMED   15599946
  REMARK    GeneRIF: negative effect of TGFbeta1 on ADAMTS-1, -5, -9, and -15
            coupled with increases in their inhibitor, TIMP-3 may aid the
            accumulation of versican in the stromal compartment of the prostate
            in BPH and prostate cancer
```

(a)

UniProtKB/Swiss-Prot entry P13569

Printer-friendly view | Submit update | Quick BlastP search

[Entry info] [Name and origin] [References] [Comments] [Cross-references] [Keywords] [Features] [Sequence] [Tools]

Note: most headings are clickable, even if they don't appear as links. They link to the user manual or other documents.

Entry information	
Entry name	**CFTR_HUMAN**
Primary accession number	**P13569**
Secondary accession numbers	None
Entered in Swiss-Prot in	Release 13, January 1990
Sequence was last modified in	Release 34, October 1996
Annotations were last modified in	Release 48, September 2005
Name and origin of the protein	
Protein name	**Cystic fibrosis transmembrane conductance regulator**
Synonyms	**CFTR** **cAMP-dependent chloride channel**
Gene name	**Name: CFTR** *Synonyms:* ABCC7
From	Homo sapiens (Human) [TaxID: 9606]
Taxonomy	Eukaryota, Metazoa, Chordata, Craniata, Vertebrata, Euteleostomi, Mammalia, Eutheria, Euarchontoglires, Primates, Catarrhini, Hominidae, Homo

References

[1] NUCLEOTIDE SEQUENCE.
PubMed=2475911 [NCBI, ExPASy, EBI, Israel, Japan]
Riordan J.R., Rommens J.M., Kerem B., Alon N., Rozmahel R., Grzelczak Z., Zielenski J., Lok S., Plavsic N., Chou J.-L., Drumm M.L., Iannuzzi M.C., Collins F.S., Tsui L.-C.;
"Identification of the cystic fibrosis gene: cloning and characterization of complementary DNA.";
Science 245:1066-1073(1989).

[2] NUCLEOTIDE SEQUENCE.
PubMed=1710598 [NCBI, ExPASy, EBI, Israel, Japan]
Zielenski J., Rozmahel R., Bozon D., Kerem B., Grzelczak Z., Riordan J.R., Rommens J., Tsui L.-C.;
"Genomic DNA sequence of the cystic fibrosis transmembrane conductance regulator (CFTR) gene.";
Genomics 10:214-228(1991).

[3] PHOSPHORYLATION SITES.
PubMed=1377674 [NCBI, ExPASy, EBI, Israel, Japan]
Picciotto M.R., Cohn J.A., Bertuzzi G., Greengard P., Nairn A.C.;
"Phosphorylation of the cystic fibrosis transmembrane conductance regulator.";
J. Biol. Chem. 267:12742-12752(1992).

(b)

Fig. 6.4: (a) a database entry of the nucleotide sequence database of Entrez. (b) a database entry of the nucleotide sequence database of Expasy

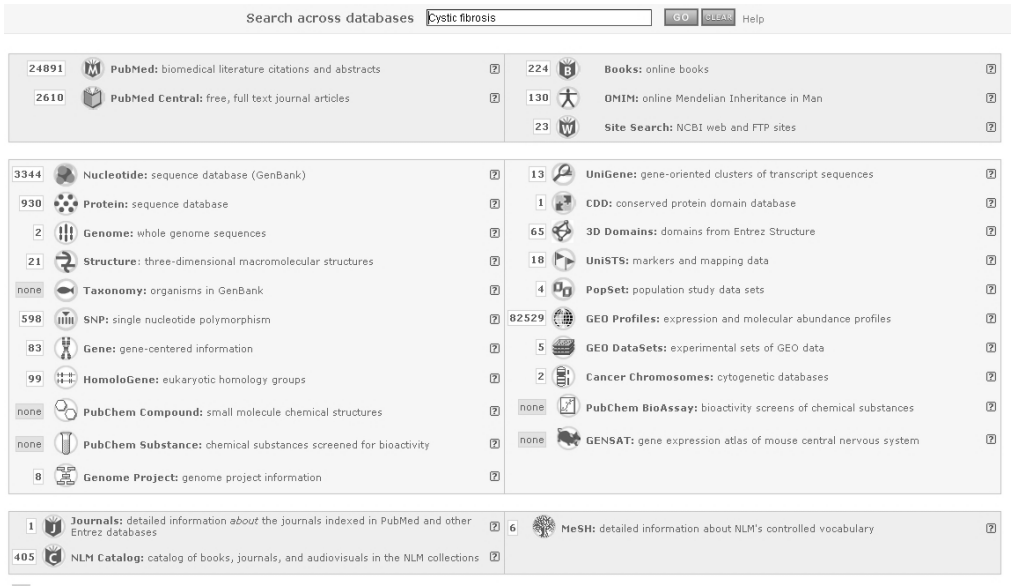
One way of creating such an application is through the use of multiple choice exercises that ask questions requiring that specific pieces of information are found in the databases. Hence, it would be necessary to use the portals and databases to answer the questions correctly. One of the challenges with using multiple choice exercises is to provide answer alternatives to the questions in which the correct answer does not stand out. The correct answer should not, however, be too difficult to discern from the incorrect answers in order to avoid confusion and frustration. This could be achieved by including common misconceptions or information that is found alongside the requested information as incorrect alternatives. Thus, the alternatives of a multiple choice exercise become tools for controlling that the usage of the portals and the databases has been understood. This approach would also make it possible to use comments on incorrect answers to explain common misconceptions and provide guidance towards the correct answer.

6.4.1 A hypothetical example

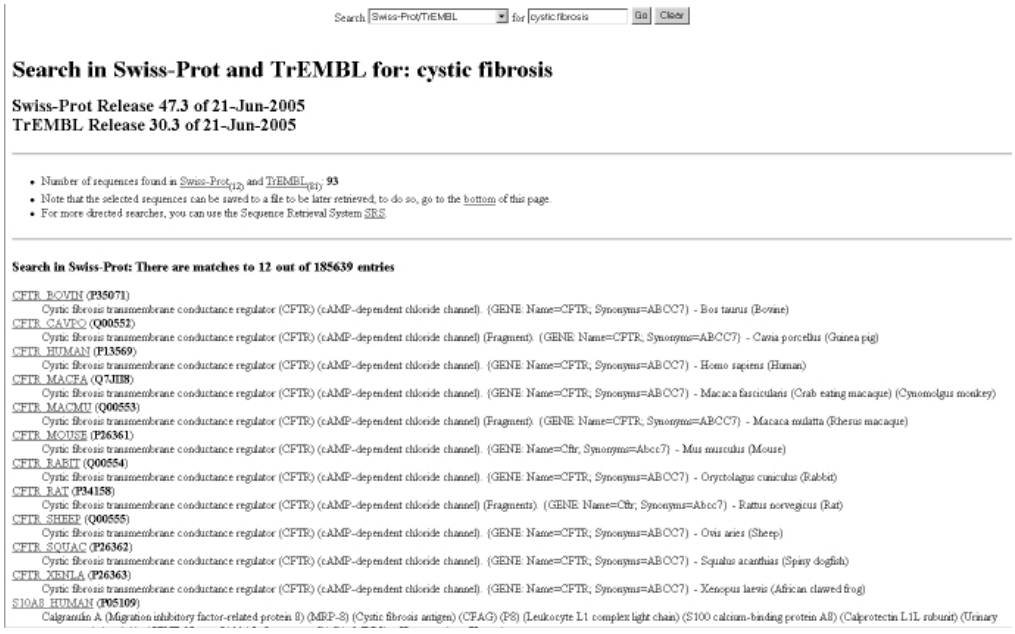
The following section discusses a general exemplification of how a web-based multiple choice exercise could be designed. A web-based multiple choice exercise should be accompanied by the information that is necessary to find the information the questions asks for. Otherwise it would, rather inconveniently, be necessary to consult at least one textbook or third-party web site to be able to find the requested information. This section therefore also discusses the information that such exercises could be accompanied with in a web-based learning environment.

Biological databases are highly specialised databases that researchers use to store their results. Thus, users of such a learning environment should not be expected to be familiar with these databases or the portals. A learning environment meant to teach the usage of the databases and portals should therefore start with an introduction to the databases and the portals.

One approach could be to start with an introduction to the database that explains what kind of information they provide. As there are many different databases, the databases should be presented separately, at least the most important ones, to reduce the possibility of confusing the various databases with each other. Each presentation could then be followed by a multiple choice exercise similar to that described in the above section.



(a)



(b)

Fig. 6.5: (a) the search results of a search for cystic fibrosis through Entrez. (b) the search results of the same search through Expasy.

Since the portals would not have been introduced at this point, direct links to the database entries should therefore be provided instead of requiring that a portal has to be used to find the correct database entry. The links should open in a separate browser window to make it possible to view the database entry and the exercise simultaneously.

An exercise regarding an entry in the nucleotide database (fig. 6.3a) of the Entrez portal (2005a) could include questions regarding which species the nucleotide sequence is found in, the id of the entry, the name of the sequence, the length of the sequence, who the authors of the second reference to this entry are, etc. The alternatives to the questions do not have to be taken from the entry, but to be able to provide reasonable comments on incorrect answers, they should. Comments like "the answer is wrong" are not particularly useful (Wolfe, 2001b). An example on a question with such alternatives could be:

Question:
How long is the nucleotide sequence of http://www.ncbi.nlm.nih.gov/entrez/viewer.fcgi?db=nucleotide&val=63025221?

Alt. 1: 63025221

Alt. 2: 2346

Alt. 3: 15854667

Alternative 2 is the correct answer, while the first alternative is the id of the nucleotide in the database, and the third alternative is the id of the first reference in the PubMed database (see fig. 6.3a). With these alternatives it would be possible to present comments like "The alternative you chose was wrong. 63025221 is the id of the nucleotide sequence in the database" or "15854667 is the id of the first reference in the PubMed database". Thus, it should be possible to learn about the information the entry provides even if the answers is incorrect.

The other databases Entrez provides access to could be presented in a similar manner, followed by similar exercises.

The next step would then be to introduce the various portals, and how they present database entries. Users should by now be somewhat familiar with the information the different databases provide, and it should therefore not be necessary to present more than a few choice entries. One approach could be to present some of the entries that were used in the introduction to the databases. Thus, the information in the entries would be familiar, and it should therefore

be easier to become familiar with how the different portals present database entries.

Another way of starting a learning environment that means to teach the usage of databases and portals could be to begin with a presentation of the various portals. Such an approach would make it possible to present entries from a selection of the databases of one portal, followed by a presentation of entries from a different selection of databases of another portal. As with the above described approach, each presentation should be accompanied by a multiple choice exercise. These exercises should also provide direct links to database entries because the principles of database searching yet have to be introduced. Thus, the same exercises as those described above could be used.

Independently of whether the first or the second approach should be used, users should by now be familiar with both the information the different databases provide, and the different presentation forms of the various portals. The next step could then be to introduce the principles of database searching.

The search engines that the portals provide are similar to the search engines that are used to search the web. The search engines of both biological portals and the web require a search phrase to be provided. The difference between the two types of search engines lies in the search phrases the search engines accept. Search engines for the web allow any kind of search phrase to be entered, including search phrases that contain logical operators such as AND, OR, and NOT. The search phrases a search engine of a biological portal allows may depend on the database that is to be searched. The Swiss-Prot database (a protein knowledge base) at Expasy (2005a), for instance, does not accept any of the logical operators. The operator AND can be included, but since this operator is automatically included in the search phrase if it is composed of more than two words separated by a space, there is no need to include it. The AND operator is also added automatically by the search engine at EBI (2005) when the protein database is searched. It does also allow the users to include any of the other operators, but since the other operators are removed from the search phrase before the search is performed, it is meaningless to include them. The search engine of Entrez (2005b), on the other hand, allows any of the logical operators to be included, but it is not apparent whether or not they are used. Biological databases may therefore be experienced as difficult to search, and a discussion regarding database searching should therefore start with short presentations of how the different databases are searched, accompanied by a set of exercises.

The exercises should start easy and gradually become more difficult, so that all students are able to complete the exercises. One way of starting easy could

be to include the search phrase in the question. The set of exercise regarding the Swiss-Prot database at Expasy could start with an exercise similar to this:

Question:
One of the following proteins is connected to the disease cystic fibrosis in humans. Which is it?

Alt. 1: CFTR_CAVPO

Alt. 2: CFTR_HUMAN

Alt. 3: S101A8_HUMAN

This exercise should be relatively easy to solve as it provides many hints to the correct answer. First of all, the text provides a search phrase that yields only two hits in the Swiss-Prot database: "cystic fibrosis human". The two hits are CFTR_HUMAN and S101A8_HUMAN. The Swiss-Prot database would have been presented before this exercise could be encountered, and the Swiss-Prot database should therefore be familiar. Hence, it should not be problematic to find out that the correct answer is CFTR_HUMAN (Expasy, 2005b); the protein S101A8_HUMAN is connected to chronic inflammations (Expasy, 2005c). Using the search phrase "cystic fibrosis" produces 12 entries from the Swiss-Prot database, of which the CFTR_CAVPO protein is one. This is a protein that is connected to chloride transportation in guinea pigs (Expasy, 2005c). It is also possible to use the answer alternatives as search phrases.

The exercises should gradually provide less and less hints, to force the users to use their knowledge about the databases and database searching to select the correct database to search and an efficient search phrase.

Chapter 7

Summary and Conclusions

This project is a part of the ongoing Flexible Learning (norw: Fleksibel Læring) (2005) project at the University of Oslo, which aims at contributing to the establishment of good learning environments for the students at the University of Oslo by integrating ICT with education and teaching.

The purpose of the project described in this document has been to explore how web-technology can be used to create an interactive web-based learning environment for bioinformatics. Bioinformatics is a complex field of science that combines mathematics, statistics, biology, and informatics; and to develop a complete learning environment that covers all aspects of this field of science in one project would have been impossible. This project has therefore focused on developing examples on how some of the algorithms used in the field of bioinformatics can be visualised in a web-based learning environment (i.e. the BioTeach system), and on discussing how exercises can be designed in such a learning environment.

