MAVIS: VISUALIZATION TOOL FOR MULTIPLE SEQUENCE ALIGNMENTS

A Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Lei Zhao

May 2011

MAVIS: VISUALIZATION TOOL FOR MULTIPLE SEQUENCE ALIGNMENTS

Lei Zhao

APPROVED:

Dr. Yuriy Fofanov, Chairman
Department of Computer Science

Dr. Dan Graur
Department of Biology and Biochemistry

Dr. Shishir Shah
Department of Computer Science

Dean, College of Natural Sciences and Mathematics

Acknowledgements

(Under construction)

MAVIS: VISUALIZATION TOOL FOR MULTIPLE SEQUENCE ALIGNMENTS

An Abstract of a Thesis

Presented to

the Faculty of the Department of Computer Science

University of Houston

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

By

Lei Zhao

May 2011

Abstract

Biological sequences like DNA, RNA, and protein are frequently analyzed in the field of molecular biology and bioinformatics. Multiple sequence alignments (MSAs) help identify similarity between the sequences and reveal their functional, structural, or evolutionary relationships. MSA analysis often requires various visualization techniques, which assist researchers to better understand, evaluate, and learn from the MSA results. One of the common techniques is the coloring of the MSA. There are several coloring tools available, but their color schemes focus more on chemical properties of the sequence residues, rather than the actual structural or evolutionary information of the whole alignment.

In this thesis, we introduce Mavis, a new approach to coloring MSAs and highlighting its quality and structure. Instead of using a pre-defined color scheme based on residue types, we designed a new algorithm to dynamically generate a color for each residue based on a user-determined similarity score between this residue and others. This new tool colors an MSA in such a way that a well aligned region is represented by a solid-color block, while a poorly aligned one by a mosaic with various colors. Thus, the alignment quality and internal structure are clearly displayed.

We implemented the core Mavis algorithm in R, the application programming interface (API) in Perl CGI, and the graphical user interface (GUI) in HTML and

JavaScript that can be viewed over web browsers. Other GUI implementations are also available for mobile devices like iPhone and iPad. We also followed common software design principles, and the system is extensible for even more add-on features and new user interfaces.

Mavis is ready-to-use for any biologists without requiring deep computer science skills and will deliver the visualization result of a typical MSA in minutes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

Biological sequences, including nucleotide sequences (DNA and RNA) and peptide sequences (proteins), play an fundamental role in life and reproduction. A nucleotide sequence consists of a chain of linked nucleotide bases. There are four possible types of bases in a DNA sequence, represented by four letters A, T, G, and C. A peptide sequence consists of linked amino acids, which have 20 different possible types. Figure 1.1 and 1.2 shows their notation rules recommended by the International Union of Pure and Applied Chemistry (IUPAC) [1].

Since the 1970s, sequences of hundreds of thousands of organisms have been decoded and analyzed by researchers [2]. With the explosively growing amount of data, it is impractical to study sequences manually. Computer-aided techniques are

Table 1.1: IUPAC nucleotide notation rules.

IUPAC Code	Base
A	Adenine
C	Cytosine
G	Guanine
T (or U)	Thymine (or Uracil)
N	any base
. or -	gap

needed to help analyze their structures, functions and evolution.

Multiple sequence alignment (MSA) is one of the computational ways to identify regions of similarity between sequences and is required by almost all comparative sequence studies. An MSA is typically formed as a matrix, in which each row represents a sequence, and each column corresponds to an hypothetically equivalent position across all the sequences aligned. Each element of the matrix contains a symbol, which can be either '-' (a gap) or a sequence symbol (an amino acid for DNA/RNA sequences, or a nucleotide base for protein sequences) [3].

```
AY169803.O  MHYRDLSTLIIVSALLLINVXLWMFILRXYLEHKRQERREREILERLRRIREIKDDSDYESNGEEEQEVMD-LVHSHGFDNPMFEL
AY169809.O  MQYKGL--LLIIIALLLINVXVWMFNLRKYLEQKKQERREREVINRLRRIREVKDDSDYESNGEEEQEVME-LVHSHGFDNPMFEL
AY169807.O  MLHRDLLLLIIISALLLTNIILWMFVLRKYLEIKKQERREREILERLRXIREIRDDSDYESNEEEEQEVRDHLVHTFGFANPMFEI
AY169811.O  MHHRDLLTLIAVSALLFINIILWIYVLRKYLEQRKQDRREREILERLRRIXEIGDDSDYESNEEEEQEVMD-LVHNHGFDNPMFEP
AJ302646.O  MHHRDLLALITTSALLLTNVVLWTFILRQYLKQKKQDKREREILERLRRIRQIEDDSDYESDGTEEQEVRD-LVHSYGFDNPMFEL
AY169815.O  MQHKDLLILIITSALLLINVILWLFVLKQCLEQKKQTKREREIIRRLRRIREIEDDSDYESNGEEEQTVRD-LIHSHGFDNPMFEL
AY169804.O  MHQRDLLILIAVSILCLICILVWTFNLRKYLEHRKQDKREREILERLRRVREIRDDSDYESBGEEEQEVMD-LIHSHGFANPLFEL
AY169810.O  MNYKELLSLIVVSVLLLAAIVIWMFILKKYLEQKEQDRREREILLKRIERLXEXRDDSDYESNGDEEQEVMH-LVHTHGFANPMFEL
AY169806.O  MYKDQIILIIIFCVVFLIAACIWLFILKTYLEQKKQDRREKELLRRLQRIIEIRDDSDYESNGEEEQEVMD-LVHEHGFVNPMFEL
```

Figure 1.1: An example of multiple sequence alignment, generated with *ClustalW* [4]. Each row in the MSA represents one protein sequence. The leftmost column shows sequence names. The right part is the alignment matrix.

An MSA result can be a large and complex matrix. To highlight similarity or other properties, certain visualization methods are used. Some alignment programs, like

Table 1.2: IUPAC amino acid notation rules.

IUPAC Code	Three Letter Code	Amino acid
A	Ala	Alanine
C	Cys	Cysteine
D	Asp	Aspartic Acid
E	Glu	Glutamic Acid
F	Phe	Phenylalanine
G	Gly	Glycine
H	His	Histidine
I	Ile	Isoleucine
K	Lys	Lysine
L	Leu	Leucine
M	Met	Methionine
N	Asn	Asparagine
P	Pro	Proline
Q	Gln	Glutamine
R	Arg	Arginine
S	Ser	Serine
T	Thr	Threonine
V	Val	Valine
W	Trp	Tryptophan
Y	Tyr	Tyrosine

ClustalW, add an asterisk, colon, semicolon, or other symbols, to show conservative columns. Some others use color to display information: assign each nucleotide or amino acid its own color, and indicate amino acid properties with an empirical color assignment. Figure 1.2 shows a protein MSA colored using Jalview [5].

