CS 466: Introduction to Bioinformatics

Project Report

Group Members

Name	NetID	Email Address
Ritesh Reddy Aluka	rareddy2	rareddy2@illinois.edu
Natalia Ozymko	nozymko2	nozymko2@illinois.edu
Medha Patil	medhap3	medhap3@illinois.edu

Link to GitHub Repository

Introduction

The COVID-19 pandemic has affected the world in so many different, unfathomable ways, and has been dominating the global news ever since its onset. Naturally, there have been a lot of comparisons between the current coronavirus and some of the other coronaviruses the world has seen before. Our key motivation for this project was the potential to explore how similar the coronaviruses actually are through the knowledge we've gained from this class. Hence, we decided to conduct this exploration and compare sequences of some of the proteins they have in common using pairwise sequence alignment - specifically, optimal global alignment based on the Needleman-Wunsch algorithm. Since the viruses stem from the same family, we expected the sequences to be fairly similar to each other. Hence, we chose to use banded alignment as an optimization of the algorithm to improve the running time. Our project discusses the optimal global sequence alignment of the protein sequences in two parts. The first part compares the protein sequences by applying the optimized global alignment algorithm to their genomic sequences. We further explore how changes in bandwidth affect the runtime of said algorithm and consequently, its efficiency in order to determine the significance of using the banded alignment optimization. We extended the project to have a second part, where we constructed a web tool that allows the user to enter a specific pair of nucleotide sequences, a bandwidth, and a scoring function of their choice, and then outputs a visualization of the optimal global alignment score along with its alignment matrix that depicts how banded alignment works on the sequences. The hope is that both parts of this project combined will show why banded alignment improves the running time of global alignment, especially with really large sequences.

Methods

Implementation of Algorithm

The two parts of the project are supported by the global pairwise sequence alignment algorithm, optimized by banded alignment, and based on the Needleman-Wunsch algorithm. This algorithm uses dynamic programming to set up a search space to look for our optimal global alignment. This search space is an alignment matrix that fills up with the highest scores of aligning every possible substring of the given sequences. Once the highest score of the alignment of the complete sequences is obtained, which is the score in the bottom right corner of the matrix, we use a backtrace with the help of a backtrace pointer matrix to find the specific alignment that has resulted in this score. These scores are calculated using a given scoring function, which contains the costs for the insertion and deletion (gap score), mismatch, and match of every possible pair of elements between the sequences.

We decided to use banded alignment to optimize the algorithm, as it is found to be very useful in cases where the sequences are similar, and it is even found to limit errors. This fulfills our purpose as we want to align coronaviruses that are very closely related and have similar genome structure. In banded alignment, we define a band of a given width, and then establish the intersection of this band with our alignment matrix. Alignment scores are calculated only in that intersection, that is, actual computation is limited to the defined band and not the entire alignment matrix. As our search space is reduced considerably, the runtime of the algorithm decreases, making it run more efficiently.

Part I: Comparing different coronaviruses

Datasets

To perform our analyses of how similar the viruses are to each other, we used the following datasets: https://www.ncbi.nlm.nih.gov/datasets/coronavirus/genomes/. Specifically, we used the AlphaCoronavirus, MERS, and SARS CoV-2 datasets. We downloaded the datasets onto our local computer, and then processed and analyzed them on a Jupyter Notebook.

Initial goal

Our initial goal was to compare the genomes of the different viruses and visualize their alignment matrix derived using the algorithm discussed earlier. However, we quickly realized that there were two glaring problems with this idea:

1. It was difficult to visualize the whole genome, as the genomes were enormous in size
2. It was not feasible to run the algorithm on the whole genome on our local computers

Our potential solution to this problem was to extract shorter sequences from the genome and use them instead. However, because we could not find any convincing ways of extracting these shorter sequences, we decided to take a different route: instead of comparing the genomes of the viruses, we decided to compare specific proteins of each coronavirus to visualize how similar/different they are to each other. This approach is explained in detail below.

Preprocessing

Although most of the data was already preprocessed and organized, we had to manipulate and store the data in a way that would make it convenient to use for analyses. We separated the names (and details) of the proteins from the actual protein sequences and stored them in

corresponding arrays such that the indices of the names matched the indices of the actual protein sequences. Needless to say, we created separate arrays for each virus.

Final analysis

As mentioned earlier, our final implementation was focused on comparing specific protein sequences rather than whole genome sequences. We selected a few proteins that the three different viruses had in common: 'M protein', 'N protein', 'S protein', 'ORF3', and 'spike glycoprotein'. However, the Jupyter Notebook can be used to easily test any other proteins by simply adding them to the proteins array. To elaborate on our final analysis, it can be described as a two-part approach:

1. The first part of the analysis was more geared towards analysing the algorithm and how it works with different viruses and different inputs. Specifically, we created graphs to analyze how the time taken for the algorithm differs for different bandwidths. Fig.1 (shown below) depicts a clear relationship between the bandwidth and the time taken to align the respective ‘M proteins’ of the MERS and SARS CoV-2 coronaviruses. Overall, we observed a directly proportional relationship between the bandwidth and the runtime: a trend that was consistent across different viruses and different proteins.

