Spring Logo:

A New Way of Visualizing HMM Logo for Sequence - Profile Alignment

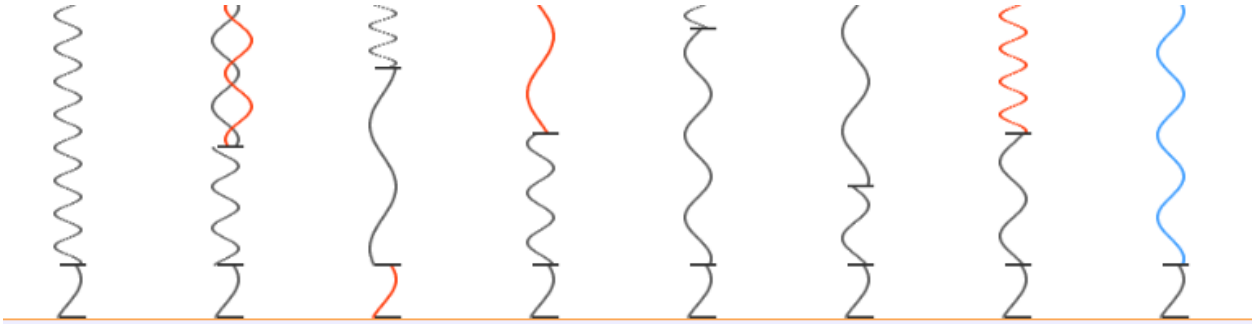

Fig.1. Spring Logo visualization employs the wave representation idea enabling analysts to compare probability and the score of the fitness together. The visualization also clarifies the reverse complement regions and the letters missed in the Hidden Markov Model.

Abstract – One bottleneck in bioinformatics algorithms such as sequence-profile alignment has to do with evaluating the final alignment score. To a limited degree, this score shows how much the sequence is similar to the model. However, the final score only shows the overall tally, but it does not give us any information about the local areas of sequences. Showing local areas of interest could be greatly aided by visualization tools that display the sequence and the model to the analyst. This way an analyst could discover the results in all areas of interest. This paper presents our design decisions in improving one of the existing visualization methods – sequence logo-. Current visualization logos in this domain focus on showing the similarity between the new sequence and the Hidden Markov Model (HMM), by representing the characters and the probability of each character in each position. We present a novel logo display, "Spring Logo", which emphasizes the degree of fitness in both probability and weight aspects. Our tool replaces comparing each character against the model one by one.

Index Terms—Bioinformatics visualization, Protein Sequence, DNA sequence, Sequence Logos, HMM Logos, Sequence Profile Alignment

1 Introduction and Background

Living cells consist of their basic units called genes. A gene carries information of construction and maintenance of that cell. These genes are gathered together and build DNA sequences. These DNA sequences are copied in a process called transcription, producing an RNA copy of the gene's information. The information of the RNA can then be translated and produces protein chains.

To visualize sequences[1], it is important to know that the RNA and DNA consist of small parts called nucleotides, and proteins consist of small parts called amino acids. Different alphabetic letters usually represent both nucleotide and amino acids.

Finding similarities between sequences begins with the assumption that if two genes have similar sequences, then they are likely to have similar functions, or there is an evolutionary relationship between them. In bioinformatics, we have the *sequence alignment* process that is a way of discovering similar regions between sequences. We typically arrange sequences in rows and try to find these regions.

In *pairwise alignment*, we compare only one sequence to another. Although pairwise alignment is useful in many cases, sometimes it leads to an incorrect result. This is mostly because we are comparing two sequences with no prior knowledge of other members (other similar sequences) of the protein or gene sequences, and as a result in our algorithm there is no information about a particular position. In general, considering a *multiple sequence alignment* of a family of sequences, one will notice that there are two types of positions across the entire alignment: 1. Variable positions in which mutations were allowed and it won't cause a loss of function in the sequence. 2. Conserved positions that would result in a loss of sequence functionality if there

[1] We refer to DNA, RNA, and Protein using the general term of "sequence"

were to be a change. (For example those positions may not accept gaps across several of the family members).

Another drawback of the pairwise alignment is the case of comparing one sequence against a huge number of sequences in sequence databases: As the sequence databases grow, the amount of time to compare each sequence to those sequences in such databases grows accordingly.

So how can we resolve the problem? We know that a group of the similar sequences (family members) could tell us much more than a single representative sequence. So we may choose to cluster the sequences in huge databases and build a general model of them, which is called a sequence family or *sequence profile*. This family has the information about all of the sequences in each position. Then we can just compare a new sequence with those families that are general representations of their sequence members. Below we bring an example of this concept:

Suppose we have 5 sequences:

HGKVLHL

HGKVLHL

HGKVAHL

HGKVGHL

HGLVLHL

In a very simple way, we could represent these data as a regular expression of the form:

HG [KL] V [LAG] HL

This means H will be in the first position, G will be in the second position, K or V will be in the third position, and so on.