1.2 Related Work

Numerous MSA visualization applications, either stand-alone or web-based, have been developed in recent decades, ranging from simple MSA viewers, to complex editing and analysis platforms. Many of them have graphical user interfaces showing

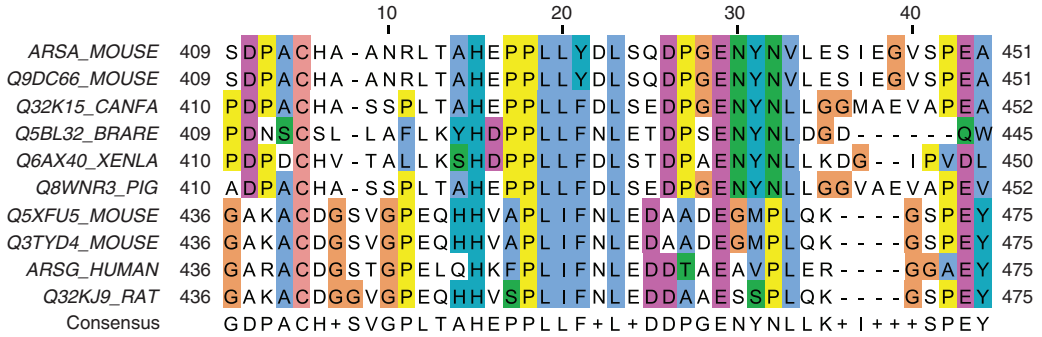


Figure 1.2: A protein MSA colored with Jalview [5, 6].

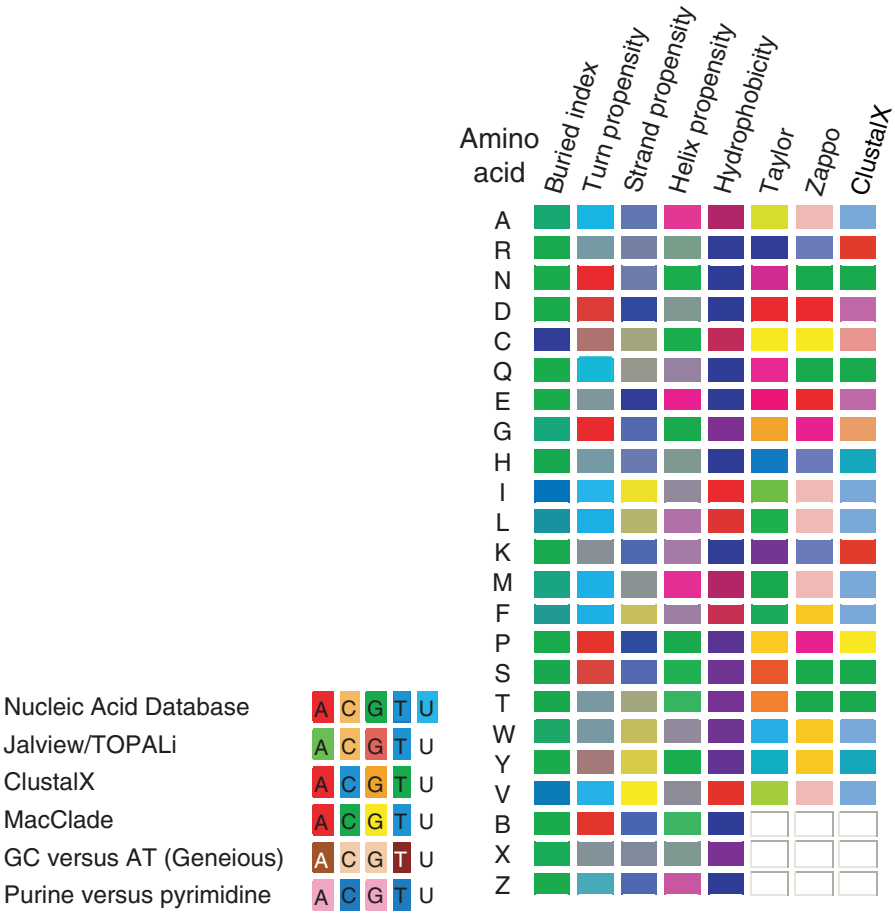
a grid of symbols, usually along with various colorings, sizings and annotations [6].

Coloring an alignment has been widely implemented to highlight specific regions, properties, or patterns. The simplest way of coloring is to use a fixed color scheme, in which each sequence symbol has its own color, and will not be changed across the alignment. The assignment of color schemes are usually based on some empirical properties or chemical classifications. Figure 1.3 shows some nucleotide and amino acid color schemes used by major visualization tools.

There are numerous color schemes available for different purposes, but Taylor [7] and Clustal [4] are widely used and often considered *de facto* standards. Figure 1.4 is Taylor's Venn diagram of color scheme.

Using these pre-determined color schemes to color sequence symbols, however, is not always helpful for phylogenetic analysis. Those fixed color schemes reflect chemical differences between specific types of symbols, rather than between sequences, regions, blocks, or structures of the alignment.

When symbols are examined within one column, the colors successfully show the



(a) Some nucleotide color schemes. **(b)** Some amino acid color schemes.

Figure 1.3: Examples of color schemes used by some visualization tools [6].

similarities. But in the horizontal aspect, comparative information within a row or a rectangular region are unable to be displayed, such as how one sequence is similar to others, which sections align better than others, which areas of the alignment are more conserved, or how robust the alignment accuracy is.

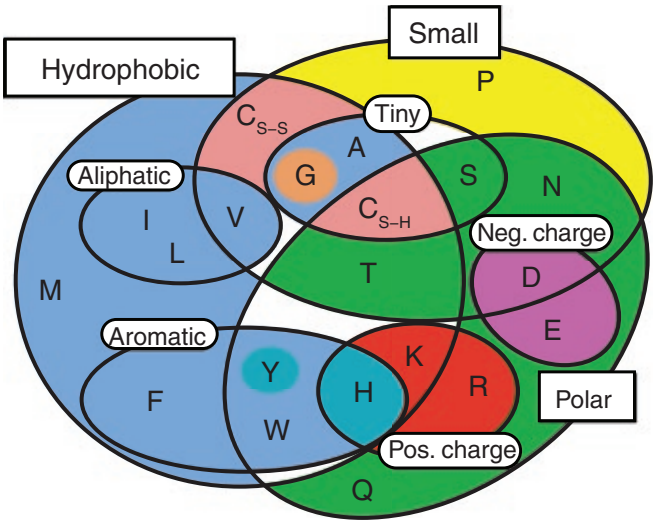
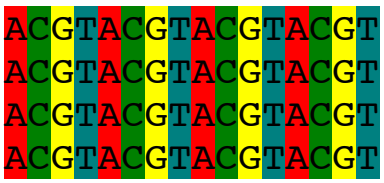


Figure 1.4: Taylor's [7] Venn diagram showing the color scheme based on the physico-chemical properties of the amino acid groups [6].

1.3 Thesis Outline

In this report, we introduce a new MSA visualization tool called *Mavis* (Multiple Alignment VISualization). Mavis focuses on highlighting evolutionary relationships and internal structures of an alignment. Instead of relying on chemical properties or any other information from specific symbol types, our approach is based on only one metric, which is the symbol-symbol similarity score. The color of a sequence residue will not depend on its symbol type, but on its similarity scores with other residues. By using this new approach, we try to provide a whole picture of an MSA, in which the structure and quality are clearly displayed.

