Project 4: Protein Sequence Comparison

Viktor Babchanik

February 28, 2018

1 Introduction

1.1 Background

The central dogma theory of molecular biology is well understood. For the most part, we understand how a DNA sequence is transcribed into RNA which then gets translated into a protein sequence. However, we do not completely understand how that protein sequence relates to its three dimensional structure. In fact, it is entirely possible for two different protein sequences to form the same three dimensional structure or for two similar protein sequences to form completely different three dimensional structures. For the most part, the mechanisms that govern the folding of a protein are still largely unknown. Although certain amino acids, such as proline and glycine, due to their intrinsic chemical properties, have a tendency to be found in certain secondary structures, tertiary structure prediction and even secondary structure prediction is practically impossible today.

By studying sequence alignment between homologs, we can get a better understanding of how sequences relate to one another and how sequence similarity can lead to similar function, structure and evolutionary descent, which can ultimately answer questions about tertiary structure prediction.

1.2 Problem of Interest

There are many sequence alignment techniques that allow us to compare protein sequences, each with their own advantages and drawbacks. A com-

mon method is the use of dynamic programming such as in the Needleman-Wunsch (global alignment) and Smith-Waterman algorithms (local alignment). These algorithms are "alignment-based" approaches and they rely extensively on gap penalities. Because they use dynamic programming, they are limited to linear functions for representing gap penalities. As a result, these alignment based approaches do not work well when the protein sequences are dissimilar because there can be too many equally weighted alignments. Also, they can be computationally intensive, especially when taking into account the large databases of sequences that must be analyzed in the ever growing field of bioinformatics. In general, while these methods can provide accurate global and local alignments, they are computationally expensive and therefore not practical in real world use.

Lately, there has been a rise in the number of "alignment-free" sequence analysis approaches. These methods do not depend on gap penalties and are generally much faster to compute. However, as the name suggests, they do not generate an alignment, but only provide a similarity score. One popular version of this approach is the k-mer method which takes all substrings of size k and compares the frequency distribution of those substrings between two protein sequences.

In this paper, we introduce a new 'alignment-free' sequence comparison that was developed by Smale and his colleagues. This approach uses the BLOSUM62 substitution matrix, a statistical-based matrix that tells you the likelihood of two amino acids being replaced by one another. This sequence comparison test computes the 'distance' between two protein sequences by computing the k-mers of all possible k lengths. It only depends on two parameters - k_{max} and β. Most importantly, it is not a function of gap lengths and it satisfies the triangle inequality. In the subsequent sections, we will analyze the effects that k_{max} and β have on the calculated distance, and we will compare this new method to other methods and see if the results are similar.

2 Method

First, we begin by writing a C++ program that implements Smale's method for sequence comparison. The reasoning behind this implementation is as follows. First, we take the BLOSUM62 matrix and note that it is symmetric, positive-valued, and positive definite. These are characteristics

that Smale says the matrix must have in order for these calculations to work. BLOSUM62 is a matrix that was generated by taking many families of proteins which have at most 62% sequence similarity, comparing their sequences, and calculating the likelihood that other amino acids were substituted for that amino acid. This is a statistical measure that gives us a frequency representing how often an amino acid gets substituted for another amino acid in nature. Next, we need to define several kernels which measure the similarity between the protein sequences. Let x and y be two amino acids. Then the kernel K^1 is defined as:

$$K^1(x, y) = BLOSUM62(x, y)^\beta$$

where β is an arbitrarily set parameter which we will investigate later, and $BLOSUM62(x, y)$ is the value found in the BLOSUM62 matrix. We repeat this calculation for every single amino acid in the sequence, comparing each amino acid in the first sequence with each amino acid in the second sequence. Overall, this gives us a score for how similar one letter amino acid sequences are between the two protein sequences. Next, we do a similar calculation for every pair of two, three, ..., (up to) k_{max} letter substring of each protein sequence, where k_{max} is defined as the length of the shorter protein sequence of the two that are being compared. Suppose you have two substrings u and v, each of length of k. Then, we define K^2_k as the product of all the scores of the individual amino acids:

$$K^2_k(u, v) = \prod_{i=1}^{k} BLOSUM62(u[i], v[i])^\beta$$

Since each subsequent product simply adds 1 extra multiplication term, we can define this recursively as:

$$K^2_k(u, v) = BLOSUM62(u, v)^\beta \cdot K^2_{k-1}(u - u[k], v - v[k])$$

We use this recurrence relation in our code to speed up the time complexity of the calculation.