But what is raised here is the fact that one does not need to bring a sequence that has an exact number of characters as a model. If the new sequence has extra characters that do not seem to

fit the model, we can classify them as an insertion in the model, and if it has fewer characters, we can consider it as a deletion in the model. Below is an example of this.

My new sequence: HGKHL

So the model is changed to below:

HG [KL] - - HL

Thus we may allow this insertion and deletion to occur between all the positions (with their own probability that is called the background probability). But it is hard to display this new information in our representation of the model, and these family models may be shown by the *Hidden Markov Models* concept.

A Markov chain is simply a collection of states, with a certain transition probability between each state. The model is referred to as 'hidden' because you do not know which series of transitions produced that state.

An HMM that models a sequence will have one state for every position in the sequence, and each of those states will have matched states (twenty possible outcomes, for each amino acid or 4 possible outcomes for each nucleotide) and two other states for the insertion and deletion.

Any sequence can have different scores (sum of all probabilities for each transition) tracing through different paths of the model from one state to the next, and we may choose the path with the highest score as the final path.

One simple representation of HMMs is show in Figure 2.

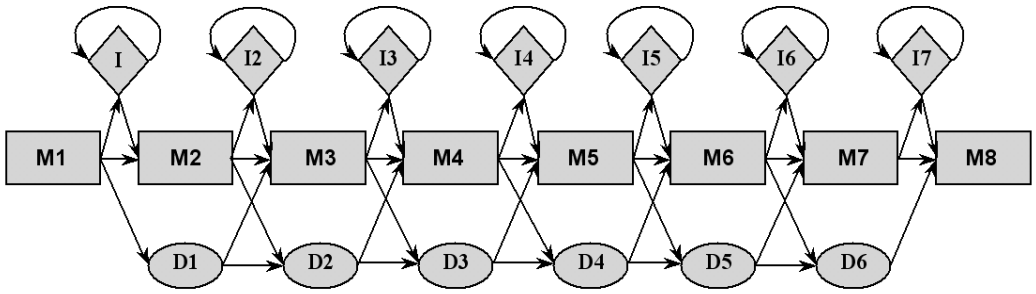

Fig. 2 Match States with Insertions and Deletion

In order to complete the simple model, we want to add the probability of each match, insertion and deletion to that. The result is shown in Figure 3. In this figure we have:

S: start state, N: N-terminal insertion state, B: beginning state of the core of HMM, M: matching state (square), D: deletion state (yellow circle), I: insertion state (pink diamond), E: ending state of the core of HMM, J: jump state from E to B to allow domain duplication, C: C-terminal insertion state, T: terminal state. Circles (S, B, D, E, T) states are dummy states that do not generate symbols. The connectivity (directed edge) between two states denotes a possible transition. The number associated with an edge is the transition probability. The connectivity between B, M, D, I, and E state is similar to the traditional profile of HMM, except that B can jump to any M state and any M state can jump to E. [4]

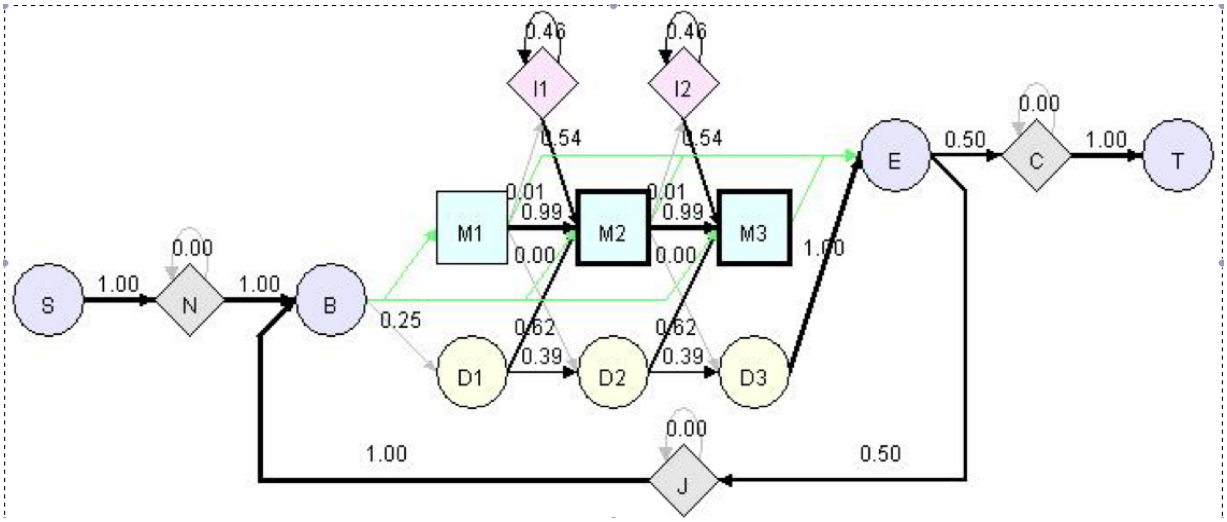

Fig. 3. A simple HMM profile HMM model visualized by HHMVE. [4]