The rest part of this thesis is organized into five chapters. Chapter 2 explains the general idea of how Mavis is designed and the steps it takes to color an MSA



(a) A perfectly aligned MSA. However, the color graph is distractive and fails to convey a 'perfect' feeling.

(b) This is an extremely poorly aligned one, but the color graph looks even better than (a).

Figure 1.5: Fixed color scheme introduces confusion.

on a high level. Chapter 3 expounds the core algorithm step by step, from multidimensional scaling, to color mapping, and to optimization. Chapter 4 describes the implementation of the algorithm, including the back-end, API, and different user interfaces. Chapter 5 presents some test results to show the validity and performance of our implementation. The whole work is discussed and concluded in Chapter 6.

The source code of Mavis back-end, API, and web interface is available in the Appendix at the end of this report.

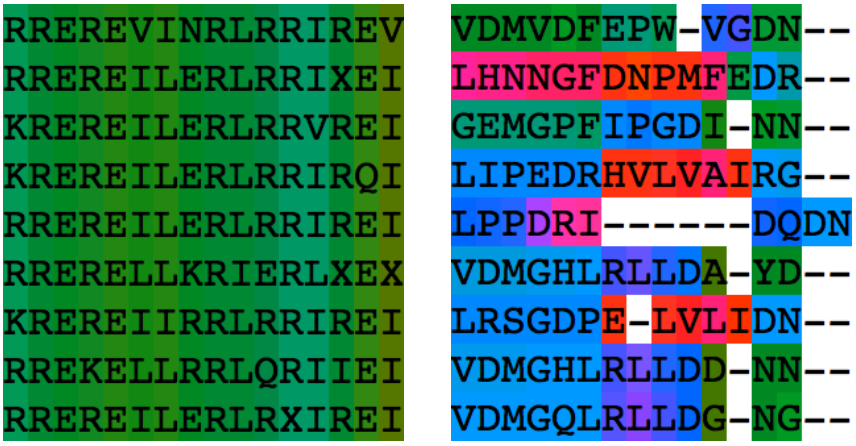
Chapter 2

Approach

2.1 Main Objective

In this thesis report, we address a novel method to visualize multiple sequence alignments using colors. The main objective is to highlight sequence similarity (or alignment confidence) with color similarity.

In the alignment matrix, a well-aligned area should be represented by a solid-color shape, while a poorly-aligned one should become a mosaic of different colors. All the alignment-level information, such as evolutionary relationships, phylogenetic characteristics and alignment accuracy, should be revealed by the color pattern of the graph. Figure 2.1 compares two types of MSA regions and their coloring results.



(a) A well-aligned MSA region. **(b)** A poorly-aligned MSA region.

Figure 2.1: Highlighting sequence similarity with color similarity.

2.2 Data Input

Figure 2.1(a) shows a well-aligned MSA region, which contains many different types of amino acids. In order to demonstrate the good alignment quality, however, these amino acids need to be painted with similar colors. Obviously, the color assignment could not depend only on their types.

Unlike most traditional coloring approaches, we use an set of pairwise scores instead. Within each column of an MSA, all sequence residues are considered to be aligned together at the same equivalent position. We employ a third-party tool to examine each pair of these residues, and estimate the confidence that they should be aligned together. This confidence estimation can also be interpreted as a similarity score between the two residues.

Every pair of residues within a same column has a similarity score. These scores are used to determine how similar the residue colors should appear to each other. In

a good-quality MSA region, all residues are confidently aligned together, therefore the colors would be almost the same. While in a bad-quality region, many alignments are doubtable, resulting in many different colors.

One good way to generate this similarity score data set is by using GUIDANCE [8, 9], which is a tool for quantifying alignment uncertainty. GUIDANCE produces a confidence score in range of $[0, 1]$, named GUIDANCE score, for each symbol, symbol pair, column and sequence in the alignment. We take the symbol-pair GUIDANCE confidence scores as the data input to Mavis.

2.3 Color Generating

The colors of the symbols are generated in such way that the color similarity represents the residue alignment confidence. We have four steps to accomplish this goal.

First, the confidence or similarity score is thought reversely, as a distance or dissimilarity score. The largest confidence score is converted into a smallest distance, and vice versa.

Second, from all these distances, an $N * N$ distance matrix is constructed, where N is the number of residues in the same column.

Third, from the distance matrix, the N residues are mapped as N points in a space of $N-1$ dimensions, such that for each pair of points, their Euclidean distance is equal to the value in the distance matrix.

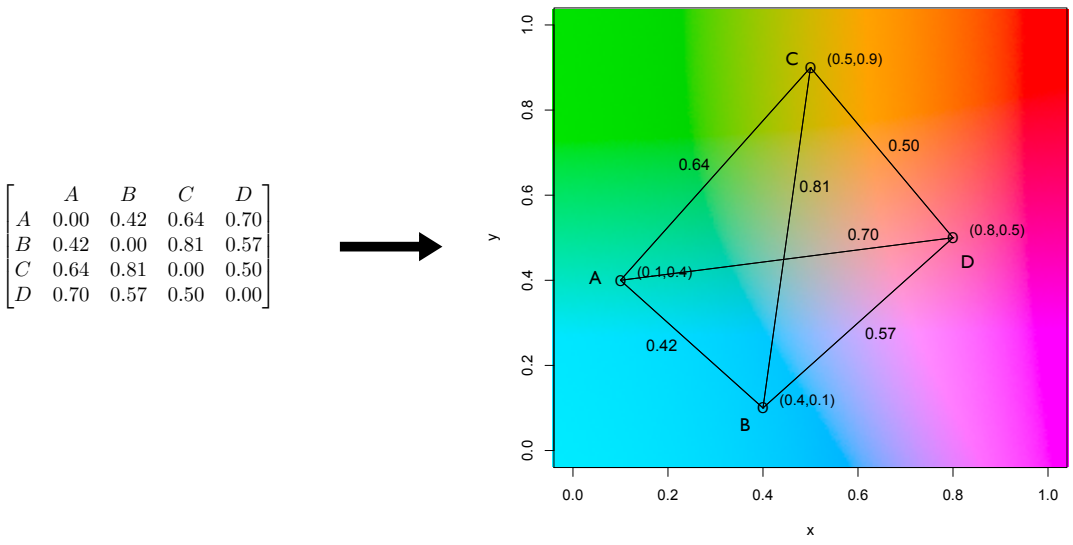


Figure 2.2: A distance matrix is converted into points on a two-dimensional color space. The distances between points are approximately preserved. The color where a point is located is assigned to the corresponding alignment residue.

Finally, we map this space onto a color space, to find a color for each point. In an appropriate color space, the Euclidean distance approximately represents the color difference, so that our alignment confidence is linked with color similarity.

One problem is, most color spaces are three-dimensional or under. In order to map the original space to a color space, it is necessary to reduce the dimension from $N - 1$ to no more than 3, while the distances are still preserved as much as possible. This dimension reduction process is done by using a statistical technique called *classical multidimensional scaling* [10].