Once we calculated the similarity scores of all possible pairs of substrings of the sequences, we notice that each k-mer gives us new information about the similarity of the two protein sequences. Therefore, we sum together the results of all the K^2_k calculations. This is a two step process.

First, we combine all k-length substring scores as follows, where S represents all k-length substrings in the first sequence and T represents all k-length substrings in the second sequence:

$$K_k^3(S,T) = \sum_{u \in S} \sum_{v \in T} K_k^2(u,v)$$

We can see the implementation of this summation in our code. On line 65 of `SeqComparison.cpp`, the method `SeqComparison::calculateK3_1()` calculates this sumation for amino acid substrings of length 1 and on line 86, the method `SeqComparison::calculateK3_k()` recursively repeats this calculation for the rest of the the k-mers.

Once all of the K_k^3 values have been calculated, we add all of these results together from K_1^3 to K_{k-max}^3.

$$K^3(S,T) = \sum_{k=1}^{k_{max}} K_k^3(S,T)$$

Now, all that is left to do is to normalize K_3. We do this by calculating $K^3(S,S)$ and $K^3(T,T)$ which represent the score for each sequence when it is compared to itself. Once we obtain these values, we obtain the normalized $\hat{K}^3(S,T)$ as follows:

$$\hat{K}^3(S,T) = \frac{K^3(S,T)}{\sqrt{K^3(S,S) \cdot K^3(T,T)}}$$

We can view this result as a dot product. Therefore, we can convert into a distance using the equation below. This is the result that our program outputs:

$$d(S,T) = \sqrt{2(1 - \hat{K}^3(S,T))}$$

Once we completed the C++ program, we then analyzed the output and the parameters of the program. First, we created a data set of proteins, which we used to perform sequence alignment on. We used the website `https://www.uniprot.org` to collect our data set in FASTA format. The protein family that I specifically chose to analyze are called the 'EF hand proteins'. These proteins contain conserved structural and sequence regions which play a critical role in the protein's ability to bind calcium ions. The

'EF hand' is made up of two alpha helices that are joined together by a loop. We randomly sampled 20 of these proteins ($\binom{20}{2}$ sequence comparisons) to create our database for testing Smale's sequence alignment method.

The first test that we conducted analyzed the relationship between β and the computed distance. We compared the results of our distance algorithm to the e-value in the Smith-Waterman algorithm by testing many different β parameters. We used our C++ program to get the distance values of Smale's proposed algorithm, and we used the url `https://fasta.bioch.virginia.edu/fasta_www2/` to retreive the e-values of the Smith-Waterman algorithm. We kept most of the options the same, but we changed FASTA to SSEARCH and we changed the BLOSUM50 matrix to a BLOSUM62 matrix with default -11 gap penalities and -1 gap extension penalities. Once we obtained all the necessary data, we created several scatterplots that plot the values of our distance algorithm (given various β values) as a function of the e-values of the Smith-Waterman algorithm. Next, we removed all the outliers in our data. Using the remaining valid data, we determined the best β value by finding the value that gave the strongest correlation with the Smith-Waterman algorithm.

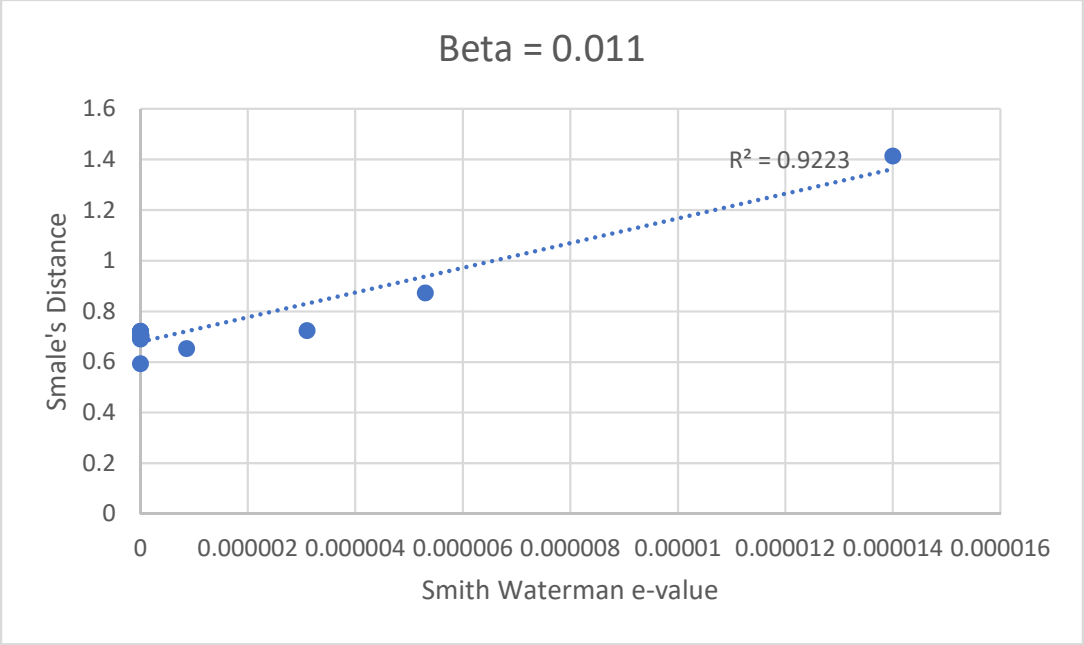
Next, we investigated the effects of k_{max} on our distance algorithm. We took one of the 'EF hand' protein sequences, CABP1_HUMAN, and compared it to pieces of itself. We took sections of the CABP1_HUMAN protein of length $1, 5, 10, 25, ..., 300$, up to k_{max} and we calculated the distance between the original CABP1_HUMAN protein and the substring of CABP1_HUMAN to see if a shorter protein can skew the results of our distance calculations. Once we obtained all the necessary data through similar means as discussed above, we plotted the distances as a function of CABP1_HUMAN substring lengths.