One common way of graphically visualizing HMMs is called a Sequence *Logo*. A logo is graphically represented by columns of letters for each position. The relative height of each letter in the stack is its relative probability (its frequency among all the other characters) at the position. Usually, colors are used for different characters. If we ignore the insertion and deletion probabilities of a HMM, we can visualize it with a sequence logo. (e.g. Figure 4)

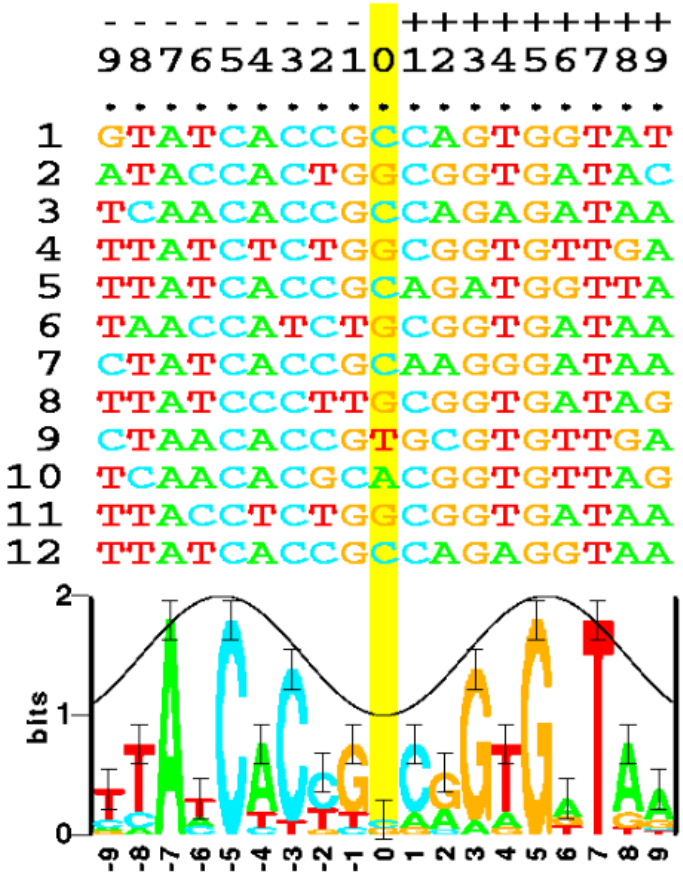

Fig. 4. Some aligned sequences and their logo.

However, this would mean ignoring a big part of the model. Therefore the next step [2] is to modify Sequence Logos in such a way that they can show which positions can be deleted and where we can expect insertions. Figure 5 shows a sequence logo, which has deleted and

inserted states. In this figure, positions with narrow match state columns are more likely to be deleted. The total width of a red-shaded (dark + light) stack visualizes the expected number of inserted letters. [2]

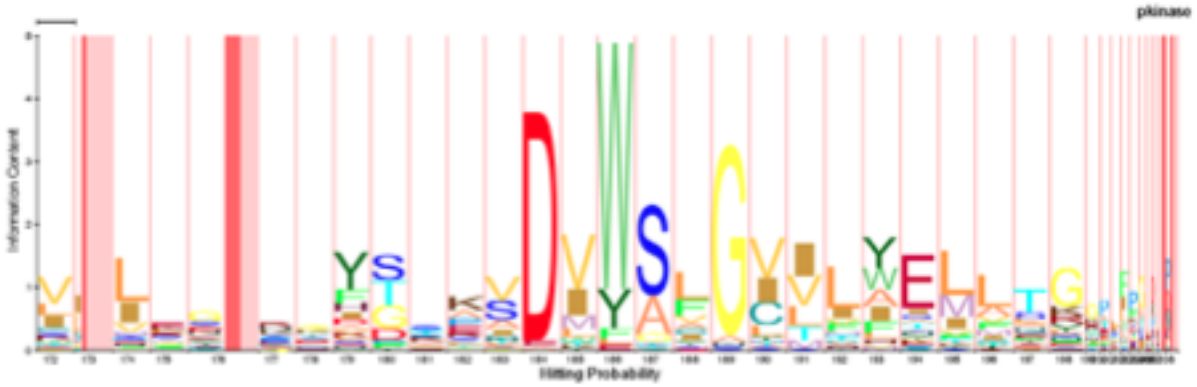

Fig. 5. Example of a HMM Logo.

Another feature that we want to add to this model is showing our new sequence against the model. One way to do this is to write the sequence below the logo. In this paper we will discuss the idea that if we could represent the path of a new sequence against the model, the analyst would be independent for comparing each character against the model each time. Here our method to represent the logo resolved this problem, but what still remains as a question is how to represent a profile against another profile? Figure 6 shows visualizing profile-profile alignment using pairwise HMM logos. [7]. The aligned states in each HMM are framed and connected by a block. Opening a block that is shaded in grey shows inserted states, and closing the block(s) shows deleted states.