2.4 Color Optimization

Up to now, our approach only operates on the column level, and does not consider the relationship between columns. The only way to create a solid-color well-aligned region, is to take cross-column information into account. Since there is no confidence score for two residues from different columns, the color difference between them does not mean anything, and should be eliminated as much as we can. In other words, the color graph needs to be as smooth as possible in the horizontal direction.

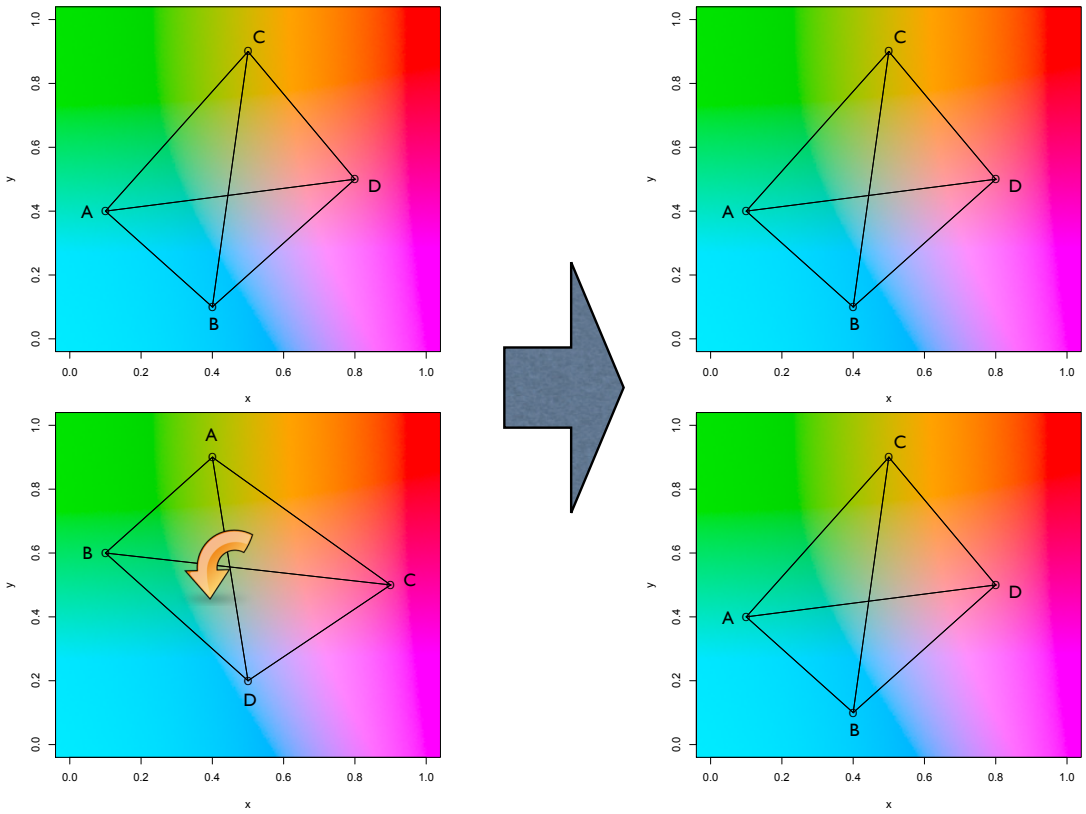


Figure 2.3: An example of color optimization by rotating.

A useful feature is, all the points in the color space can be arbitrarily rotated and/or flipped, without changing the distance between any pair of symbols within a column. For instance, all reds and all greens are virtually identical in the sense of distances and quality of alignment. Converting them into the same color pattern will better support this idea and remove unnecessary color noises.

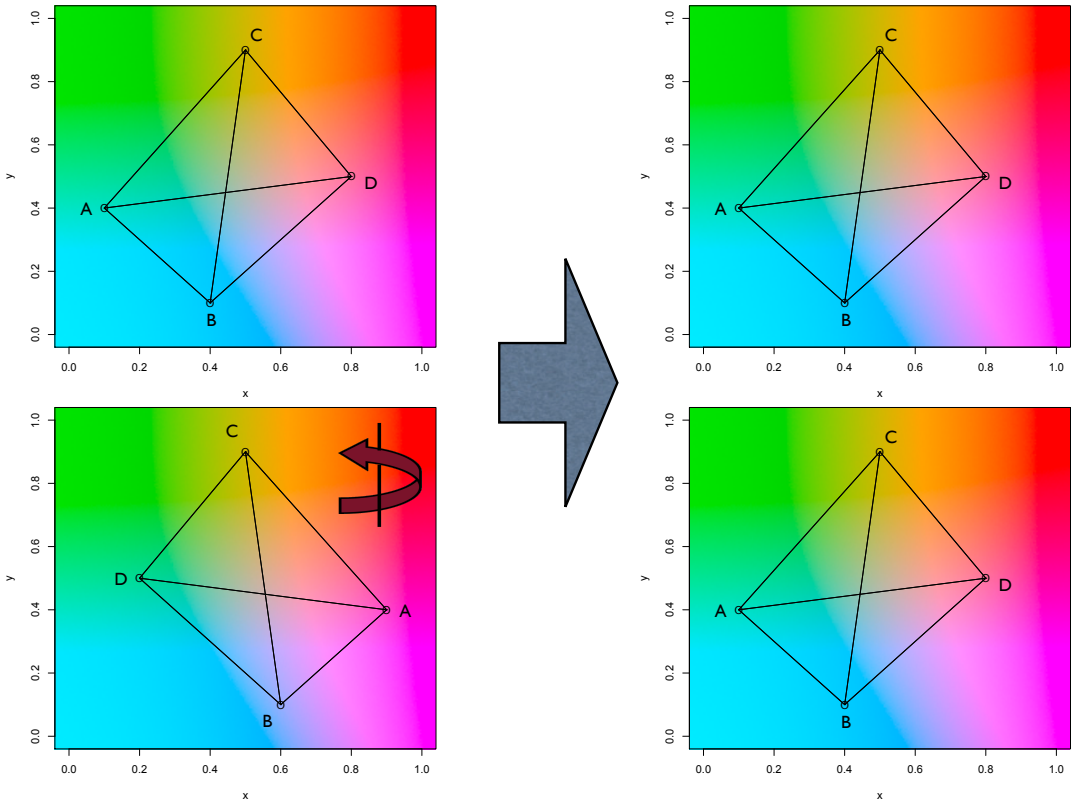


Figure 2.4: An example of color optimization by flipping.

To optimize our color assignment, we define a penalty function to quantify the overall noise level of an alignment. This function calculates the color variance within each row and sum them up. By rotating and flipping, the ideal color assignment

should result in the lowest penalty value.

For larger MSAs, finding the exact optimum may become time-consuming and sometimes impractical. Sub-optimal results are also acceptable for visualization purpose. In Mavis, we use an approximate solution to speed up the optimization procedure.