Most of the work with this project has been concerned with the development of the BioTeach system. There were three major challenges that had to be dealt with in the development of this system. The first challenge was to implement the algorithms that were to be visualised. The second, and most difficult, challenge was to visualise these algorithms in a way that could improve the understanding of the algorithms. The BioTeach system provides two different approaches to meet this challenge. The visualisation of the Needleman-Wunsch algorithm presents the exact calculations that are involved in each step of an alignment of a pair of protein sequences. The

visualisations of the microarray data clustering algorithms specifically emphasize the steps most crucial to the understanding of the algorithms.

The third challenge was to create visualisation interfaces that appear uniform on all computers. One of the problems with web applications is that different web browsers may present the contents of a web application differently. Another problem is that some computers may use screen resolutions that are too low for the visualisations to fit completely into the browser window. Some measures have been taken to reduce these problems (i.e. the limits on the length of the sequences and the gap penalty limit in the Needleman-Wunsch visualisation). The visualisations have been tested in three different browsers, MS Internet Explorer, Mozilla, and Opera; and the visualisations appear approximately uniform in these browsers.

The BioTeach system clearly shows that it is possible to create web-based visualisations of algorithms. The interesting question is whether or not either of the visualisations improves the understanding of the algorithms that are visualised. There are, however, no empirical results to base such a prediction on. There are two reasons for this. First, there was, unfortunately, not enough time to conduct any studies of the effects of the visualisations. Second, if there had been time to conduct any studies, it would have been difficult to produce any reliable empirical results. The current implementation of the visualisations provides little information about the biological and statistical foundation the visualised algorithms are based on. It would therefore have been difficult for anyone who is not familiar with the principles of pairwise sequence alignment and clustering of microarray data to participate in a study of the effectiveness of the visualisations. A suitable test population would therefore have been the students that follow the introduction course to bioinformatics. There were less than 10 students that completed this course. Although instructive, a statistical study of the effectiveness of the visualisations based on such a small population would not render any statistically significant conclusions.

It is difficult to predict the usefulness of the visualisations without the support of empirical results. It should, however, be possible to use the Sourcer's Apprentice (SA) as a reference for a preliminary prediction of the usefulness of the visualisations. SA starts with a tutorial that explains and tests the skills that are necessary to use multiple sources of information effectively and correctly. The tutorial is then followed by a practice environment in which the skills are trained on sets of excerpts from authentic history texts. The visualisations of the BioTeach system are designed to visualise, step-by-step, how the implemented algorithms work on simplified, but authentic, datasets, and could therefore be, in combination with the textual explanations of the algorithms, viewed as tutorials to the algorithms. The visualisations do also

allow the datasets that are visualised to be experimented with, and the visualisations could therefore be viewed as virtual laboratory exercises similar to the laboratory exercises that are used in chemistry and physics. It should therefore, despite that the clustering visualisation is not supplemented by an explicit exercise, be possible to say, as a preliminary prediction, that the BioTeach system should be a useful tool for improving the understanding of the algorithms that are visualised.

One way of confirming or disproving this prediction can be to conduct a pre-test/post-test experiment in which the test population is divided into a control group and a treatment group. The first step of such an experiment is to administer a pre-test to that tests the population's understanding of the algorithms that are visualised by the BioTeach system. The pre-test could, for instance, be a paper and pencil test that contains exercises that require a sequence alignment with the Needleman-Wunsch algorithm, a hierarchical clustering, and a k-means batch clustering to be conducted by hand. Clustering datasets with the k-means online algorithm and SOMs are complicated and time consuming to conduct by hand, and ordinary questions regarding these algorithms should therefore be included instead of exercises that require such clusterings to be conducted. The next step could then be to provide the control group with detailed articles on the different algorithms, while the treatment group is provided with access to the BioTeach system in addition to be provided with the same detailed articles. The last step is to administer a post-test similar (or identical) to the pre-test after the two groups have had time to study the algorithms. The results of the pre- and post-tests for the two groups can then be compared to see if the results for the treatment group are better than those of the control group.

The BioTeach-system should, however, be tested more extensively to remove any bugs before any studies are performed. The interfaces should also be implemented with online-help such as tooltips before any studies are preformed. A tooltip is a textbox that is displayed when the mouse-pointer is placed over a component for a certain period of time. The clustering application is implemented with tooltips that provide a short description of the components of the interface, but these tooltips should be implemented with more elaborate descriptions and help.

The BioTeach system is not a complete learning environment for bioinformatics, and it could be developed further if it proves to be a useful educational tool. The system could, for instance, be implemented with clustering exercises that explicitly test the understanding of the implemented clustering algorithms. A hierarchical clustering exercise, for instance, could present a dataset accompanied by a dendrogram in which the numbers of the

genes are replaced by text fields. The exercise would then be to enter the number of the genes into the correct text fields. Another extension could be to include descriptions of the mathematical and biological foundation upon which the implemented algorithms are based to enable others than those who are familiar with the implemented algorithms to use the system. The exercises discussed in chapter 6 could also be interesting to implement

References

Anderson, M. D. (2001) Individual Characteristics and Web-Based Courses. In Wolfe, C. R. (Ed.), *Learning and Teaching on the World Wide Web,* Academic Press.

Baxevanis, A. D. (2005) Assessing Pairwise Sequence Similarity: BLAST and FASTA. In Baxevanis, A. D. and Ouellette, B. F. F. (Eds.), *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, Third Edition,* John Wiley & Sons.

Berrar, D. P., Dubitsky, W., Downes, C. S, and Granzow, M. (2003) Introduction to microarray data analysis. In Berrar, D. P., Dubitsky, W., and Granzow, M., (Eds.), *A practical approach to microarray data analysis,* Kluwer Academic Publishers.

Britt, M. A. and Gabrys, G. L. (2001) Teaching Advanced Literacy Skills for the World Wide Web. In Wolfe, C. R. (Ed.), *Learning and Teaching on the World Wide Web,* Academic Press.

Callaway, D. R (2001) *Inside servlets: server-side programming for the Java platform 2nd edition,* Addison Wesley.

Causton, H., Quackenbush, J., and Brazma, A. (2003) *Microarray/gene expressions data analysis,* Blackwell Publishing.

Draghici, S. (2003) *Data analysis tools for DNA microarrays,* Chapman & Hall/CRC.

Durbin, R., Eddy, S., Krogh, A., and Mitchison G. (1998) *Biological sequence analysis: probabilistic models of proteins and nucleic acids,* Cambridge University Press.

EBI (2005) *European Bioinformatics Institute* [online]. Available: http://www.ebi.ac.uk, July 27, 2005.

Entrez (2005a) *NCBI Sequence Viewer v2.0* [online]. Available: http://www.ncbi.nlm.nih.gov/entrez/viewer.fcgi?db=nucleotide&val=63025221, July 27, 2005.

Entrez (2005b) *Entrez cross-database search* [online]. Available: http://www.ncbi.nlm.nih.gov/Entrez, July 27, 2005.

Expasy (2005a) *ExPASy Proteomics Server* [online]. Available: http://www.expasy.org, July 27, 2005.

Expasy (2005b) *UniProtKB/Swiss-Prot entry P13569 [CFTR_HUMAN] Cystic fibrosis transmembrane conductance regulator* [online]. Available: http://www.expasy.org/uniprot/P13569, July 27, 2005.

Expasy (2005c) *UniProtKB/Swiss-Prot entry P05109 [S101A8_HUMAN] Calgranulin A* [online]. Available: http://www.expasy.org/uniprot/P05109, July 27, 2005.

Expasy (2005d) *UniProtKB/Swiss-Prot entry Q00552 [CFTR_CAVPO] Cystic fibrosis transmembrane conductance regulator [Fragment]* [online]. Available: http://www.expasy.org/uniprot/q00552, July 27, 2005.

Flexible Learning (2005) *Fleksibel læring ved UiO* [online]. Available: http://www.fleksibel-laering.uio.no, July 27, 2005.

Higgs, P. G. and Attwood, T. K. (2005) *Bioinformatics and molecular evolution,* Blackwell Publishing.

Lazlo, M. J. (1996) *Computational geometry and computer graphics in C++,* Prentice Hall.

Lewis, J. And Loftus, W. (1998) *Java Software Solutions: foundations of program design,* Addison Wesley.

NCBI (2005) *Bioinformatics Factsheet* [on-line]. Available: http://www.ncbi.nlm.nih.gov/About/primer/bioinformatics.html, June 7, 2005.

Pevzner, P. A. (2000) *Computational molecular biology: An algorithmic approach*, the MIT Press.

Quackenbush, J. (2005) Using DNA Microarrays to Assay Gene Expression. In Baxevanis, A. D. and Ouellette, B. F. F. (Eds.), *Bioinformatics: A Practical Guide to the Analysis of Genes and Proteins, Third Edition,* John Wiley & Sons.

Ripley, B. D. (1996) *Pattern recognition and neural networks,* Cambridge University Press.

Sandnes, F. E. (2002) *Moderne applikasjonsutvikling med Java for web: tynne klienter og fete tjenere,* Tapir akademisk forlag.

Wolfe, C. R. (2001a) Learning and Teaching on the World Wide Web. In Wolfe, C. R. (Ed.), *Learning and Teaching on the World Wide Web,* Academic Press.

Wolfe, C. R. (2001b) Creating Informal Learning Environments on the World Wide Web. In Wolfe, C. R. (Ed.), *Learning and Teaching on the World Wide Web,* Academic Press.

Appendix A

Downloading, installing and configuring the BioTeach-system

This appendix is concerned with the installation and configuration of the BioTeach-system. The appendix also provides instructions as to how and where the source code of the system can be downloaded.

The BioTeach-system is installed and configured in the following steps:

1. Download the archives portal_and_clustering.zip and needle.war from http://www.ifi.uio.no/bioinf/Prosjekter/BioTeach (this page can also be accessed through the downloads-link in the visualisation menu of the system.

2. Unpack the archive portal_and_clustering.zip on a web-server, or in a directory that is accessible through the web. The portal and the clustering applet should now be accessible through the web. If not, check directory and file permissions.

3. deploy the needle.war-archive on a servlet-container such as Tomcat

4. enter the BioTeach-directory created when the archive portal_and_clustering.zip was unpacked

5. open the file menu.html in a text-editor

6. change the href-attribute of the link to the alignment-application to: <path to servlet container>/needle

7. locate and enter the needle-directory on the servlet container (should be created when the needle.war-archive is deployed)

8. open the file web.xml located in the WEB-INF-subdirectory in a text editor

9. change the value of the init-parameter matrixpath to <path_to_needle_directory_on_servlet_container>/matrices
   There are two instances of this init-parameter, and both have to be changed.

The alignment application has only been tested with the Tomcat servlet container, and additional steps may therefore be needed in order to install the alignment application on a different servlet container.

It is also possible to download the bio.jar library (see appendix B) from the downloads-page. This library is embedded in the needle.war-archive, and it is not necessary to download bio.jar to install the BioTeach-system. The library is intended for visitors who might be interested in using the library in their own projects.

The downloads-page also provides a link that downloads the source code. The source code is packaged in a zip-archive that produces the following directory tree when unpacked:

```
|-<alignment>
      |-<demo>
            |-<control>
            |-<model>
            |-<view>
      |
      |-<exercise>
            |-<control>
            |-<model>
            |-<view>
|-<bio>
|-<clustering>
```

The alignment-directory of the archive contains the source code for the Needleman-Wunsch visualisation application and the Needleman-Wunsch exercise; the demo- subdirectory contains the source code for the visualisiation, while the exercise-subdirectory contains the source code for the exercise. The visualisation application and the exercise application are implemented according to the JSP Model 2-architecture (see appendix B), and the subdirectories are therefore further divided into three subdirectories each. The control-subdirectory contains the source code for the control layer of the applications, the model-subdirectory contains the source code for the model-layer, and the view-subdirectory contains the source code for the view-layer.

The bio-directory of the archive contains the source code for the bio.jar library (see appendix B), and the clustering-directory contains the source code for the clustering applet.

Appendix B

Technical Solution of the BioTeach-system

This appendix serves as an introduction to the technology that is used in the implementation of the BioTeach-system, and a description of the program flow of the system.

B.1 Applied technology

There are various technologies that could be used to implement the BioTeach system. This section provides a short introduction to the technology that is used in the implementation of the BioTeach-system.

B.1.1 Java Applets

Java Applets (Lewis & Loftus, 1998) are Java applications that are designed to be embedded and accessed through web-pages. Applets can be implemented with the same functionality as standard java applications, and are well suited

for designing web-applications that uses graphical interfaces or components to interact with the users (e.g. the clustering application of the BioTeach-system). An applet must extend either the JApplet-class or the Applet-class in order to be run in a web-browser (or other environment designed to run applets). Applets are embedded in web pages by including the tag

<APPLET code=”name_of_class_that_extends_JApplet.class”
codebase=”path_to_java_class_files” width=”x” height=”y”>

The code attribute in the tag is mandatory, and must contain the name of the class that extends the JApplet (or Applet) class. The codebase attribute is only necessary if the class-files that compose the applet are located in a different directory than the web-page in which the applet-tag is included. The width and height attributes sets the width and height of the applet.

B.1.2 The JSP Model 2 architecture

The JSP Model 2 architecture (Callaway, 2001) is another approach to web-application design. The alignment application of the BioTeach system is designed around this architecture. The JSP Model 2 architecture divides an application into three layers: the control layer, the model layer, and the view layer. The control layer handles all interaction with the users, the model layer acts as an intermediary between the control layer and the data source, while the view layer generates the presentations of the data. The purpose of dividing an application into these layers is to separate business-logic (e.g. data processing) from application-logic (e.g. request processing), and to separate business and application logic from presentation logic (e.g. web pages). Applications that implement this architecture are said to be server-side applications because the application is executed on a server. Applets are said to be client-side applications because they are executed on the computer that opens the page in which the applet is embedded. Java-based server-side applications require a container, or server, such as Tomcat to function.

The communication between the users and a server-side application is handled by the http-protocol (Sandnes, 2002). The http-protocol is request/response based, that is, the actions of the users create an http-request that is sent to the web-server, and the server generates an http-response that is returned to the users. The requests that are generated can be of different types. Opening a link in a web page, for instance, generates a GET-request. The alignment application uses the GET-request in links, and the POST-request in forms.

In the JSP Model 2 architecture all requests are processed by the control layer. The control layer of the alignment application of the BioTeach-system is represented by an http-servlet. In short, an http-servlet is a class of Java-based applications that are designed to receive and process http-requests, and generate http-responses (please refer to Callaway, 2001, and Sandnes, 2002, for a thorough discussion of servlets). An application that is to be used as an http-servlet must contain a class that extends the class javax.servlet.HttpServlet. The HttpServlet-class receives the http-requests as objects, and handles the different requests with different methods. The GET-request, for instance, triggers the doGet-method of an http-servlet, while a POST-request triggers the doPost-method. An http-servlet must, however, be given a logical name and mapped to an URL in order to be able to process any form of http-requests. The Tomcat server uses an xml-file, web.xml, to accomplish this. The following xml-code gives the ControlServlet-class of the Needleman-Wunsch alignment visulisation a name and maps it to respond to a certain set of URLs:

```
<servlet>
      <servlet-name>demoCtrl</servlet-name>
      <servlet-class> demo.control.ControlServlet
      </servlet-class>

      <init-param>
        <param-name>matrixpath</param-name>
        <param-value>
            C:\Programfiler\Apache Software Foundation\
            Tomcat 5.0\webapps\needle\matrices
      </param-value>
      </init-param>
</servlet>

<servlet-mapping>
     <servlet-name>demoCtrl</servlet-name>
     <url-pattern>*.demo</url-pattern>
</servlet-mapping>
```

The servlet-name-tag at the top contains the name of the servlet (e.g. demoCtrl), while the servlet-class-tag contains the servlet-class (e.g. demo.control.ControlServlet) that is associated with the name. The servlet-mapping-tag maps the servlet associated with the servlet-name to respond to all requests for URLs with the postfix .demo (e.g. start.demo, end.demo, etc).

The web.xml file can also be used to define init-parameters for a servlet. Servlets can easily access init-parameters given in the web.xml file through the method getInitParameter, and init-parameters are an easy way of configuring a servlet.

The model-layer in the JSP Model 2 architecture is concerned with retrieving and processing information from the data source (the bio.jar library in the alignment application). The model-layer is composed of JavaBeans (Callaway, 2001; Sandnes 2002), which are Java-classes that retrieve information from the data source, and format the retrieved information so the information can be used by the view-layer. JavaBeans can be embedded in web pages, and the information stored in a JavaBean can be accessed from a web page by including certain directives in the web page. JavaBeans cannot, however, be used to create complex presentations in which the length of the document is unknown, or that require the data stored in the JavaBeans to be iterated over.

The score matrix of the Needleman-Wunsch alignment visualisation and the exercise is defined by the users, and it is therefore impossible to know the dimensions of the score matrix on beforehand. Thus, the presentation of the Needleman-Wunsch visualisation and exercise cannot be created by embedding JavaBeans in the web pages that contains these presentations. JavaBeans are, however, still used in the model-layer to retrieve and format information from the bio.jar library, but the retrieved and formatted information is returned to the control-layer instead of being accessed directly from the web pages of the view-layer.

The problem with creating the score matrix of the Needleman-Wunsch algorithm is solved by using custom tags. Custom tags (Callaway, 2001; Sandnes, 2002) are a powerful tool for creating dynamic web pages because the tags can be designed to produce any kind of HTML-code. A custom tag is essentially a Java-class that is given a logical name, which is used to execute the contents of the Java-class in a web-page. The Java-class that implements a custom tag is commonly referred to as a tag-handler class. A tag-handler class has to extend either the javax.servlet.jsp.tagext.TagSupport-class or the javax.servlet.jsp.tagext.BodyTagSupport-class. The custom tags that are implemented in the alignment application do not have a body (i.e. there is no HTML or other code between the opening tag and the closing tag), and the tag-handler-classes therefore extends the former class. The custom tags of the alignment application produces the HTML-code for the configuration form, the visualisation interface, the exercise interface, and the correct score matrix of the exercise. The tag-handler-classes for these tags and the web pages that include these tags make up the view-layer of the alignment application.

The tag-handler classes are made available as custom tags by mapping the classes to a logical name in a tag-library-descriptor, or tld-file. The tag-handler class for the custom tag that produces the configuration form is mapped to a logical name by including the following code in a tag-library-descriptor:

```
<tag>
     <name>NWForm</name>
     <tag-class>demo.view.FormTag</tag-class>
     <body-content>empty</body-content>

     <attribute>
      <name>action</name>
      <required>true</required>
     </attribute>
  </tag>
```

In this code segment maps the class demo.view.FormTag to the name NWForm. The attribute-tag implies that the NWForm-tag accepts one attribute, the action-attribute, and that the attribute is mandatory, otherwise the tag will not work. The demo.view.FormTag-class must therefore implement a method named setAction in order for the given attribute to be made available in the tag-handler class. Other attributes can be implemented by adding additional attribute tags, and by implementing methods in the tag-handler-classes named setNameOfAttribute (e.g. setAge if the name of the attribute is age).

Custom tags are included in a web-page by including the directive

```
<%@ taglib uri="path_to_tag_library_descriptor" prefix="pre-
                    fix_of_own_choice" %>
```

at the top of the web page. A custom tag without body or attributes is included by including the following directive where the HTML-code generated by the custom tag is to be inserted:

```
<prefix_of_own_choice:tag_name />
```

Web-pages that include embedded JavaBeans, custom tags, or any other form of Java-technology (e.g. scriptlets) are commonly referred to as Java Server Pages or JSP-documents.

B.1.3 The bio.jar library

The bio.jar library is the data source of the alignment application of the BioTeach-system, and contains the Java-classes that implement the Needleman-Wunsch alignment algorithm. This section discusses the general steps that are involved when using the bio.jar library to conduct a global pairwise sequence alignment, the data structures that are used to represent the score matrix of an alignment, the optimal alignment (or alignments) of a pair

of sequences, and the optimal path through a score matrix. How these data structures are retrieved is also discussed. Readers are advised to consult the source code for the exact implementation of the Needleman-Wunsch alignment algorithm. Please refer to appendix A for instructions on downloading the source code.

The classes that implement the Needleman-Wunsch alignment algorithm are:

- the Needleman-class
- the Element-class
- the SubstMatrix-class

The Needleman-class implements the algorithm, the Element-class represents a cell in a score matrix, and the SubstMatrix-class represents a substitution matrix.

A global pairwise sequence alignment starts by creating an instance of the Needleman-class, and running the init-method of the Needleman-class. The init-method creates an instance of the SubstMatrix-class, which the Needleman-class uses to retrieve the scores of aligning a residue pair from the substitution matrix that is used in the alignment. The init-method then runs the setWeight-method of the Needleman-class, which computes the score matrix of the alignment, and the align-method of the Needleman-class, which finds the optimal alignment (or alignment) of the pair of sequences.

The score matrix of an alignment is represented by a two-dimensional Element array of (n+1) columns and (m+1) rows, n being the length of the first sequence and m being the length of the second sequence. Each Element-object in the array represents a cell in the score matrix, and contains the score and the pointers of a cell. The optimal path through a score matrix is represented by a two-dimensional boolean array of (n+1) columns and (m+1) rows. Each boolean value given in this array either states that the Element object given in the corresponding row and column of the Element array is included in the optimal path (i.e. the boolean value is true), or that the Element-object is not included in the optimal path (i.e. the boolean value is false). The optimal alignment (or alignments) of a pair of sequences is stored in a two-dimensional String array of x columns and 2 rows. Each String-object in this array represents a sequence, and each column represents an optimal alignment of a sequence pair.

The score matrix is retrieved from the Needleman-class by calling the getWeightMatrix-method, the optimal path is retrieved by calling the getPath-

method, the optimal score of an alignment is retrieved by calling the getScore-method, and the optimal alignment (or alignments) is retrieved by calling the getAlignment-method.

B.1.4 Graham's Scan

Graham's scan (Lazlo, 1996) is a linear programming algorithm that is used to find the convex hull of a set of points. This algorithm is implemented in the CoordinateArea-class of the clustering algorithm to find the convex hull of clustered points. The algorithm starts by selecting an extreme point, p0, e.g. the point with the highest x-coordinate and the lowest y coordinate. The remaining points are then sorted around p0. A hypothetical line is then drawn between p0 and the next point in the sorted point collection, p1. The position of the third point in the sorted point collection is then compared to the line between p0 and p1. If p2 is to the left of the line, p2 may be in the convex hull, and is stored. The hypothetical line between p_0 and p_1 is then replaced by a hypothetical line between p_1 and p_2. If p2 is to the right of the line, p1 is not in the convex hull. p_1 is discarded while p_2 is stored. The hypothetical line between p0 and p1 is then replaced by a line between p0 and p2.

The next step is to compare the position of the fourth point, p_3, to the hypothetical line between the two previously stored points (i.e. p_1 and p_2, or p_0 and p_2) If p3 is to the left of the hypothetical line, p_3 may be in the convex hull, and is stored. The previous hypothetical line is then replaced by a hypothetical line between the previously stored point and p_3. If p_3 is to the right of the hypothetical line, the point stored in the previous step is not in the convex hull, and is discarded. The position of p_3 is then compared to the hypothetical line between the two points that was stored prior to the previous point. If p_3 is to the left of this line, p_3 may be in the convex hull, and is stored. Otherwise, the last point to be stored prior to the point that was removed is not in the convex hull, and is discarded. The algorithm continues the process of finding the points that compose the convex hull of the set of point by

1. draw a hypothetical line between the two points that was last stored

2. retrieve the next point from the sorted set of points (hence referred to as the current point)

3. compare the position of the current point with the hypothetical line and

    1. if the current point is to the left of the line: store the the current point and go to step 1

    2. if the current point is to the right: remove the point that was last stored, and draw a hypothetical line between the point that now is the lastly stored point and the point that was stored prior to the new lastly saved point. Got step 3.

Repeat this cycle until the last point in the sorted set is reached.

B.2 Application organisation and program flow

This section describes the organisation of the portal and the applications of the BioTeach system and the program flow of the applications, and is intended as a supplement to the source code. Readers are advised to consult the source code for the exact implementation of the system. Please refer to appendix A for instructions on downloading the source code.

The BioTeach system is divided in three separate modules/applications:

- the portal
- the alignment application
- the clustering application

B.2.1 The portal

The portal is the top level presentation layer created to provide a uniform interface to the two applications. The two applications are, as described in the walkthrough, presented in the center frame without changing the menu or the header, thus creating the desired uniform look. The portal is designed with HTML-framesets, HTML-pages, and cascading style sheets. The directory tree of the portal is organized in the following manner:

```
<path to portal>
        |
        |-HTML-documents
        |
        |-<gfx>
                |-images and graphics
        |
        |-<css>
                |-Cascading style sheets
```

B.2.1.1 Index.html

Index.html is the main page of the application and contains the framesets. Framesets are used to create a grid of frames which then is used to position the various HTML-documents on the screen. This file contains framesets that divide the screen into 5 frames (See fig. B.1 for layout):

- top; the header-frame
- bottom; the frame used for contact information
- menu; the menu frame.
- main; the presentation frame
- help; the help menu frame

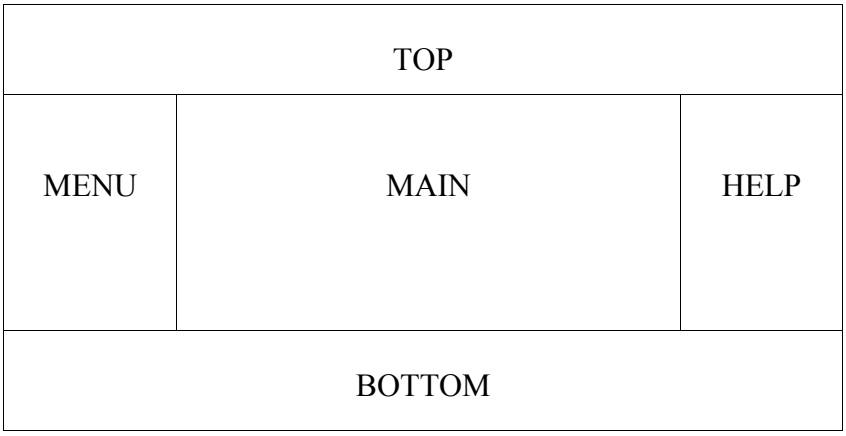


Fig. B1: The layout of the frames of index.html

B.2.1.2 top.html

Top.html is located in the top frame and contains the logo. The logo is placed within a HTML-table to center it on the screen.

B.2.1.3 menu.html

Menu.html is located in the menu frame and contains the menu. The menu consists of ordinary HTML-links which opens the applications in the centre frame. The HTML-links are placed within two HTML-tables, an outer table for centring and an inner table for vertical alignment, to create an orderly menu. To ensure that the modules are presented in the main frame, the target parameter of each link is set to the main frame. The menu also contains a link to the download page of the BioTeach system.

B.2.1.4 help.html

help.html is located in the help frame and contains the help menu. The help menu is composed of ordinary HTML-links that opens the help documents in a separate browser window so as not to interrupt with the running of any of the applications. The target parameters of these links are set to _new in order to ensure that the links opens in a separate browser window.

B.2.1.5 Bottom.html

Bottom.html is located in the bottom frame, and contains contact information.

B.2.1.6 Main.css

Main.css is a cascading style sheet that controls the appearance of both the framework and the clustering application. Main.css was originally intended to be included in the alignment application as well, but it proved difficult to include this style sheet in the alignment application. Instead, an identical style sheet is included in the alignment application.

Cascading style sheets are very powerful, as they make it possible to centralise all settings concerning the appearance of a site or application, and changes in a style sheet appear immediately in the documents that are linked to a style sheet. Style sheets has the ability to format standard HTML-tags, e.g. the body-tag, as well as creating own classes of formatting with the notation

SPAN.<class-name>{<settings>}

Main.css sets the following:

- the background colour to ivory by formatting the body tag
- the appearance of the text of the <H1>-tag
- the background colour of the <TH>-tag (table header)

and creates the following classes:

- optimal; used to mark the optimal path through the score matrix of the alignment
- normal; used on normal text
- max; used to mark the calculation(s) which yields the max score
- error; used on error message text

B.2.2 The Alignment application

The alignment application is composed of two separate applications: the visualisation application and the exercise application. Both the visualisation application and the exercise application are implemented according to the JSP Model 2 architecture.

The alignment application is packaged as a web archive, or war-file, that is organised as follows:

```
<path to alignment application>
    |- HTML and JSP documents
    |-<css>
          |- cascading style sheet
    |-<gfx>
          |- images and graphics
    |-<matrices>
          |- substitution matrix files
    |-<META-INF>
          |- MANIFEST.MF (generated by Tomcat)
    |-<WEB-INF>
          |- web.xml
          |
          |-<classes>
                |- the demo and the exercise package directory tree
          |-<lib>
                |- bio.jar
          |-<tld>
                |-<demo>
                      |- alignlib.tld
                |-<exercise>
                      |-exercise.tld
```

All HTML and JSP documents are linked to the style sheet found in the css directory. It was, as explained in the previous section, difficult to link the HTML and JSP documents in this application to the style sheet used in the portal, so an identical style sheet had to be created for this application.

Selecting the alignment application from the portal menu opens the page index.html. This document provides a short introduction to the application and links to the alignment visualisation and the alignment exercise.

Selecting the visualisation application from the index.html page opens the page demoIndex.html. This page contains a short introduction to the visualisation and provides links to the simple and the advanced visualisation.

B.2.2.1 Starting a simple visualisation

Selecting the simple visualisation opens the page simpleDemo.html. This page describes the parameters that are used in the simple visualisation, and provides a button that starts the simple visualisation (the simple visualisation presents the alignment of the sequences RWA and LSP using a gap penalty of -5 and the Blosum62 substitution matrix). The start-button is named "demo" and is embedded in a form that sends a POST-request to the page startSimple.demo. startSimple.demo and all other pages with the postfix .demo is a mapping to the control layer of the visualisation application represented by the class ControlServlet in the package demo.control (see web.xml in the WEB-INF directory for all servlet mappings in the alignment application). Hence, pressing the start-button of the simple visualisation creates an instance of the ControlServlet and sends a POST-request to the instance of the ControlServlet. Creating an instance of the ControlServlet also initialises the instance. Initialisation of the instance is handled by the init-method of the ControlServlet-class. This method reads the init-parameter "matrixpath" found in the web.xml file. This parameter contains the path to the substitution matrix files that are implemented in the visualisation.

Once the initialisation process is completed, the visualisation application proceeds to process the POST-request generated by the start-button of the simple visualisation. The ControlServlet handles all POST-requests generated by the visualisation application with the doPost-method, and the doPost-method uses the name of the button that generated the request to handle the different POST-requests. The start-button of the simple visualisation is named "demo" and this button triggers the setAttributes-method of the ControlServlet. This method first sets the parameters of the alignment, and then checks if the parameters are valid. The parameters of the simple visualisation are hard-coded in the application, and are therefore always valid. The method then creates an instance of the NeedleBean-class. The NeedleBean-class is one of the two classes that are situated in the model-layer of the application (i.e. contained in the demo.model package). The NeedleBean-class acts as an intermediary between the ControlServlet and the bio.jar library. The purpose of using an intermediary instead of implementing the Needleman-Wunsch algorithm directly into the visualisation application is to separate the implementation of the algorithm from the visualisation of the algorithm. It is therefore possible to use the implementation of the algorithm in other applications by importing the classes of the bio.jar library. Creating an instance of the NeedleBean-class also creates an instance of the Needleman class in the bio.jar library. The Needleman-class is responsible for conducting the alignment of a pair of sequences.

Once the instance of the NeedleBean class is created, the setAttributes-method of the ControlServlet calls the init-method of the NeedleBean-class, which in turn conducts the alignment of the pair of sequences by running the init-method of the Needleman-class. The init-method of the NeedleBean-class then retrieves and stores the score matrix computed by the Needleman-class. The ControlServlet then calls the method getMatrix in the NeedleBean-class. This method retrieves the direction matrix from the Needleman-class and converts both the score matrix and the direction matrix from primitive integer and boolean matrices into matrices of Integer and Boolean objects. The converted matrices are stored in an instance of the Matrix-class along with the length of each of the two aligned sequences. The Matrix-class is the second of the two classes that are situated in the model layer (i.e. located in the demo.model package). This class is designed to store all necessary information about an alignment of a pair of sequences. The instance of the Matrix-class is then returned to the ControlServlet.

The ControlServlet then creates an instance of the HttpSession-class. Instances of this class are used to store and pass information between the control-layer and the view-layer. The information stored in the HttpSession is:

- the pair of sequences represented by the strings s1 and s2
- the gap penalty
- the path to the matrices
- the name of the substitution matrix used in the alignment
- the score matrix of the alignment
- the direction matrix of the alignment
- the optimal alignments retrieved from the Needleman-class through the getAlignments-method of the NeedleBean-class
- the optimal path through the score matrix retrieved from the Needleman-class through the getOptimalPath-method of the NeedleBean-classthe score of aligning two residues represented by the Integer-object weight. When a visualisation is started, there are no residues that have been aligned, so the weight is set to zero

The HttpSession is also used to store information about the state of the application:

- the coordinates of the score matrix cell whose score and directions are to be visualised represented by the Integer-objects x and y. At the start of a visualisation these objects are set to zero

- the Boolean object xinit and yinit are used to determine whether or not the first row and the first column are being initialised. At the start of a visualisation xinit is set to true, while yinit is set to false

- the Boolean object finished is used to determine whether or not the score matrix is completed. Set to false at the start of a visualisation

- the Boolean object simple is used to determine whether or not the current visualisation is a simple visualisation. Set to true when the visualisation is simple

Once all necessary information is stored in the HttpSession, the setAttributes-method of the ControlServlet returns the path to the next page to be displayed by the view-layer: the page align.jsp. The doPost-method of the ControlServlet then redirects the application to the align.jsp page, which starts the visualisation.

B.2.2.2 Starting an advanced visualisation

Selecting the advanced visualisation from the demoIndex.html page sends a GET-request to the ControlServlet of the visualisation application as the link points to the page advanced.demo. Thus, using this link creates and initialises an instance of the ControlServlet-class. The initialisation process is identical to the process described in the previous section. The ControlServlet handles all GET-requests with the doGet-method. The doGet-method calls the setMatrices-method, which reads all available substitution matrices into a Properties-object. The available matrices are read from the file matrix.properties in the matrices-directory. The matrix.properties lists the available substitution matrices on the form:

Matrix1 = name_of_matrix_1
Matrix2 = name_of_matrix_2
.
.
.
MatrixN = name_of _matrix_N

The names of the matrices have to correspond with the names of the matrix-files (e.g. Blosum62.txt) without the .txt extention. This list of matrices is used to populate the list of available substitution matrices in the configuration form of the advanced visualisation.

Once the matrix-names have been read, the ControlServlet creates an instance of the HttpSession-class, and stores the Properties-object in the HttpSession and returns to the doGet-method. The doGet-method then redirects the application to the page advDemo.jsp.

The page advDemo.jsp is linked to the tag library alignlib.tld found in the demo-directory of the tld-directory. The tag library alignlib.tld contains the mappings between the custom tags and the tag-handler classes. A link between a jsp-page and a tag library is created by including the directive