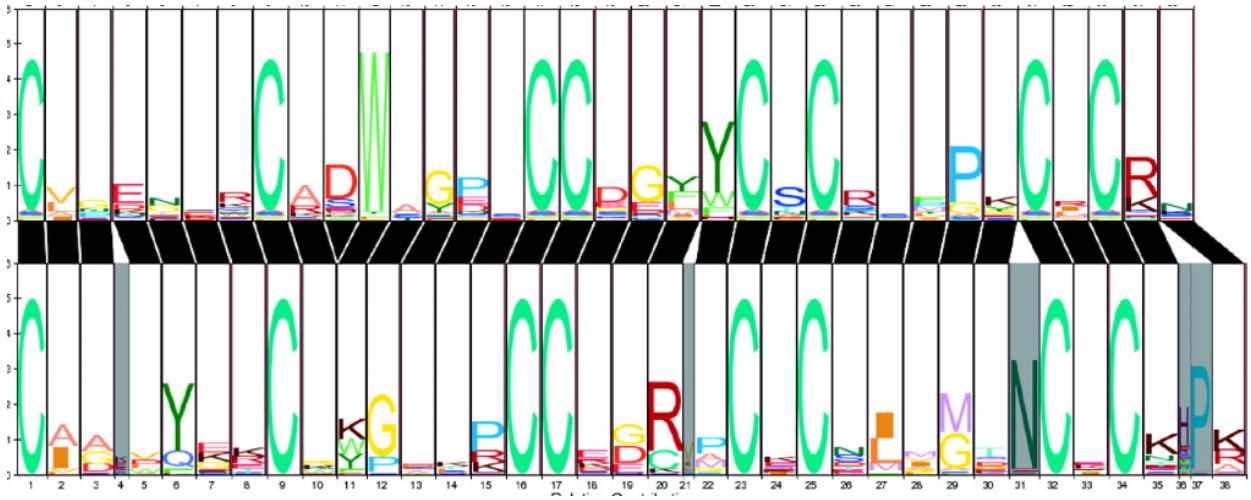

Fig. 6. Alignment of two HMM families, an HMM Logo is drawn for each profile family.

2 **Related Works**

Gary Churchill of Cold Spring Harbor Laboratory first proposed HMMs for use in bioinformatics. However, the original development of Profile Hidden Markov Models, took place at the University of California, Santa Cruz, in David Haussler's group. The UCSC group developed an HMM implementation called the Sequence Analysis Method (SAM). Meanwhile at Washington University, Sean Eddy developed a suite of Profile Hidden Markov Model tools called the HMMer package (pronounced Hammer). This became the basis for the Protein Family database, commonly known as Pfam.

The HMMer package is now the most widely used system for profile-HMM analysis, however this package has no visual representation by itself, and that is why some analysts prefer to use the SAM package instead. For example the "make logo" tool of the SAM software package does make a sequence logo.

The first sequence logo was proposed in a paper by D. Schneider and Stephens [8]. After that there arose lots of logo generators such as WebLogo, LogoBar, MoRAine, and GENIO/logo. These logos are used to represent sequence-profile alignment as well as profile-profile alignments. The evolution of these methods has been discussed in the previous part.

3 Design Decisions

Visualizing a sequence alignment task requires several types of information in each position. These include:

(1) If a match occurs:

(1.1) information about the probability of that letter in the model.

(1.2) the weight of that letter in the model,

but not the letter itself.

(2) If an insert occurs:

(2.1) the probability of having an insertion of this letter in that position in the model,

(2.2) the weight of having an insertion of this letter in that position in the model.

(3) If a delete occurs:

(3.1) the probability of having a deletion of this letter in that position in the model.

(4) In the DNA or RNA sequences, if a match, insert or delete occurs, whether a nucleotide matches to its reverse complement or not, whether a reverse complement inserted instead of the original nucleotide, and whether a reverse complement has been deleted instead of the original nucleotide.

(5) The probability and the score of the letters that were not in the training sequences in a particular position in all three matches, deletions, and insertion states.

To our best knowledge, no visualization tools exist to integrate all of this information. The existing logos show parts 1.1, 2.1, 3.1, and 4, but most of them visualize this type of information by showing the letters. No visualization tool exists to represent parts 1.2, 2.2, and 5.

We designed a visual encoding that captures all of these features in a single representation with the intent to eliminate the need of showing the letters and using too many color encodings.

3.1 Representing the Probability

Sequence logo designers use the relative height of each letter within one column (each position) to display the match probability [2]. We decided to use each spring's height to represent the relative probability within the whole model, instead of dynamic definition in each column. (The old models used this because in their logos, the height of column represents entropy, but we chose this because we saw that knowing the entropy has no meaning to the analysts.)