2.5 MSA Visualization

Mavis creates a colored matrix for an MSA. Users may choose to display residue symbols or not. The user interface is available in a web browser, in order to achieve better flexibility and portability. Other user interfaces are also provided for devices like mobile phones and tablets.

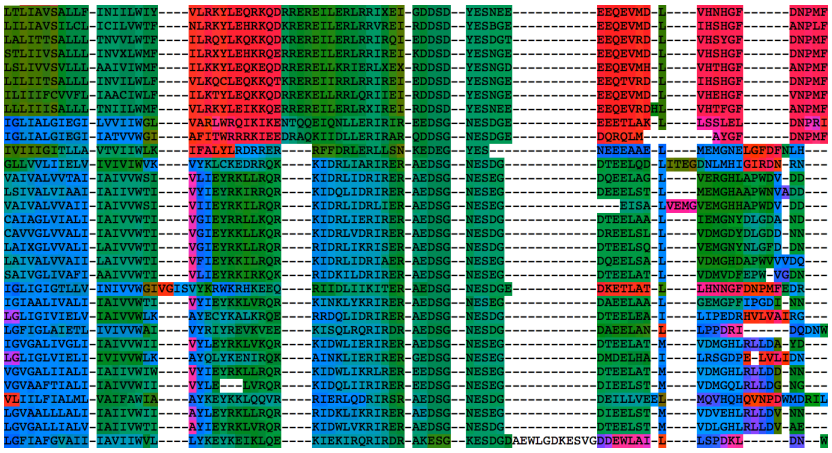


Figure 2.5: An example of MSA coloring result.

An API between the front-end and the back-end is designed to make the system extensible for more add-ons and user interfaces.

Chapter 3

Algorithms

Mavis algorithm of color computation consists of four stages. First, multidimensional scaling reduces the number of dimensions and convert the confidence/similarity scores into coordinates. Second, coordinates are mapped onto part of the *CIE Lab color space* [11], and colors are created. Third, colors are flipped and rotated to smooth the graph and remove color noises. Finally, calculate the average color/hue of each sequence, sort them, and output for further display.

We will discuss all these stages step by step in this chapter.

3.1 Multidimensional Scaling

This stage begins with the confidence scores described above. Each column in the MSA has a complete set C of pairwise scores. Each pair of non-gap symbols i, j in

this column has a score $c_{i,j} = c_{j,i} \in [0, 1]$. Here 1 stands for absolute confidence or similarity, while 0 for the opposite.

This dataset can be easily converted to a set D of dissimilarities or distances between two symbols, by

$$d_{i,j} = d_{j,i} = 1 - c_{i,j}$$

These distance values are re-organized as an $n \times n$ symmetric distance matrix M

$$\begin{array}{cccccc} & 1 & 2 & 3 & ... & n \\ 1 & 0 & d_{1,2} & d_{1,3} & \cdots & d_{1,n} \\ 2 & d_{2,1} & 0 & d_{2,3} & \cdots & d_{2,n} \\ 3 & d_{3,1} & d_{3,2} & 0 & \cdots & d_{3,n} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ n & d_{n,1} & d_{n,2} & d_{n,3} & \cdots & 0 \end{array}$$

where n is the number of non-gap symbols in the current column.

These n symbols are placed into a two-dimensional space, in which the distances between each other are still preserved as much as possible. This procedure involves a statistical technique called *classical multidimensional scaling (classical MDS)*, also known as *(*Torgerson-Gower scaling) or *principle coordinates analysis (PCoA)* [12]. Multidimensional scaling takes a distance matrix and assigns a coordinate to each item in a lower dimensional space, such that the Euclidean distance between any two coordinates is approximately equal to the original distance value between the corresponding items.

An statistical function called *cmdscale()* [13] is utilized in this multidimensional

scaling procedure. This function is written in R, which is a programming language for statistical computing [14], and is provided in Rs stat package. It reads a distance matrix and returns a set of points in a k-dimensional space, where k is a user-defined parameter to the function, and must be less than the number of points n [15, 16].

In our approach, k is set to 2, meaning the distance matrix is scaled down to a two-dimensional space. However, a column of an MSA may have as few as one or two non-gap symbols, so that the distance matrix of this column is only 1×1 or 2×2. In such cases, $k = 2$ will not be a valid parameter to *cmdscale()* because of the $k < n$ requirement. So we let

$$k = \begin{cases} 0 & \text{if } n = 1 \\ 1 & \text{if } n = 2 \\ 2 & \text{otherwise} \end{cases}$$

Then we perform the multidimensional scaling on the distance matrix M. When $n < 2$, the generated coordinates are lower than two-dimensional, and the missing coordinate values are filled by zeros.

3.2 Color Space Mapping

The above procedure returns a set P of n coordinates (x, y) in a two-dimensional space, representing the n symbols in a column of a multiple sequence alignment. This two-dimensional space is further mapped to a three-dimensional color space.

The CIE Lab color space [11] is a color-opponent space. The name *Lab* stands

for the three dimensions: L for lightness, and a and b for the color opponents based on nonlinearly compressed *CIE XYZ color space* coordinates [17, 18]. Compared to the *RGB* and *CMYK* color models, the design of Lab color emphasizes more on approximating human vision rather than physical devices. This feature makes it suitable for our visualization purpose [19].

Coordinates in P have only two dimensions x and y. To map P to the three-dimensional CIE Lab space, L for lightness is set to a fixed value 75, and two other dimensions a and b are assigned as x and y scaled from $[0, 1]$ to $[0, 100]$.

$$a = x * 100$$

$$b = y * 100$$

At the end of this step, n CIE Lab colors (L, a, b) are created.

3.3 Hue Rotation and Optimization

One of the limitations in many previous MSA visualization techniques is, they introduced unnecessary confusion by using a fixed color scheme. Even for exactly same aligned columns, colors may be totally different, which won't represent any biological dissimilarities. To minimize this effect, we convert the colors from CIE Lab space to *CIE LCH space*, and perform rotations to eliminate noisy colors to the greatest extent possible. In this way, we smooth the color pattern and emphasize the real similarities and dissimilarities among sequences and blocks.

The *CIE LCH color space*, also known as polar-Lab color space, is a cylindrical transformation of the CIE Lab space so that the a and b axles are converted to a polar coordinate system. The radial coordinate C measures *chroma* and the angular coordinate H measures *hue*. LCH space makes it easier to rotate colors, by changing their *hue* value. The CIE LCH colors can be converted from CIE Lab colors using an R library called *colorspace* [20, 21].

Hue values are circular, meaning 0 and 2π are identical. Given two angles p and q, the difference between them is given by

$$\operatorname{diff}(p, q) = (p - q + \pi) \bmod 2\pi - \pi$$

This calculation returns the difference in the interval $[-\pi, \pi)$, or in degrees $[-180°, 180°)$. The absolute value of the difference

$$|\operatorname{diff}(p, q)| = |(p - q + \pi) \bmod 2\pi - \pi|$$

returns in the interval $[0, \pi]$, or in degrees $[0°, 180°]$.