For our final analysis, we took the best β value (that we determined above), and we calculated the distances and e-values for all of the $\binom{20}{2}$ protein sequences. Again, we graphed these results using a scatterplot. By studying the correlation of the two variables, we determined if the relationship between the distance calculation and the Smith-Waterman algorithm was significant.

3 Results and Analysis

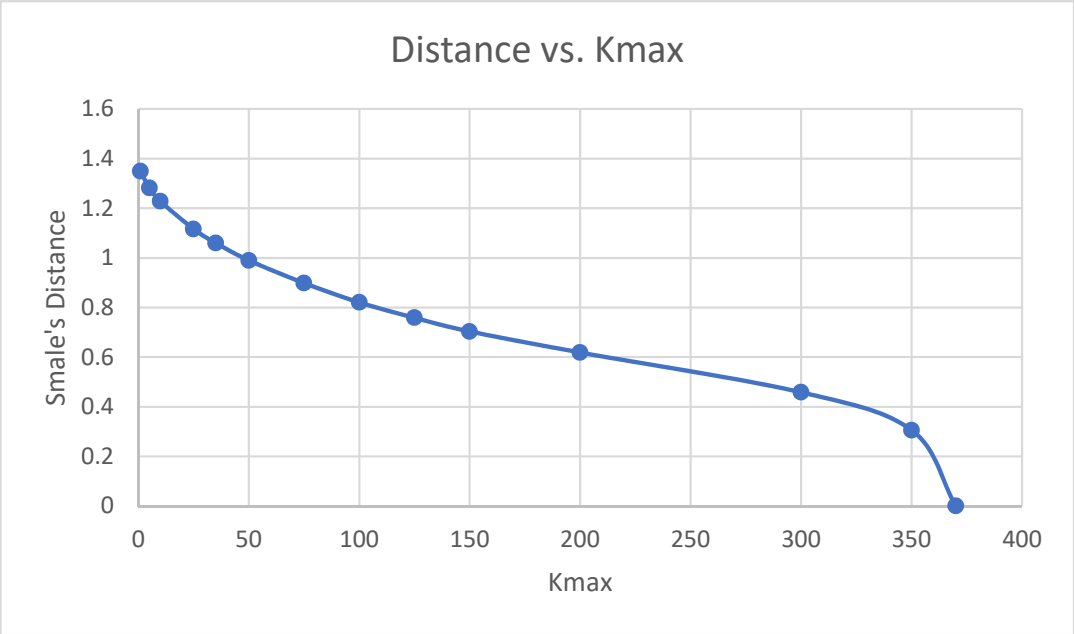
After plotting data with the β parameter set to 0.001, 0.008, 0.009, 0.01, 0.011, 0.02, 0.05, 0.11 and 1, we determined that the best value for β is 0.011 according to our data. It had a reasonable R^2 value = 0.9223. $\beta = 0.01$ was a close second with $R^2 = 0.9147$ and the other β were not very reasonable because R^2 was too low. (All the graphs for this section are available in 'GraphsPart1.pdf' and the data that is being plotted can be found at the Tables section of this report.)