3.2 Representing the Weight

No logo visualization exists to represent the weight of the letters in each position within the model. What is the weight, and why is it important? One problem with HMMs is called *overspecialization*. It means that if the sequence members that are building the model, are highly similar, the model will be highly similar to those sequences and cannot accept a sequence that is partly similar to them. To prevent this, several methods of *sequence weighting* have been proposed.

One kind of weighting method is the position-specific weighting method proposed by Henikoffs [3].

$$Weight = \frac{1}{m * k},$$

Equation 1. Henikoffs' Weighting Method

where, m is the number of different characters in the column and k is the number of times that character appears in that position.

For example considering these four sequences, shown below, the probability of positioning R in the first column is equal to the probability of positioning R in the second column ($\frac{1}{2}$), but the weight of R being in the first column is $\frac{1}{4}$ whereas the weight of R being in the second column is $\frac{1}{6}$.

So it means that although the probability of these two R is the same, there is a difference between these two Rs.

```
R R
R G
A T
A R
```

Fig. 7. Example of four related protein sequences. The R in the first column is more powerful in the model, in comparison with the R in the second column.

We chose this sequence weighting method because it is simple for our implementation. In our wave representation logo, each oscillation refers to a fixed number of weights. High weight characters produce short wavelengths while short weight characters produce short wavelengths (Figure 8). Simple arcs represent characters which have their weight below a threshold.

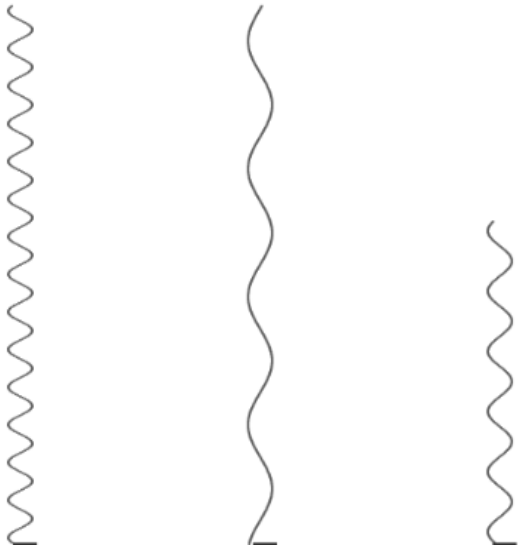

Fig. 8. Different height represents different probability, and different wavelength represents different weight.

3.3DNA has two direction

One fact about DNA sequences that is different from protein sequences is DNA has a reverse complement which means that an A nucleotide can be matched to a T, and a C nucleotide can be matched to a G. Figure 9 shows this puzzle conception.

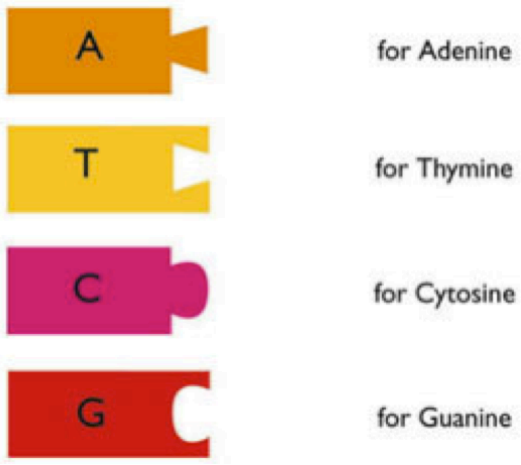

Fig. 9. DNA nucleotides are like puzzle pieces.

One option for capturing the two-direction nature of DNA is to reverse the orientation of the characters in the logo as some designers did (Figure 10). But, this representation has some drawbacks. First, reverse shape of each character should be defined separately. Additionally, this option has to represent the different shape of characters.

Fig. 10. Orientation of some characters changed because of the reverse complement nature of DNA nucleotides.

Another option that was proposed by Nielsen *et al.* in [1] is using the leaf-like shape. In this context it means that if the character is the original one, we force the maximum *amplitude* to occur close to the spring shape's start edge, but if it belongs to the complement one, we force maximum amplitude to occur close to the spring shape's end edge. The result looks like Figure 11.

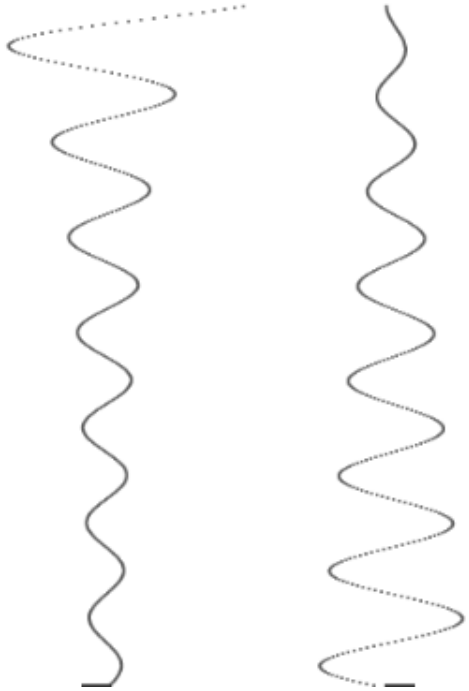

Fig. 11. One option for showing direction is using leaf like shapes. These shapes are showing two different directions.