Given a set A of angles a_i, the average angle $\overline{A}$ is

$$\overline{A} = \arctan \frac{\sum_{i=1}^{n} \sin a_i}{\sum_{i=1}^{n} \cos a_i}$$

and the variance $\operatorname{Var}(A)$ is

$$\operatorname{Var}(A) = \frac{\sum_{i=1}^{n} \operatorname{diff}(\overline{A} - a_i)^2}{n}$$

Given a colored MSA Z with m rows and n columns, to estimate its overall noise

level, a penalty function pen(Z) is defined by

$$\text{pen}(Z) = \sum_{j=1}^{m} \text{Var}(h_{1j}, h_{2j}, \cdots, h_{nj})$$

where h_{ij} is the hue value of the symbol at the ith column and jth row.

We look for an optimized set R of n rotation angles r_i, such that after rotating each hue in every ith column of Z clockwise by r_i, the rotated MSA Z' achieves the minimized return value of the penalty function pen(Z').

The optimization procedure is performed by R's general-purpose optimization function *optim()* [13]. Among five available optimization algorithms provided by this function, we choose the *bounded Broyden-Fletcher-Goldfarb-Shanno* (L-BFGS-B) method [22]. This algorithm allows box constraints, meaning each variable can be given a lower and/or upper bound. Plus, it uses a limited amount of memory. The initial values are set to 0 and boundaries to $[0, 2\pi]$. More detailed comparison between different optimization functions and algorithms will be discussed later in Chapter 6.

3.4 Hue Flipping

One characteristic of the coordinates generated from multidimensional scaling is, they can not only be rotated, but also flipped, without changing the distances between each other. Adding flipping ability to the above rotation optimization procedure may result in lower minimal penalty values and better coloring results. To keep the penalty function simple and fast, we perform a heuristic flipping *before* the rotation

step.

For each pair of adjacent columns C_i and C_{i+1}, $1 \leq i \leq n-1$, compare their hues row by row. Rows with gaps in these two columns are not take into consideration. For those rows with non-gap symbols on both sides, record their hue values

$$h_{i,1}, h_{i+1,1}, h_{i,2}, h_{i+1,2}, \cdots, h_{i,k}, h_{i+1,k}$$

into two lists H_i and H_{i+1}:

$$H_i = h_{i,1}, h_{i,2}, \cdots, h_{i,k}$$

$$H_{i+1} = h_{i+1,1}, h_{i+1,2}, \cdots, h_{i+1,k}$$

In both lists, compare two neighboring hue values $h_{i,j}$ and $h_{i,j+1}$, $1 \leq j \leq k-1$, using diff() function described in section 3.3. If $\text{diff}(h_{i,j}, h_{i,j+1}) < -\frac{\pi}{8}$, the change from $h_{i,j}$ to $h_{i,j+1}$ is called a *clockwise change*; if $\text{diff}(h_{i,j}, h_{i,j+1}) > \frac{\pi}{8}$, the change is called a *counterclockwise change*; otherwise, it's considered to be unchanged.

If a column contains more clockwise changes than counterclockwise changes, it's called *clockwise column*; if it contains more counterclockwise changes, it's called *counterclockwise column*; otherwise, it's a neutral column. Note that since gap rows are filtered out of the comparison, the type of a column may change when compared to different adjacent columns. For instance, a column may be clockwise compared to the previous column, and counterclockwise compared to the next column.

If two adjacent columns are in opposite types, flip the latter one to make them in same type. Then the flipped alignment are sent back to the procedure discussed in section 3.3.

3.5 Sequence Sorting

The alignment graph is supposed to provide information on how sequences will group with each other. We sort sequences after coloring optimization, based on the average hue of each one, which is calculated by the average angle function proposed previously in section 3.3.

The fact that hue values are circular, results in an additional question on where to cut the circle to get a sorted list of sequences. In a sorted circle of n average hue values, $h_1, h_2, \cdots, h_n$, we calculate the absolute difference between each pair of adjacent ones (h_1 is adjacent to h_n).

$$\Delta h_{1,2} = |\text{diff}(h_1, h_2)|$$

$$\Delta h_{2,3} = |\text{diff}(h_2, h_3)|$$

$$\cdots$$

$$\Delta h_{n,1} = |\text{diff}(h_n, h_1)|$$

Then, the circle is split at the biggest gap

$$\Delta h_{max} = \Delta h_{i,i+1} = |\text{diff}(h_i, h_{i+1})|$$

to best avoid separating similar sequences.

3.6 Flowchart

3.7 Time Complexity

Chapter 4

Implementation

We implemented the entire visualization pipeline on a Mac OS X server. The implementation consists of an R script as back-end, a Perl CGI script as API, and an HTML/JavaScript web page as user interface.

The R script *coloring.r* takes the pair-wise similarity score file as input, performs multidimensional scaling and color optimization, sorts sequences, and output colors and sequence information into two tab-separated files *colors.tsv* and *seqinfo.tsv*. The front-end web page takes an alignment ID as parameter and sends it to the API. The Perl CGI script parses the color file and sequence information file, along with the FASTA-format alignment file, and sends them back to front-end in JSON format [23]. The front-end uses JavaScript to generate the alignment matrix and render it with colors and symbol letters.

4.1 R Script

The script *coloring.r* begins with reading three arguments from command-line calls: score file name, colors file name and sequence information file name. The first file is for input and the latter two are for output. An example of calling this script on UNIX operating system is given below.

```
Rscript coloring.r data/score.tsv data/colors.tsv data/seqinfo.tsv
```

The file *score.tsv* contains four columns separated by tabs: column number, first row number, second row number, and similarity score between these two rows in this column. The row and column numbers start from 1, and the similarity score values range from 0 to 1. Figure 4.1 shows a few example lines in a *score.tsv* file.

#COL_NUMBER	#ROW_NUMBER_1	#ROW_NUMBER_2	#RES_PAIR_HIT/NALT
1	1	2	1.000000
1	1	6	0.933333
1	1	10	0.866667
1	2	6	0.933333
1	2	10	0.866667
1	6	10	0.133333
2	1	6	1.000000
2	1	10	1.000000
2	6	10	1.000000

Figure 4.1: An example of similarity score file *score.tsv*, showing the format of the file. The four columns are: column number, the first row number, the second row number, score value.

The script *coloring.r* consists of four parts. The first part reads the input file, performs multidimensional scaling and create color for each symbol in the CIE LCH

space. The second part rotates and flips color hues and optimize the penalty function. The third part sorts the sequences by their average hues and calculates each sequences average RGB color. The last part outputs the symbol colors and sequence information into corresponding files.