The results that we got are very strange. According to Smale's paper, the optimal β value is 0.11387. The value that we determined is 10 times smaller. When we used 0.11387 as the β parameter, the distance between every sequence in our dataset was 1.41421 giving us no correlation between the Smith-Waterman e-value and our distance algorithm.

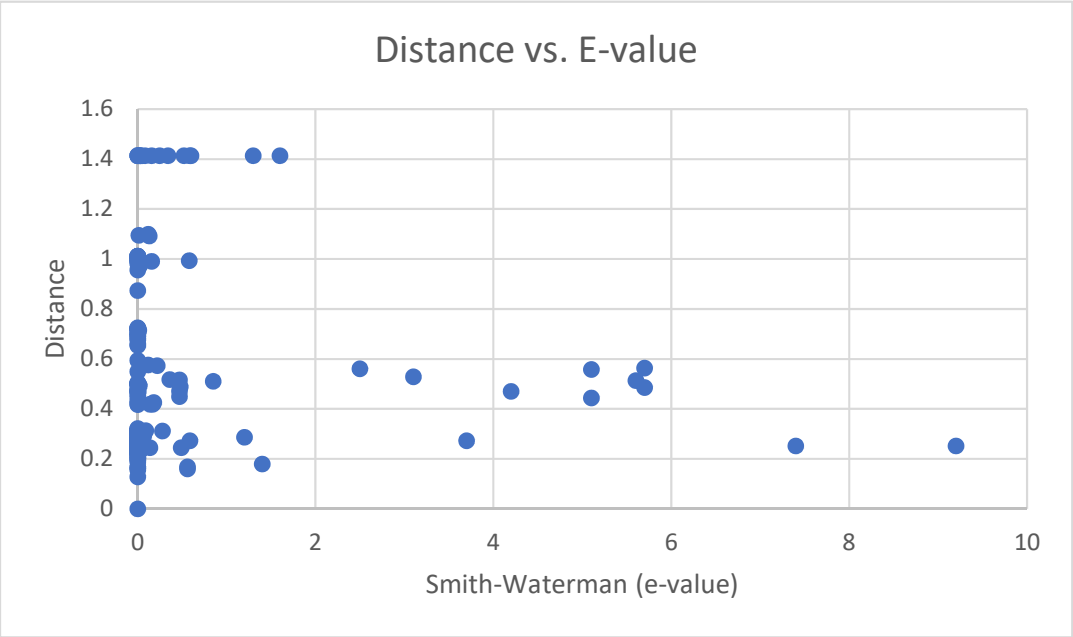


For part two of our analysis, we kept β constant at 0.011 (the best value that we determined from above) and we varied k_{max} with lengths $1, 5, 10, ...300$, 350, and 370. We found that in general, the distance calculation is not affected much by having different protein sequence lengths.

(The data that was used to plot the following graph can be found at Table 2 of the Tables section of the report)



However, this does not mean that using short protein sequences cannot be problematic for this algorithm. Looking at the graph, we notice that it is not entirely linear. Both the head and the tail of the graph have steeper descents than the rest of the graph. If you use this algorithm with two similar protein sequences with different residue lengths, they would get a large distance score by this algorithm despite being similar. Proteins sequences must be sufficiently large enough for the distance result to have any meaning.



For the third part (graph above), I used the SSEARCH website to lookup the e-values for the remaining 190 'EF hand' protein sequence pairs. Using a β value of 0.011 which I determined to be the best for my dataset, I graphed the relationship between between Smale's distance algorithm and the e-value from the Smith-Waterman algorithm. (Table 3 shows the data that is being plotted in the graph below).

The results for this section were very surprising. According to our data, there does not seem to be a clear relationship between our distance algorithm and e-value of the Smith-Waterman algorithm. I was expecting that distance would increase as the e-value increases, since expected value is a measure of the expected number of proteins that have the same score in a database.

Sadly, there is not enough information for me to conclude this relationship. Perhaps the sequences that I was using were not sufficient in demonstrating this relationship, or perhaps the distance equation is taking into account details in the protein sequence that the Smith-Waterman algorithm is not.

4 Discussion

Overall, this algorithm is a step forward in research for new bioinformatic algorithms. However, just like the other algorithms it still has flaws. One

flaw I've noticed is that longer sequences with more than 1000 residues take a really long time to compute. In fact, this algorithm is $O(n^3)$ since we there are three levels of summations. The Smith-Waterman algorithm is more efficient with $O(n^2)$. In today's fast paced world where millions of sequences need to be processed, this may not be an optimal algorithm. Nevertheless, it still has its advantages such as satisfying the triangle inequality which has many important applications.