By choosing this method, the column widths would be different all over the models depending on the existence of tall large weight letters, or short low weight characters.

Instead, we represent this as a wave with phase 180 degrees more than our original wave. Thus, for each match, insertion, and deletion that occurs by the reverse character, we will have a representation like Figure 12.

Fig. 12. Our method showing reverse complement matching.

3.4 Regularize the Overfitting problem

Another potential difficulty is known as *overfitting,* and illustrated by the example shown in Figure 7. Consider these four sequences to be members of our training set, and suppose that they are protein sequences. The first column contains two distinct amino acids: R and A. Using the methods described so far, the probability of any of the other 18 amino acids appearing in the first position is 0. However, if we set all these probabilities to 0 in the model, we will have a model unable to recognize family members that do not begin with R and A. Remember the weighting methods that wanted to solve the overfitting problem. This is somewhat similar to that problem but instead of the weight concept we need to solve the problem of a probability of 0 because here we know that some members might exist that have other characters in that particular position.

A variety of approaches known as *regularization* have been proposed to solve the overfitting problem. The simplest is called *pseudocounts:* this means that, even if a given character does not appear in a column, we will give it a fake probability, and after that we will consider that probability like a real one. The number depends on whether the missing character belongs to amino acids or nucleotides. Below we show the formula for calculating the probability of all characters, even existing or missing ones, in a different way:

$$\Pr obabilityOfEachCharacter = \frac{obserevedcountsofthatcharacter + PseudocountofthatCharacter}{Obserevdcountsoverallcharacters + pseudocountsoverallcharacters}.$$

Equation 2. Pseudocount method is used or calculation the probability.

In this formula the pseudocounts over all characters is deferent between protein and DNA sequences. For proteins it is 20 and for DNA is 4. So for example in the first column of figure 7, the probability of A is $\frac{3}{24}$ and the probability of all other characters is $\frac{1}{24}$ if it belongs to the protein sequences. If it belongs to the DNA sequences, A's probability would be $\frac{3}{8}$, and all missing characters would be $\frac{1}{8}$.

In order to show these fake letters in the model, we chose a fixed length spring with one oscillation, which we always put at the bottom of each column that has these kinds of letters. Each of these springs represents all the missing characters.

Figure 13 shows how the last decision looks like.

Fig. 13. Missing characters in the model have the same probability and the same weights, and are shown at the end of each column by a simple arc.

3.5 Insertion

We may have the probability of an insertion of different letters with different insertion weightings after or before all the positions. We may show the springs in these columns with different color-coding (in our representation we chose blue for that purpose). For saving the

place, we only visualize the insertion characters in the model that an insertion really happened in the new sequence against the model.

3.6 Deletion

There is a probability of deletion in all the positions (we do not have this particular column in our model). Although this probability is not identical in all positions, we will show the deletion just by leaving that column in its own color and we won't show any fitting in that column as Figure 14 shows.

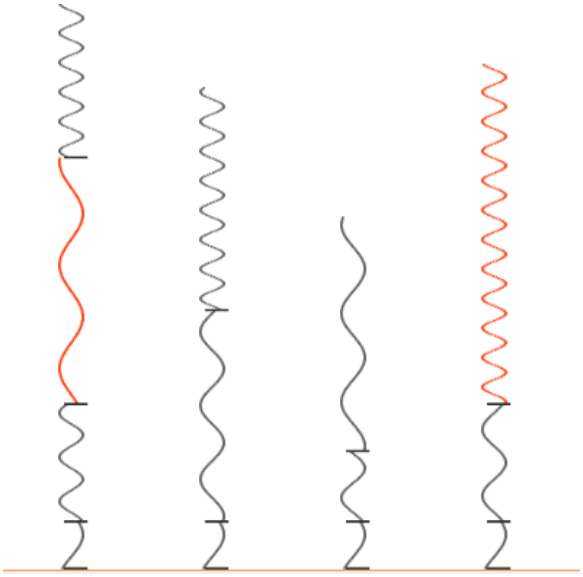

Fig. 14. Column 2 and 4 show a deletion in this two positions by the lack of orange or blue spring.

3.7 Color Coding

We used gray to show the model in areas that matches may occur, and orange color for showing our sequence against the model. In some columns you see a blue spring. These are columns that an insertion occurs and the rest of the spring is shown in gray instead of our new sequence's spring that is shown by a blue color.

So there are three types of columns:

1. Gray springs with one orange spring:

   This shows a match of our red spring against the model, and gray springs are the letters that are not matched by this new sequence, but each of them has their own probability and weight of matching.

2. Gray springs with one blue spring:

   This shows an insertion of a new character in that position in comparison with the model. Each of the springs in this column has a different chance of insertion in this position, but just one of them is inserted (each has its own probability and weight). The probability of this insertion has been shown by a number under the column.