The color file *colors.tsv* has five columns separated by tabs: column number, row number, and the symbol color in RGB triplet (red, green, blue) in the range from 0 to 1, which will be converted into hexadecimal format used in the user interface module. The sequence information file seqinfo.tsv also has five columns: row number, average hue from 0 to 360 degrees, and average color in RGB triplet. Figure 4.2 shows how a typical *color.tsv* file looks like.

```
1       1       0.1065524       0.5616907       0.3943046
1       2       0.106256        0.5616652       0.3936842
1       6       0.1065487       0.5616904       0.3942968
1       10      0.1065581       0.5616912       0.3943165
2       1       0.09057406      0.5677889       0.2867890
2       2       0.09057406      0.5677889       0.2867890
2       6       0.09057406      0.5677889       0.2867890
2       10      0.09057406      0.5677889       0.2867890
3       1       0.07634505      0.5664451       0.2581746
3       6       0.1113592       0.570106        0.3250781
3       10      0.07634505      0.5664451       0.2581746
```

Figure 4.2: An example of coloring result file *color.tsv*, showing the format of the file. The five columns are (from left to right): row number, average hue, red, green and blue.

4.2 API

A Perl CGI script *api.pl* and a Perl package *Mavis::API* act as the API between the R back-end and HTML/JavaScript front-end. The Perl CGI *api.pl* takes two parameters from HTTP requests, *action* and *id*. The default action is *alignment*, meaning the alignment information, including sequences, colors, average colors and order of sequences, should be returned. In this action, an additional *id* parameter is expected as the alignment ID. Another available action is *list*, which requests a list of all existing alignment IDs.

The API returns the query result to the web page in JSON (JavaScript Object Notation) format, which is a lightweight text-based data-interchange standard. JSON is derived from a subset of JavaScript syntax and can be easily parsed in JavaScript as well as many other languages, like Perl. We included the Perl JSON 2.51 module to provide JSON encoding/decoding ability to our API [23].

Figure 4.3 is an example of the API response in JSON format:

```
{ "status"    : 1,
  "id"        : "demo",
  "colors"    : [ ["#666","#666","#666","#666"],
                  ["#665","#667","#667","#665"]],
  "sequences" : [ {"name":"Seq02", "color":"#26e", "seq":"DEMO"},
                  {"name":"Seq01", "color":"#26e", "seq":"DEMO"}]
}
```

Figure 4.3: An example of Mavis API response in JSON format.

4.3 User Interfaces

We implement an web interface with HTML and JavaScript to send API requests and draw visualization graphs. The JavaScript code dynamically communicates with API, creates an HTML table, and fills in colors and symbols. This part is simplified by jQuery, which is a popular cross-platform JavaScript library that abstracts complicated DOM selection, CSS manipulation and Ajax effects.

4.4 Design Principles

Chapter 5

Results

5.1 Mavis for Web

testrrr

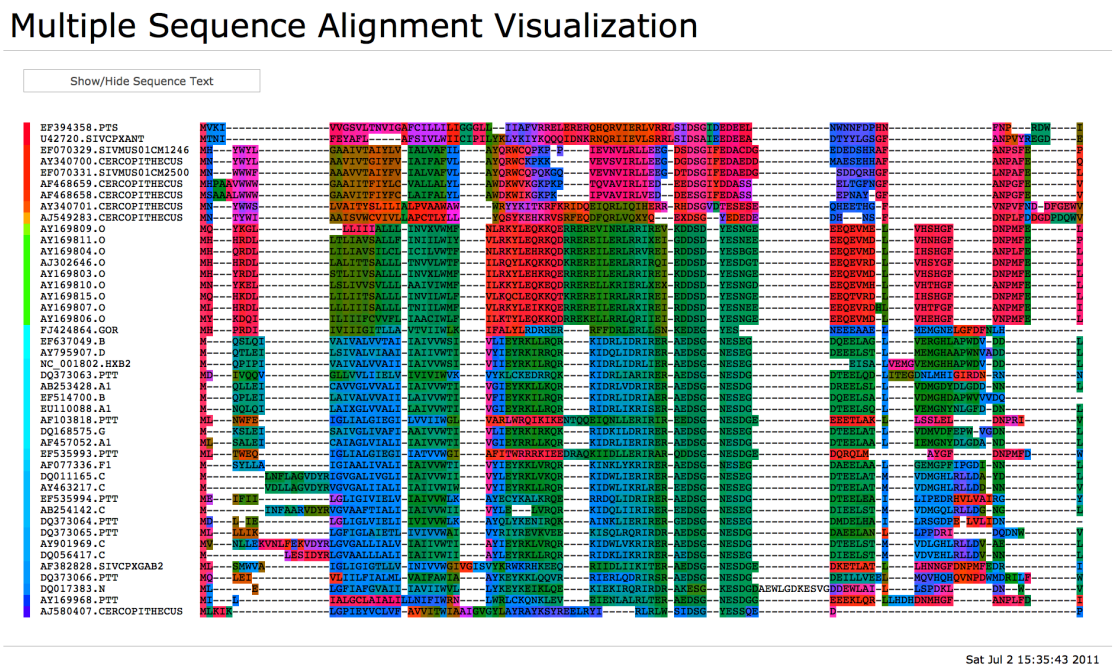


Figure 5.1: An MSA rendered by Mavis in Firefox 5.

5.2 Mavis for iPhone

5.3 Mavis for iPad

5.4 Validity

5.5 Benchmarks

To evaluate the validity and performance of our approach, we tested it in two different ways. First, we run it on a few artificial datasets, to see if it creates the color patterns as we expected. Then, we run it on the BAliBASE dataset, observe how alignment size, in both number of rows and columns, will affect the excution time, and compare it with other tools.

Chapter 6

Discussion

In this report, we introduced a new way of coloring and visualizing multiple sequence alignments based on symbol-symbol confidence (or similarity) scores. Our main objective has been to create an intuitive way of showing the quality and internal structure of an alignment, using colors, which is the most natural and sensible way for human eyes. The basic idea is, the greater the dissimilarity between sequences, the more obvious difference in colors.

6.1 Color Spaces

Most previous techniques were based on fixed color schemes, meaning each sequence symbol, like an amino acid or a nucleotide, is assigned to a pre-selected color. This is of course a simple and straightforward solution, and is easy for observers to find a particular symbol or pattern. However, this fixed-scheme approach does not emphasize

the relationship between adjacent columns and the internal regions and structures in the level of the whole alignment. This is the main reason why we don't use any predetermined color scheme, but calculate colors only based on the distance matrices, and rotate and flip colors to make them as smooth as possible. This strategy brings significant improvement to the alignment coloring, by showing the better-aligned regions and blocks in same or similar colors, while those irrelevant ones in quite different colors.

Since we are to convert the scaled coordinates to colors, choosing a suitable color space is an fundamental task. At first we scaled the distance matrix down to a three-dimensional space and mapped it directly to the RGB color space. However, we soon found a problem, that some colors, like very dark ones and very light ones, did not perfectly serve the purpose of representing distances, yet made the graph more noisy. We realized that we don't really need the whole color space and all possible colors. Instead, those colors with proper range of lightness and different hues will be enough to do the job. So we chose to scale down to a two-dimensional space and map it to CIE Lab space with a fixed lightness value (75). The reason why we didnt choose one-dimensional scaling is that, a linear space either could not map to enough number of colors (for example, use only one primary color, like red), or could not preserve the distance information (for example, use only hues). So two dimensions is a good balance.