4.1 Conclusion

In this report, we looked at Smale's distance algorithm and how it compares to other optimal dynamic programming algorithms such as the Smith-Waterman algorithm. We showed that an optimal β range is around 0.01 and we analyzed how the algorithm behaves when the sequence sizes are drastically different. Unfortunately, we were unable to conclude that Smale's distance is correlated with the e-value of the Smith-Waterman algorithm. In the future, we should consider using a larger protein data set, perhaps one with more conserved sequences.

References

[1] Saghi Nojoomi and Patrice Koehl. *String kernels for protein sequence comparisons: improved fold recognition*

[2] Wen-Jun Shen, Hau-San Wong, Quan-Wu Xiao, Xin Guo, and Stephen Smale. *Towards a Mathematical Foundation of Immunology and Amino Acid Chains*

[3] Fasta Database Search.
`https://fasta.bioch.virginia.edu/fasta_www2/`

[4] EF Hand Domain - Wikipedia
`https://en.wikipedia.org/wiki/EF_hand`

[5] Alignment-free sequence analysis - Wikipedia
`https://en.wikipedia.org/wiki/EF_hand`

[6] Heuristic Alignment Algorithms
`http://www.life.umd.edu/labs/delwiche/bsci348s/lec/AlignHeuristic.html`

[7] Secondary Structure Prediction
`http://www.russelllab.org/gtsp/secstrucpred.html`

[8] R.B. Russell, M.J.E. Sternberg. *Structure Prediction: How good are we?*

[9] Protein Family Search Tool.
`http://pfam.xfam.org/family/PF00036`

[10] Dill KA1, Ozkan SB, Weikl TR, Chodera JD, Voelz VA. *The protein folding problem: when will it be solved?*

5 Tables

Table 1: Comparison of Smith-Waterman e-values with various β parameters in Smale's distance

File	S-W:	$\beta = 0.001$	$\beta = 0.008$	$\beta = 0.009$	$\beta = 0.01$	$\beta = 0.011$	$\beta = 0.02$	$\beta > 0.05$
2	1.30E-43	0.24569	0.37282	0.429063	0.502332	0.593654	1.35769	1.41421
3	6.10E-25	0.58827	0.608062	0.631509	0.667655	0.719623	1.3252	1.41421
4	3.10E-06	0.591932	0.613925	0.637316	0.673213	0.724727	1.32571	1.41421
5	3.70E-05	0.468051	0.537099	0.570773	0.618049	0.681541	1.33224	1.41421
6	1.70E-18	0.594684	0.61302	0.635869	0.671282	0.722415	1.32426	1.41421
7	6.20E-24	0.588585	0.610017	0.633494	0.669571	0.721369	1.3251	1.41421
8	1.10E-09	0.525363	0.570372	0.598952	0.640685	0.698455	1.32729	1.41421
9	1.40E-05	1.41421	1.41421	1.41421	1.41421	1.41421	1.41421	1.41421
10	0.084	1.41421	1.41421	1.41421	1.41421	1.41421	1.41421	1.41421
11	3.10E-11	0.525363	0.570372	0.598952	0.640685	0.698455	1.32729	1.41421
12	1.90E-15	0.53005	0.56568	0.593325	0.634406	0.691877	1.32505	1.41421
13	1.70E-08	0.548236	0.579708	0.606217	0.645939	0.701862	1.32626	1.41421
14	1.60E-11	0.552865	0.59241	0.619077	0.658444	0.713471	1.32721	1.41421
15	0.00014	0.450838	0.508338	0.542606	0.591445	0.657411	1.32937	1.41421
16	0.16	0.25543	0.545868	0.676453	0.831348	0.990284	1.41203	1.41421
17	8.60E-07	0.447511	0.503499	0.538046	0.587355	0.653952	1.32989	1.41421
18	5.30E-06	0.821042	0.810624	0.821867	0.841791	0.873351	1.33471	1.41421
19	1.60E-20	0.594204	0.60947	0.632122	0.667495	0.718772	1.32383	1.41421
20	0.00064	0.502521	0.542599	0.572653	0.616785	0.677791	1.32831	1.41421

Table 2: Effects of K_{max} on Distance

K_{max}	Distance	β
1	1.3498	0.011
5	1.28184	0.011
10	1.2281	0.011
25	1.11664	0.011
35	1.06033	0.011
50	0.990361	0.011
75	0.898598	0.011
100	0.820679	0.011
125	0.758759	0.011
150	0.703754	0.011
200	0.618748	0.011
300	0.458538	0.011
350	0.305864	0.011
370	0	0.011

Table 3: Too large to fit in PDF file, attached as Excel document file