3. Just the gray springs which shows a deletion. A number under the column shows the probability of that deletion.

So we use three colors gray, blue, and orange and also a small black line behind each spring showing differences.

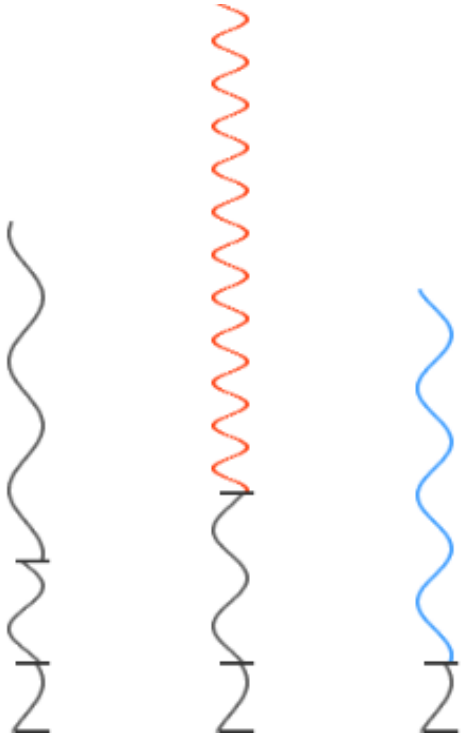

Fig. 15. The first column shows a deletion, the second column shows a match, and the third column shows an insertion between positions.

We checked these colors with Vischeck to be sure that this color-coding is useful for colorblind people as well.

The results are shown in the images below.