6.2 Optimization Algorithms

In R, there are several general purpose optimization packages which offer facilities for solving our color rotation and flipping problems. Two popular functions are optim() and nlminb() from package stats. Function optim() provides implementations of five algorithms: *Broyden-Fletcher-Goldfarb-Shanno (BFGS)*, *bounded BFGS (L-BFGS-B)*, *conjugate gradient (CG)*, *Nelder and Mead (Nelder-Mead)*, and *simulated annealing (SANN)*. *Nelder-Mead* [24], which is the default one, returns robust results but is relatively slow. *CG* [25] in faster in larger optimization problems, but more fragile. *BFGS* is a balance, and *L-BFGS-B* further provides the ability of box constraints, that each variable can be given a lower/upper bound. *SANN* [26] is more powerful on rough surfaces but relatively slow. Another function *nlminb()* offers similar box constraint optimization and similar performance to *L-BFGS-B*, so these two algorithms are chosen to a further test.

Optimization algorithms always suffer from the local versus global minimum problem, and the final result are more or less unstable and depending on the initial values. To decide which one of *L-BFGS-B* and *nlminb()* is more stable in our approach, we run a test on both of them. The test dataset is the alignment of 44 Vpu protein sequences from GUIDANCE server [9]. We randomly created 100 sets of initial values, performed optimizations, and see how the return value of the penalty function (described in section 3.3) changed. Table 6.1 shows the results of the comparison. Obviously *L-BFGS-B* is more stable in our algorithm.

Table 6.1: Comparison of *L-BFGS-B* and *nlminb()*

Algorithm	Min. Penalty	Avg. Penalty	Max. Penalty	Std. Deviation
Initial	370,584	425,728	444,771	13,998
L-BFGS-B	136,909	178,850	247,218	17,871
L-BFGS-B (3 times)	136,909	178,850	247,218	17,871
mlninb()	146,759	181,688	426,535	34,579
mlninb() (3 times)	142,888	171,380	325,469	21,402

References

[1] A. Cornish-Bowden, "Nomenclature for incompletely specified bases in nucleic acid sequences: recommendations 1984.," *Nucleic Acids Research*, vol. 13, pp. 3021–3030, May 1985. PMID: 2582368 PMCID: 341218.

[2] D. A. Benson, I. Karsch-Mizrachi, D. J. Lipman, J. Ostell, and D. L. Wheeler, "GenBank," *Nucleic Acids Research*, vol. 36, pp. D25–D30, Dec. 2007.

[3] R. C. Edgar and S. Batzoglou, "Multiple sequence alignment," *Current Opinion in Structural Biology*, vol. 16, no. 3, pp. 368–373, 2006.

[4] J. D. Thompson, T. J. Gibson, and D. G. Higgins, "Multiple sequence alignment using ClustalW and ClustalX.," *Current Protocols in Bioinformatics*, p. 2.3, 2002.

[5] A. M. Waterhouse, J. B. Procter, D. M. A. Martin, M. Clamp, and G. J. Barton, "Jalview version 2—a multiple sequence alignment editor and analysis workbench," *Bioinformatics*, vol. 25, pp. 1189 –1191, May 2009.

[6] J. Procter, J. Thompson, I. Letunic, C. Creevey, F. Jossinet, and G. Barton, "Visualization of multiple alignments, phylogenies and gene family evolution," *Nature Methods*, vol. 7, no. 3 Suppl, pp. S16–25, 2010.

[7] K. Lin, A. C. W. May, and W. R. Taylor, "Amino acid encoding schemes from protein structure alignments: multi-dimensional vectors to describe residue types," *Journal of Theoretical Biology*, vol. 216, no. 3, pp. 361 – 365, 2002.

[8] O. Penn, E. Privman, G. Landan, D. Graur, and T. Pupko, "An alignment confidence score capturing robustness to guide tree uncertainty," *Molecular Biology and Evolution*, vol. 27, no. 8, pp. 1759 –1767, 2010.

[9] O. Penn, E. Privman, H. Ashkenazy, G. Landan, D. Graur, and T. Pupko, "GUIDANCE: a web server for assessing alignment confidence scores," *Nucleic Acids Research*, vol. 38, no. Web Server, pp. W23–W28, 2010.

[10] I. Borg and P. J. F. Groenen, *Modern multidimensional scaling: theory and applications*. Springer, 1997.

[11] K. McLaren, "XIII—The development of the CIE 1976 (L* a* b*) uniform colour space and colourdifference formula," *Journal of the Society of Dyers and Colourists*, vol. 92, no. 9, pp. 338–341, 1976.

[12] J. C. Gower, "Some distance properties of latent root and vector methods used in multivariate analysis," *Biometrika*, vol. 53, no. 3-4, pp. 325–338, 1966.

[13] R Development Core Team, *R: a language and environment for statistical computing*. R Foundation for Statistical Computing, Vienna, Austria, 2009. ISBN 3-900051-07-0.

[14] R. Gentleman and R. Ihaka, "R: a language for data analysis and graphics," *Journal Of Computational And Graphical Statistics*, vol. 5, no. 3, pp. 299–314, 1996.

[15] F. Cailliez, "The analytical solution of the additive constant problem," *Psychometrika*, vol. 48, no. 2, pp. 305–308, 1983.

[16] M. A. A. Cox and T. F. Cox, "Multidimensional scaling," in *Handbook of Data Visualization*, pp. 315–347, Berlin, Heidelberg: Springer Berlin Heidelberg, 2008.

[17] C.I.E., *Commission internationale de l'Eclairage proceedings, 1931*. Cambridge University Press, Cambridge, 1932.

[18] T. Smith, "The C.I.E. colorimetric standards and their use," *Transactions of the Optical Society*, vol. 33, no. 3, pp. 73–134, 1931.

[19] D. Margulis, *Photoshop LAB color: The canyon conundrum and other adventures in the most powerful colorspace*. Peachpit Press Berkeley, CA, USA, 2005.

[20] R. Ihaka, P. Murrell, K. Hornik, and A. Zeileis, *colorspace: color space manipulation*, 2009. R package version 1.0-1.

[21] A. Zeileis, K. Hornik, and P. Murrell, "Escaping RGBland: selecting colors for statistical graphics," *Computational Statistics & Data Analysis*, vol. 53, pp. 3259–3270, 2009.

[22] R. H. Byrd, P. Lu, J. Nocedal, and C. Zhu, "A limited memory algorithm for bound constrained optimization," *SIAM Journal on Scientific Computing*, vol. 16, p. 1190, 1995.

[23] D. Crockford, "The application/json media type for javascript object notation (json)," tech. rep., RFC 4627, July, 2006.

[24] J. A. Nelder and R. Mead, "A simplex method for function minimization," *The Computer Journal*, vol. 7, no. 4, pp. 308 –313, 1965.

[25] R. Fletcher and C. M. Reeves, "Function minimization by conjugate gradients," *The Computer Journal*, vol. 7, no. 2, pp. 149 –154, 1964.

[26] C. J. P. Bélisle, "Convergence theorems for a class of simulated annealing algorithms on rd," *Journal of Applied Probability*, vol. 29, no. 4, pp. 885–895, 1992.