```
<%@ taglib uri="path_to_tag_library" prefix="pfx" %>
```

at the top of the page. For advDemo.jsp the uri attribute is "/WEB-INF/taglib/demo/alignlib.tld", and the used prefix is "demo". By including a directive with these attributes, all tags given in the tag library aligntag.tld can be included in advDemo.jsp by including the directive

<demo:tag_name attributes />

One of the two tags that are included in alignlib.tld is the tag NWForm, which is mapped to the class FormTag in the view-layer (i.e. the class is found in the demo.view package). The FormTag-class prints the HTML-code for the configuration form of the advanced visualisation to the jsp-page the tag is included in. The NWForm-tag requires a target-attribute to be provided in order to function properly. The target attribute must contain a mapping to the ControlServlet (i.e. a filename with a .demo postfix) in order for the advanced visualisation to function properly. The directive used to include the NWForm-tag in the advDemo.jsp page is:

<demo:NWForm action="startAdvanced.demo"/>

Thus, selecting the advanced visualisation from the demoIndex.html page

1) creates an instance of the ControlServlet

2) reads the path to the substitution matrices

3) reads the list of available substitution matrices and stores the list in a Properties-object, which is stored in a HttpSession

4) redirects the application to the page advDemo.jsp

5) creates an instance of the FormTag-class, which prints the HTML-code for the configuration form of the advanced visualisation

Creating an instance of the FormTag-class stores the action-attribute given in the directive and runs the doStartTag-method. This method prints the text fields and the start-button to the advDemo.jsp page. The list of available substitution matrices is also populated by retrieving the Properties-object from the HttpSession, and reading the matrices from the Properties-object. The advDemo.jsp page also includes directions as to how an advanced visualisation is configured.

Pressing the start-button sends a POST-request to the ControlServlet. The ControlServlet handles this request as it handles the POST-request of the simple visualisation with one exception: the parameters to the algorithm (i.e. the sequences, gap penalty, and substitution matrix) are not hard-coded. These parameters are supplied by the users through the configuration form, and have to be retrieved from the POST-request. The setAttributes-method of the ControlServlet handles the retrieval of the parameters, before an instance of the NeedleBean-class is created and the alignment is conducted (see previous section).

Once the necessary information has been stored in the HttpSession, the ControlServlet redirects the application to the align.jsp page that starts the visualisation

B.2.2.3 The visualisation

The visualisation starts once the page align.jsp is opened. This page is linked to the tag library alignlib.tld using the same directive as the advDemo.jsp page. Align.jsp incudes, apart from a header and information about how to view the selected substitution matrix, the tag Align. This tag is mapped to the class AlignTag in the view-layer (i.e. included in the package demo.view), and requires the attribute action to be set. The directive used to include this tag in align.jsp is

<demo:Align action=”show.demo”/>

This tag prints the score matrix of an alignment as well as the computation table (or the optimal alignments) and the set of buttons that controls the visualisation.

Thus, once align.jsp opens, an instance of the AlignTag-class is created, and the visualisation interface is printed to the align.jsp page. The printing of the interface starts with the doStartTag-method in the AlignTag-class, which is called as the instance of the AlignTag-class is created. This method calls the getAttributes-method, which retrieves all attributes from the HttpSession-object. The printing of the interface to align.jsp is then started by calling the visualise-method. The visualise-method is the first of many methods that are involved in the printing of the interface. The methods involved in the printing process eventually decomposes the printing of the interface into the printing of the cells of the score matrix, the printing of the computation table (or the optimal alignments), the printing of the link to the substitution matrix, and the printing of the control-buttons. The printing of the interface starts with the printing of the one-celled outer table that is used to centre the other components. The score matrix and the other components of the interface are printed in steps starting with the score matrix. The first step is to format the table that represents the score matrix followed by the printing of the header row. The score and the pointers of remaining cells of the score matrix is printed, row by row and cell by cell, until the current cell, cell F(x, y), is reached. The computation table for the current cell is then printed, followed by the link to the chosen substitution matrix and the control-buttons.

Pressing any of these buttons sends a POST-request to the doPost-method of the ControlServlet:

- the next-button updates the state-information stored in the HttpSession to enable the AlignTag-class to display the score, pointers, and computations of the next cell

- the previous-button updates the state-information stored in the HttpSession to enable the AlignTag-class to display the score, pointers, and computations of the previous cell

- the finish-button updates the state-information to enable the AlignTag-class to display the completed score matrix and the optimal alignments

- the new alignment-button resets the HttpSession to remove any stored information, and calls the doGet-method to redirect the application to the advDemo.jsp page

- the main page-button resets the HttpSession and redirects the application to index.html

B.2.2.4 Running an exercise

Selecting the exercise application from the page index.html sends a GET-request to the page start.exercise. The page start.exercise and all other pages with the postfix .exercise are mapped to the control-layer of the exercise application represented by the ControlServlet class in the package exercise.control (see web.xml). Thus, using the link to the exercise application creates an instance of the ControlServlet of the exercise application, and initialises the instance. The initialisation of the ControlServlet is identical to the initialisation process of the ControlServlet of the visualisation application, that is, the path to the directory that contains the substitution matrix files is read from web.xml. The doGet-method in the ControlServlet-class handles all GET-requests and is identical to the doGet-method of the ControlServlet of the visualisation application. Thus, the GET-request sent to the ControlServlet triggers the doGet-method, which reads the available substitution matrices from the matrix.properties file in the matrices directory and stores them in a Propery-object. The Property-object is in turn stored in an instance of the HttpSession-class. The doGet-method then redirects the exercise application to the page exercise.jsp.

The page exercise.jsp is linked to the same tag-library as the page advDemo.jsp using the directive

```
<%@ taglib uri="/WEB-INF/tld/demo/alignlib.tld"
                prefix="print" %>
```

Exercise.jsp includes the same description included in the advDemo.jsp page as well as the same configuration form. The configuration form is included with the directive

```
<print:NWForm action="start.excercise"/>
```

The NWForm tag is, as in the visualisation application, handled by the class FormTag found in the view-layer of the visualisation application. The only difference between the configuration form produced by the FormTag-class in the exercise application and the visualisation application is the action triggered by the start-button. The start-button of the exercise application sends a POST-request to the page start.exercise (i.e. the ControlServlet of the exercise application), while the start-button of the visualisation application sends a POST-request to the page startAdvanced.demo. Thus, using the link to the exercise application

1) creates an instance of the ControlServlet-class in the package exercise.control

2) reads the path to the substitution matrices from web.xml

3) reads the list of available substitution matrices from the matrix.properties file, stores the list in a Properties-object, which is stored in a HttpSession

4) redirects the exercise application to the exercise.jsp page, which creates an instance of the FormTag-class that prints the configuration form to the page.

Pressing the start-button created by the FormTag-class sends a POST-request to the ControlServlet. The ControlServlet handles POST-requests with the doPost-method. The doPost-method of the exercise application uses, as the doPost-method of visualisation application, the names of the buttons that triggered the requests to handle a POST-request correctly. The start-button of the exercise (named "init") triggers the method setAttributes. This method is similar (but not identical) to the setAttributes-method of the ControlServlet in the visualisation application. The two differences between the two methods are

1) the setAttributes-method of the ControlServlet in the exercise application does not contain any hard-coded parameter

2) the setAttributes-method of the ControlServlet in the exercise application returns a different page (i.e. computeTable.jsp) to which the exercise application is redirected.

Thus, pressing the start-button produced by the FormTag in the exercise application conducts a pairwise sequence alignment as described in the section concerning the visualisation (section B.3.2.3), that is, it uses an instance of the NeedleBean-class in the demo.model-package to conduct the alignment, stores the necessary information in an HttpSession. The exercise application is then redirected to the page computeTable.jsp.

The page computeTable.jsp is linked to the tag-library exercise.tld found in the exercise-subdirectory of the tld-directory. This tag-library is included in computeTable.jsp with the directive

```
<%@ taglib uri="/WEB-INF/tld/excercise/excercise.tld"
                    prefix="ex" %>
```

In addition to the description of how to perform an exercise, the page computeTable.jsp includes an instance of the class ExerciseTag found in the view-layer of the exercise application (i.e. located in the package exercise.view) with the directive

```
<ex:Init action="inputTable.excercise" />
```

The ExerciseTag-class prints a score matrix based on the parameters entered in the configuration form to the page in which the ExerciseTag instance is included, a link to the selected substitution matrix, and a set of buttons that controls the exercise. Each cell in the score matrix printed by this class contains a text field in which the score is to be entered, and three check boxes that represent the possible pointers. Pressing either of the buttons sends a POST-request to the ControlServlet of the exercise application:

- the submit button triggers the compare-method of the ControlServlet
- the new-button resets the HttpSession, and redirects the application to the page exercise.jsp
- the main page-button resets the HttpSession, and redirects the application to the page index.html

The compare-method creates an instance of the Corrector-class located in the model-layer of the exercise application (i.e. the class is located in the package exercise.model). The compare-method retrieves the scores and directions entered into the score matrix by the users, and compares these scores and directions to the scores and directions of the correct score matrix.

If the score matrix supplemented by the users match the correct score matrix, the method redirects the exercise application to the page correctTable.jsp. correctTable.jsp is linked to the tag-library exercise.tld using the same directive as is used in the page computeTable.jsp, and includes an instance of the CorrectMatrixTag-class by including the directive

```
<ex:showCorrect action="correct.excercise" />
```

The CorrectMatrixTag-class prints the correct score matrix with the optimal path marked with red, and the optimal alignments. The class also prints two buttons to the page the instance is included in:

- the New exercise-button resets the HttpSession, and redirects the exercise application to the page exercise.jsp

- the Main page-button resets the HttpSesssion, and redirects the exercise application to the page index.html

If the submitted score matrix is incorrect, the compare-method creates an error message that describes where in the score matrix the incorrect score or direction occurs. If a score and a direction is incorrect, two error messages is created, the first describes where in the score matrix the incorrect score occurs, and the second error message describes where in the score matrix the incorrect direction occur. The error message (or messages) is stored in the HttpSession, and the exercise application is then returned to the page computeTable.jsp. The ExerciseTag-class then prints a score matrix that contains the scores and directions entered by the users in the previous step of the exercise, the error message (or messages), the link to the substitution matrix, and the set of buttons that controls the exercise. It is then possible to correct the errors and re-submit the score matrix.

B.2.3 The clustering application

The clustering application is a Java Applet and is organised as follows:

<path to clustering application>
|
|-HTML documents
|
|-<classes>
|-java class files
|
|-<gfx>
|-images and graphics

B.2.3.1 The interface components

Selecting the clustering application from the visualisation menu opens the page index.html. This page serves as an introduction to the components of the interface of the clustering application, and provides a link that opens the page cluster.html. This page includes the clustering applet by including the tag