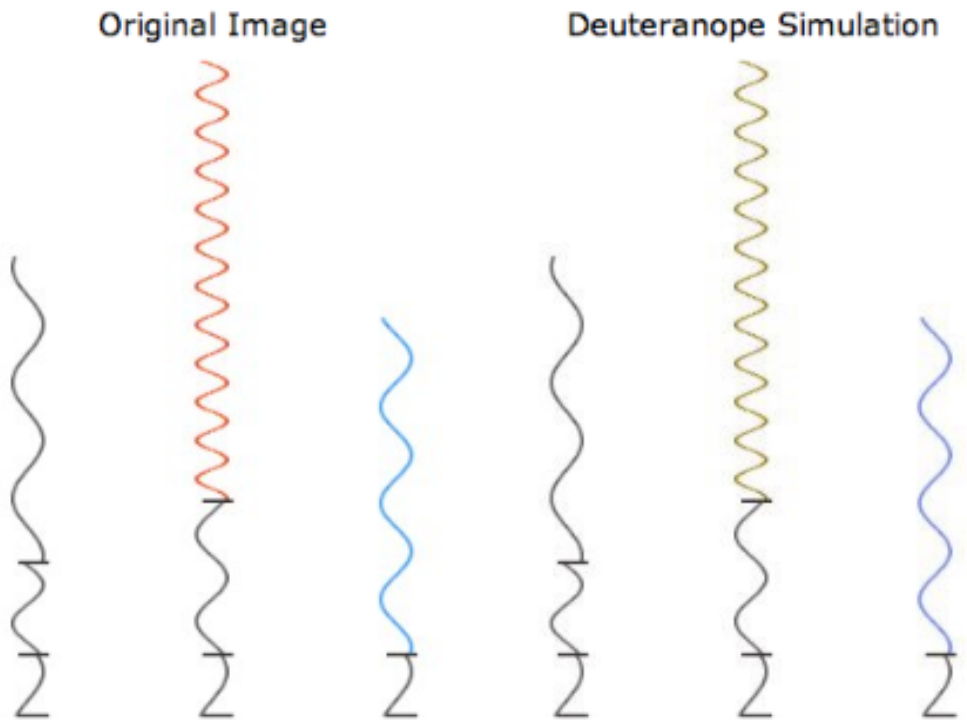

Fig. 16. Deuteranope (a form of red/green color deficit) run by vischeck.com

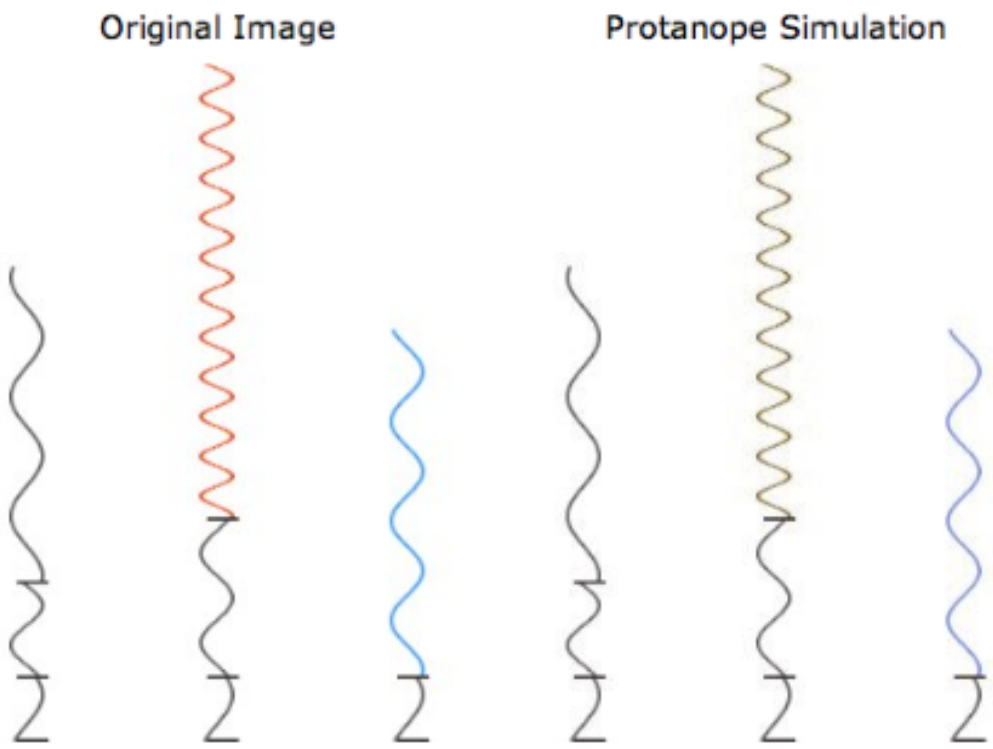

Fig. 17. Protanope (another form of red/green color deficit) run by vischeck.com

4 Implementation of Spring Logo

Our sequence alignment visualization tool, Spring Logo, has not implemented completely because we are in the first software implementation's life cycle that is the prototyping. We decided to implement the interface of the software in order to show it to the analysts. After receiving their feedback, we will complete our system. In this stage we used Action Script 3, Flex Builder framework of implementation. Our prototype has the ability of inserting a spring with given parameters from the user (length, number of oscillations, position, color, and so on.) So we could build a model that shows a particular logo in order to compare the visual effectiveness of our model.

5 Conclusion, limitations, and Future work

In this paper we explored the use of wave representation as a novel encoding for showing a HMM logo, and sequence profile alignment task against this model.

We demonstrated the utility of our spring representation in visualizing the information we need. We tried to enhance logo visualization by using colors and different shapes (alphabets) as less as possible, and improving the model by adding weight features to it. To clarify, the intent of this model is not to give any visual preference to a particular letter based on its genetic characteristics. Instead, we merely wished to show how a letter sequence could be clearly visualized. Therefore, this model has nothing to do with the visual design of any particular letter. As a result, the visual orientation is only significant in that it highlights a given letter's fitness against the model.

One other feature of our visualization model demonstrates the spring-like structure for more than one letter. For example, if the probability and weight of the characters were relatively equivalent, we would have conserved some of the leftover vertical space.

One of the limitations of our visualization is that all of the columns in our current model contain no information about their background probability[2]. Furthermore, our design still does not show the background probability for each recently inserted or deleted column.

One option to show the background probability of the columns is using column width. Future researchers may choose to develop an interactive system that allows analysts to zoom towards a selected column in order to clearly display the columns that have less width (see figure 6).

[2] The probability of having a particular column in the model

Another limitation is that our model does not yet allow the analyst the ability to conveniently compare profiles. This is due to the fact, that we have not yet solved the persistent problem of profile-profile alignments raised by the design flaws shown in earlier models (e.g. figure 7).

Our future work consists of running an experiment to show the usability of our proposed method. To improve the system, we may add interactive elements to each spring in a way that a user could see the exact number of relative and background probabilities as well as the associated weights, which conform to both the model and the new sequence.

REFERENCES

[1] Cydney B. Nielsen, Shaun D. Jackman, Inanc¸ Birol, and Steven J.M. Jones. ABySS-Explorer: Visualizing Genome Sequence Assemblies. IEEE Transactions on visualization and computer graphics, vol. 15, no. 6, 2009.

[2] Benjamin Schuster-Böckler, Jörg Schultz and Sven Rahmann. HMM Logos for visualization of protein families. BMC Bioinformatics 2004, BMC Bioinformatics vol. 5, no. 7, 2004

[3] Rachel Karchin. Hidden Markov Models and Protein Sequence Analysis.

[4] Jianyong Dai, Jianlin Cheng, HMMEditor: a visual tool for profile hidden Markov model, BMC Genomic, 2007

[5] wikipedia

[6] Martin Gollery, Handbook of Hidden Markov Models in Bioinformatics, Chapman & Hall/crc press, 2008

[7] Benjamin Schuster-Böckler and Alex Bateman, Visualizing profile-profile alignment: pairwise HMM logos, Oxford University Press, ,bioinformatics, Vol. 21 no. 12, 2005

[8] Schneider TD, Stephens RM. Sequence Logos: A New Way to Display Consensus Sequences. *Nucleic Acids Res,* vol. 18 no.20, 1990