```
<APPLET CODE="ClusterApplet.class" codebase="classes/"
                WIDTH=700 HEIGHT=570>
```

The code-parameter points to the class file that extends the JApplet-class, while the codebase-parameter points to the directory that contains the classes that are used by the applet. The width and height parameters set the width and height of the applet respectively. Both index.html and cluster.html is linked to the style sheet of the portal. Opening the cluster.html page creates an instance of the ClusterApplet-class and initialises the instance.

The initialisation process (handled by the init-method) sets some of the state variables, and sets up the interface of the clustering applet. The applet interface is composed of a main panel, which in turn are composed of two panels; the left panel and the right panel. The left panel contains the coordinate system, the algorithm and variant list, the list of predefined examples, the show-linkage check box, and the canvas upon which the dendrogram of the hierarchical clustering is drawn. The right panel contains the raw data table, the buttons that control the visualisation, and the clustered data table.

The coordinatesystem is composed of a header label that asks the users to plot the data in the coordinate system below, an instance of the CoordinateArea-class, and a label that displays the coordinates of the position of the mouse-pointer when it is situated within the coordinate system. The CoordinateArea-class is a custom component extended from the JComponent-class. This class contains all the logic for depositing points in the coordinate system and for displaying the graphical visualisations of the clustering algorithms, as well as the logic for handling mouse-pointer movements within the coordinate system. Moving the mouse-pointer within the coordinate system triggers the method mouseMoved-method in the CoordinateArea-class. This method transforms the mouse-pointer's position within the applet interface into the mouse-pointer's position relative to the origin of the coordinate system with the method normalizeCoords. The method also triggers the updateCursorLocation in the ClusterApplet class, which updates the label that displays the position of the mouse-pointer in the coordinate system. Pressing a mouse button within the coordinate system triggers the mouseClicked-method, which transforms the mouse-pointer's position in the applet interface into the mouse-pointer's

position relative to the origin of the coordinate system, deposits a point in the coordinate system, creates an instance of the MicroArrayPoint-class, and stores the MicroArrayPoint in the vector that contains all points that have been deposited in the coordinate system. The MicroArrayPoint-class is used to store all useful information about a deposited point such as coordinates of the point in the applet interface, the coordinates of the point relative to the origin of the coordinate system, the number of the point, which cluster and centroid the point belongs to, and so forth.

The mouseClicked-method of the CoordinateArea-class also triggers the updateClickPoint-method in the ClusterApplet-class. The updateClickPoint-method in the ClusteringApplet-class retrieves the vector that contains the deposited points, and adds the last point in the vector to the raw data table. The method also sets some of the state variables of the applet, but as which state variables are set depend on which algorithm is chosen, these state variables are discussed when the different algorithm visualisations are discussed.

The raw data table and the clustered data table are instances of the class JTable, standard components of the java.swing package. These tables are implemented with a custom background renderer in order to colour the background of the cells in the second and third column according to the colour scheme that is used in the heat maps. The custom background renderer is represented by the class ColouredCellBgRenderer-class. This class ensures that the cells in the second and third column of the tables receive the appropriate green or red colour background based on the x- and y-coordinates of a point.

The canvas for the dendrogram is an instance of the DendrogramArea-class, which is an extension of the JComponent-class. This class contains all the logic for drawing a dendrogram on the canvas. This class will be discussed in more detail during the discussion of the visualisation of the hierarchical clustering algorithm.

The buttons that controls the visualisation are:

- the new-button
- the predefined examples-button
- the place centroids-button (only available in the visualisation of the k-means clustering algorithm)

- the stepwise clustering-button
- the epochwise clustering -button (only available in the visualisation of Self Organizing Maps)
- the automated clustering-button
- the finish-button

Pressing any of these buttons triggers the actionPerformed-method in the ClusterApplet-class.

Pressing the new-button resets the applet by resetting the states of the coordinate system and the dendrogram, removing all deposited points from the coordinate system and the dendrogram by calling the clear-method of the CoordinateArea and the DendrogramArea, emptying the raw data table and the clustered data table, displaying the canvas, and resetting the algorithm lists and the buttons.

Pressing the predefined examples-button activates the list of predefined examples. Activating this list empties the canvas, the coordinate system and the clustered data table, deactivates the coordinate system (i.e. the CoordinateArea class will no longer respond to mouse activities in the coordinate system), and creates an instance of the Example-class. The example class generates a set of points distributed in 2, 3, or 4 clusters composed of 50 points each (the number of points in each cluster can be adjusted by supplying a higher or lower number when an instance of the class is created). The instance of the Example-class that is created when the predefined examples list is activated generates a set of points distributed in 2 clusters. The Example-class stores the points in a vector, which is retrieved by the ClusterApplet and sent to the coordinate system by calling the method setPoints in the CoordinateArea class. This method draws the points in the vector in the coordinate system. The ClusterApplet also sends the vector to the raw data table by calling the method setRawData in the ClusterApplet-class.

The stepwise clustering- and automated clustering-button are only available under certain circumstances depending on the chosen algorithm. In general these buttons are available once at least three points have been deposited in the coordinate system. The place centroids-button is only available after the k-means algorithm has been chosen, and at least three points have been deposited in the coordinate system. The epochwise clustering-button is only available after Self Organizing Maps have been chosen and at least three points have been deposited in the coordinate system. The finish-button is only available after a visualisation has been started.

The finish-button is only available once a visualisation is started. Pressing this button triggers the actionPerformed-method of the ClusterApplet-class, which finishes a visualisation, and resets the clustering application.

B.2.3.2 Running a visualisation of the hierarchical clustering algorithm

Selecting the hierarchical clustering from the algorithm list triggers the actionPerformed-method of the ClusterApplet-class, which empties the canvas by calling the clear-method of the DendrogramArea-class, removes any lines that have been drawn in the coordinate system by calling the removeLines-method of the CoordinateArea-class, sets the state of the clustering application to hierarchical clustering, empties the clustered data table, populates the list of algorithm variants to display the implemented linkage options, enables the automated-button and the stepwise-button if three or more points are plotted in the coordinate system (the buttons are disabled otherwise), disables the epochwise-button, displays the canvas, and enables the show linkage-check box.

A visualisation of the hierarchical clustering algorithm is started by depositing at least 3 points in the coordinate system and pressing either the automated clustering-button or the stepwise clustering-button. Pressing either of these buttons triggers the actionPerformed-method of the ClusterApplet-class, but the two buttons are handled differently.

Pressing the automated clustering-button disables all buttons and components except the finish-button. The vector containing the points deposited in the coordinate system is retrieved by calling the getPoints-method of the CoordinateArea-class, and converted into an array. The array containing the deposited points is sent to the canvas by calling the setBoundingBox-method of the DendrogramArea-class. This method places the points on the canvas with equal distance between them. The automated clustering is then started by calling the startAutomated-method of the ClusterApplet-class. This method empties the clustered data table, creates an instance of the ClusteringAnimator-class, and calls the start-method of the ClusteringAnimator-class. The ClusteringAnimator-class implements a timer that is set to trigger its own actionPerformed-method every 2 seconds for the hierarchical clustering visualisation that does not show the linkage, and every 1 second if the linkage is to be shown. The timer is started by calling the start-method of the ClusteringAnimator-class, and is stopped by calling the stop-method. An instance of the HierarchicalClustering-class is then created. This class contains all the logic that is needed to cluster a set of points according to the hierarchical clustering algorithm. Creating an instance of this class transforms the points in the point-vector into instances of the Cluster-class. These instances are stored in a cluster-vector.

Each time the actionPerformed-method of the ClusteringAnimator-class is triggered, the cluster-method of the HierarchicalClustering-class is called. The cluster-method computes the distance between each cluster (singleton or non-singleton) according to the chosen linkage, and finds the two clusters that are nearest each other. The cluster pair is removed from the vector that contains the clusters, combined in a larger cluster, and the new cluster is inserted at the end of the cluster-vector. The actionPerformed-method then retrieves the modified vector of clusters from the HierarchicalClustering-class and sends the vector to the coordinate system by calling the method setCurrentClusters of the CoordinateArea-class. The repaint method of the CoordinateArea-class is then called, which calls the drawHull-method. The drawHull-method of the CoordinateArea-class uses the Graham's Scan algorithm to draw the convex hulls of each cluster in the cluster-vector. The actionPerformed-method also sends the cluster-vector to the canvas by calling the setClusterVector-method of the DendrogramArea-class. The repaint-method of the DendrogramArea-class is then called. This method draws the dendrogram on the canvas. Lastly, the actionPerformed-method calls the showClustered-method of the ClusterApplet-class, which rearranges the clustered data table to display the points in the order that they are clustered.

The procedure of visualising the hierarchical clustering algorithm if the show linkage-box is ticked off is somewhat different. Ticking off this box instructs the clustering application to show the pair of points that defines the least distance between a pair of clusters. In order to do so, the actionPerformed-method of the ClusteringAnimator-class retrieves the vector that contains the clusters from the previous step and sends the vector to the coordinate system by calling the method setPreviousClusters in the CoordinateArea-class. The cluster-method of the HierarchicalClustering-class is then called, and the actionPerformed-method then retrieves the new cluster-vector and sends it to the coordinate system by calling the setCurrentClusters-method of the CoordinateArea-class. The variable stepLinkage is set to 1, and the variable linkageDrawn in the CoordinateArea-class is set to false before the repaint method of the coordinateArea is called. By setting the linkageDrawn-variable to false, the repaint-method of the CoordinateArea-class draws the convex hulls of the previous clusters and a blue line between the pair of points that define the least distance. The vector containing the current clusters is also sent to the canvas, and the repaint-method of the DendrogramArea-class is called to draw the dendrogram on the canvas.

The convex hulls of the current clusters are drawn in the coordinate system the next time the actionPerformed-method is called. The method checks if the stepLinkage-variable is set to 1, sets the linkageDrawn-variable of the CoordinateArea-class to true, and the linkageStep-variable to zero before the

repaint method of the CoordinateArea-class is called. Setting the linkageDrawn-variable to true causes the repaint-method of the CoordinateArea-class to replace the convex hulls of the previous cluster-vector and the blue line between the pair of points that define the least distance with the convex hulls of the current cluster-vector.

The cycle of calling the cluster-method, and repainting the coordinate system and the dendrogram is repeated until all clusters are combined into one large cluster, or until the finish-button is pressed.

A stepwise visualisation of the Hierarchical clustering is identical to the automated clustering visualisation with one exception. The timer that triggers the actionPerformed-method in the ClusteringAnimator-class is replaced by the users. Pressing the stepwise clustering-button triggers the actionPerformed-method in the ClusterApplet-class, which checks if the button has been pressed to start a stepwise visualisation, or if the button has been pressed to display the next step in the visualisation (the stepwise-button is used both as a start-button and as a next-button in the stepwise visualisation). If the button has been pressed to start a stepwise visualisation, the actionPerformed-method deactivates all other buttons and components, and activates the finish-button. The coordinate-system is also deactivated so that it is impossible to deposit new points in the coordinate system during the visualisation, and the canvas is cleared of any dendrogram. The label of the stepwise clustering-button is changed so that the button shows the text “next” (i.e. the stepwise-button becomes the next-button). The actionPerformed-method then calls the startStepping-method. This method creates an instance of the HierarchicalClustering-class, runs the cluster-method of the HierarchicalClustering-class, repaints the coordinate system and the canvas with the appropriate clusters and dendrogram, and rearranges the points in the clustered datatable. If the show linkage box has been ticked off, the method also creates an instance of the StepAnimator-class. This class enables the blue line between the pair of points that define the least distance between a pair of clusters to be drawn.

Pressing the next-button triggers the actionPerformed-method of the ClusterApplet-class, which calls the step-method. This method runs the cluster-method of the HierarchicalClustering-class, repaints the coordinate system and the canvas, and rearranges the points in the clustered data table.

Once the visualisation is completet, automated or stepwise, the buttons are reset, and the components are reactivated.

B.2.3.3 Running a visualisation of the k-means clustering algorithm

Selecting the k-means algorithm from the algorithm list triggers the actionPerformed-method of the ClusterApplet-class. This method sets the state of the clustering application to k-means-visualisation, removes the canvas from the interface, changes the automated clustering-button to display the text "Place centroids" (i.e. the automated clustering-button becomes the place centroids-button), disables the show linkage-box, and populates the variant list with the options batch-variant and online-variant.

Depositing three or more points in the coordinate system activates the place centroids-button. Pressing this button updates the state of the clustering application to allow the centroids to be placed in the coordinate system, and changes the text displayed in the place centroids-button to "Automated clustering" (i.e. the place centroids-button becomes the automated clustering-button). The coordinate system is now set up to handle points deposited in the coordinate system as centroids, that is, the deposited points are coloured and labelled C1, C2, C3, etc. and placed in a vector that contains the centroids. Placing at least two centroids enables the automated clustering-button and the stepwise-button. Pressing either of these buttons starts a visualisation of the k-means clustering algorithm, and deactivates all other buttons and components except the finish-button.

Pressing the automated clustering-button triggers the actionPerformed-method in the ClusterApplet-class, which deactivates all components except the finish-button, and calls the startAutomated-method. This method retrieves the vector that contains the deposited points and the vector that contains the deposited centroids from the coordinate system, retrieves the chosen variant from the variant list and calls the startAutomated-method of the ClusterApplet-class. This method creates an instance of the ClusteringAnimator-class, and calls the start-method of this class. Creating an instance of the ClusteringAnimator-class for the k-means clustering visualisation sets the ClusteringAnimator up to display a k-means visualisation, creates an instance of the Timer-class and sets the delay between the steps to 1 second for the batch variant and zero for the online variant, and creates an instance of the KmeansClustering-class. The KmeansClustering-class contains all the logic for performing a k-means batch clustering and a k-means online clustering. The start-method of the ClusteringAnimator-class starts the timer, which triggers the actionPerformed method of the ClusteringAnimator-class with the given delay until the stop-method is called.

Running an automated visualisation of the batch variant of the k-means clustering algorithm causes the actionPerformed-method of the ClusteringAnimator to call the cluster-method of the KmeansClustering-class

with the parameter -1. Supplying the -1 parameter to this method means that the cluster-method clusters the dataset according to the batch-variant. The first step in the visualisation of the batch variant is to assign each point to the closest centroid. Thus, the cluster-method of the KmeansClustering-class calls the method assignPoints. This method iterates over the deposited points and assigns each point to the closest centroid. The method also sets the colour of each point to the same colour as the centroid the point is assigned to. The actionPerformed-method of the ClusteringAnimator-class then retrieves the vector that contains the centroids from the KmeansClustering-class and sends them to the coordinate system by calling the method setCentroids of the CoordinateArea-class. The actionPerformed-method also instructs the CoordinateArea-class to draw the deposited points with the colours that corresponds to the colour of the centroid each point is assigned to by calling the method setUsePointColour of the CoordinateArea-clas with the parameter true. The actionPerformed-method then calls the repaint-method of the CoordinateArea, which draws the deposited points in the colour that corresponds to the colour of the centroid each point is assigned to. Lastly, the actionPerformed-method calls the showClustered-method of the ClusterApplet-class, which updates the clustered data table to show the deposited points in the clustered order. The backgrounds of the cells in the first column of the clustered data table are also coloured with the same colour as the centroid each point is assigned to.

The next time the cluster-method is called by the actionPerformed-method of the ClusteringAnimator-class, the cluster-method calls the method setCentroidCoord. This method recalculates the position of the centroids. The method then reassigns the points to the closest centroid. The actionPerformed-method then retrieves the vector that contains the centroids and sends them to the coordinate system by calling the setCentroids-method of the CoordinateArea-class, before the repaint-method of the CoordinateArea-class is called. The CoordinateArea-class then draws the centroids in their new positions, and updates the colour of the points to correspond to the colour of the closest centroid. Lastly, the clustered data table is updated in the same way as in the first step.

This cycle is repeated until the points remain stable (i.e. the points that are assigned to each centroid remain the same in two consecutive cycles). Pressing the finish-button before the clustering is completed causes the clustering application to complete the clustering.

Running an automated visualisation of the k-means online algorithm is slightly different. The actionPerformed-method iterates over the deposited points, and calls the cluster-method of the KmeansClustering-class for one

point at a time. Instead of using -1 as a parameter in the call to the cluster-method, the parameter is the index of the current point in the vector of deposited points. The cluster-method then calls the adjustCentroid-method, which in turn finds the centroid that is closest to the current point and moves the closest centroid somewhat closer to the point. The actionPerformed-method of the ClusteringAnimator-class then retrieves the vector containing the repositioned cluster, and sends it to the coordinate system by calling the setCentroids-method of the CoordinateArea. The position of the centroid that is closest to the current point in the coordinate system is then updated by calling the repaint-method of the CooridinateArea-class. The actionPerformed-method also updates the clustered data table to show the points in the order they are clustered (i.e. calls the showClustered-method of the ClusterApplet-class). This cycle is repeated until the points remain stable and the centroids stop moving, or until the finish-button is pressed.

Conducting a stepwise visualisation of the k-means algorithm is identical to the automated clustering with the exception that the instance of the ClusteringAnimator is replaced by the users. Pressing the stepwise clustering-button triggers the actionPerformed-method of the ClusterApplet-class, which changes the state of the clustering application to that of a k-means clustering, transforms the stepwise clustering-button into the next-button, deactivates all components except the next-button and the finish-button, and calls the startStepping-method. The startStepping-method creates an instance of the KmeansClustering-class, and runs the cluster-method of this class. This method performs the same actions as in the automated clustering. The startStepping-method then retrieves the vector containing the adjusted centroids and sends it to the coordinate system. The coordinate system is updated in the same manner as in the automated clustering, as is the clustered data table. Pressing the next-button triggers the actionPerformed-method of the ClusterApplet-class, which calls the step-method. This method repeats the cycle of adjusting the positions of the centroids and the updating of the clustered data table. This procedure can be repeated until the clustering is completed (see above), or the clustering can be completed by pressing the finish-button.

B.2.3.4 Running a visualisation of Self Organizing Maps

Selecting SOM (Self Organizing Maps) from the list of algorithms triggers the actionPerformed-method of the ClusterApplet-class, which sets the state of the clustering application to a SOM-visualisation, removes the canvas, deactivates the show linkage-box, and populates the list of variants with the options grid, and line.

Depositing at least three points in the coordinate system activates the automated clustering-button, the stepwise clustering-button, and the epochwise clustering-button.

Pressing the automated clustering-button triggers the actionPerformed, which deactivates all components except the finish-button and calls the startAutomated-method. This method creates an instance of the ClusteringAnimator-class, and calls the start-method of this class. Creating an instance of the ClusteringAnimator-class sets up the timer in accordance to the SOM-visualisation, and creates an instance of the SelfOrganizing-class. This class contains all the logic that is needed to conduct a clustering using SOMs.

Creating an instance of the SelfOrganizing-class creates the grid or the line of units that is used in the clustering. The grid and the line of units are composed of 1/3 as many units as there are deposited points. The number of units in the grid is adjusted so that the grid is always a square of rectangle. The units are generated randomly around the origin of the coordinate system, and organised in neighbourhoods (see section 5.5 of chapter 5).

Calling the start method of the ClusteringAnimator-class retrieves the grid or line of units, which is sent to the coordinate system by calling the setCentroids-method of the CoordinateArea-class. The repaint-method of the CoordinateArea is then called to draw the grid or line of units in the coordinate system. The start-method then starts the timer, which triggers the actionPerformed-method. The actionPerformed-method iterates over the deposited points, and calls the adjustCentroids-method of the SelfOrganizing-class for each deposited point. The adjustCentroids-method moves the nearest unit and its neighbour(s) somewhat closer to the point. The actionPerformed-method of the ClusteringAnimator-class then retrieves the centroid-vector from the SelfOrganizing-class and sends it to the coordinate system by calling the method setCentroids of the CoordinateArea-class. The actionPerformed-method then calls the repaint-method of the CoordinateArea, which draws the adjusted grid or line of units in the coordinate system. This process is repeated for each deposited point 3000 times, or until the finish-button is pressed.

Pressing the stepwise-button triggers the actionPerformed-method of the ClusterApplet-class, which sets the state of the clustering application up for a SOM-visualisation, transforms the stepwise clustering-button into the next-button, deactivates all components but the next-button and the finish-button, and calls the startStepping-method. This method creates an instance of the SelfOrganising-class, which generates the grid or line of units, retrieves the grid or line, sends the vector containing the grid or line to the coordinate system, and calls the repaint-method of the CoordinateArea-class. This method then draws the grid or line of units in the coordinate system. Pressing

the next-button triggers the actionPerformed-method of the ClusterApplet-class, which calls the step-method. This method calls the adjustCentroids-method of the SelfOrganizing with the current point as a parameter, which adjusts the position of the unit closest to the current point. The vector containing the adjusted unit is then retrieved and sent to the coordinate system by calling the setCentroids-method of the CoordinateArea-class. The repaint-method of the CoordinateArea-class is then called to draw the adjusted grid or line in the coordinate system. This procedure can be repeated for each deposited point 3000 times by pressing the next-button, or the clustering can be completed by pressing the finish-button.

Pressing the epochwise clustering-button triggers the actionPerformed-method of the ClusterApplet-class, which sets the state of the clustering application up for a SOM-visualisation, transforms the epochwise clustering-button into the next-button, deactivates all components but the next-button and the finish-button, and calls the startEpochStepping-method. This method creates an instance of the SelfOrganizing-class, retrieves the generated grid or line of units and sends it to the coordinate system, and calls the repaint-method of the CoordinateArea-class to draw the grid or line in the coordinate system.

Pressing the next-button triggers the actionPerformed-method in the ClusterApplet-class, which calls the method epochStep. This method iterates over the deposited points and calls the adjustCentroid-method of the SelfOrganizing-class to adjust the grid or line of units. The method retrieves the adjusted grid or line once all the deposited points have adjusted a unit once (i.e. one full iteration over the deposited points, or an epoch), and sends the adjusted grid or line to the coordinate system by calling the setCentroids-method of the CoordinateArea. The adjusted grid or line is then drawn in the coordinate system by calling the repaint-method of the CoordinateArea-class. This process can be repeated 3000 times by pressing the next-button, or the clustering can be completed by pressing the finish-button.

The interface of the clustering application is reset once the visualisation (SOM or any other visualisation). Resetting the interface includes reactivating the components, and resetting the state of the application.