2. The second part of the analysis was focused on comparing the proteins mentioned earlier, across the different coronaviruses. We ran the algorithm on the proteins of one virus and compared their alignment scores with the proteins of a different virus. To visualize and compare these protein alignment scores, we generated heatmaps, an example of which can be seen in Fig.2. To compute these scores, we used the highest bandwidth possible, so these scores are essentially an output of the global alignment algorithm.

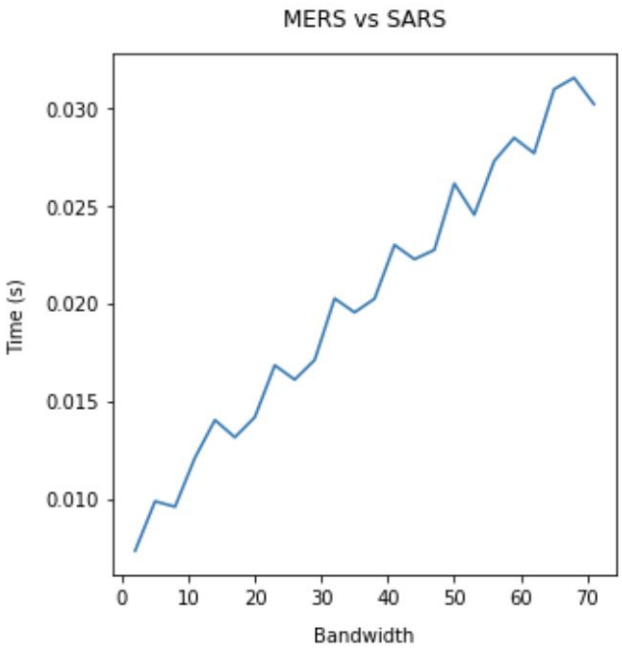

Fig.1: Plot of time against bandwidth for banded alignment of the 'M protein' in MERS and SARS CoV-2

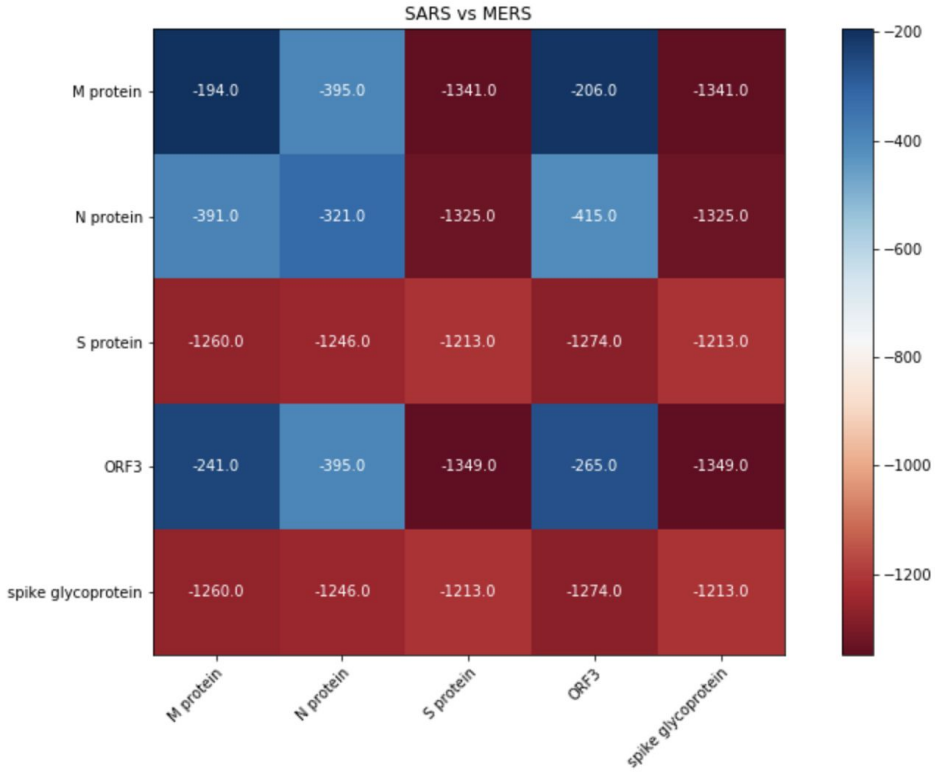

Fig.2: Heatmap comparing some of the proteins that SARS CoV-2 and MERS have in common. The scoring matrix for the algorithm used to generate the heatmaps was: match score = 2, mismatch / gap score = -1

Part II: Visualization

Implementation

The visualization part of the project was implemented using a combination of HTML and JavaScript along with the D3 library. The visualization contains six text boxes: two to input the pair of sequences to align, one to input the bandwidth, and three for the match, mismatch, and gap scores. There is also an "Align Sequences" button that takes the user's input and calculates the alignment by running our algorithm. Once that happens, the output of our algorithm is passed to our visualization, which displays the results of the alignment.

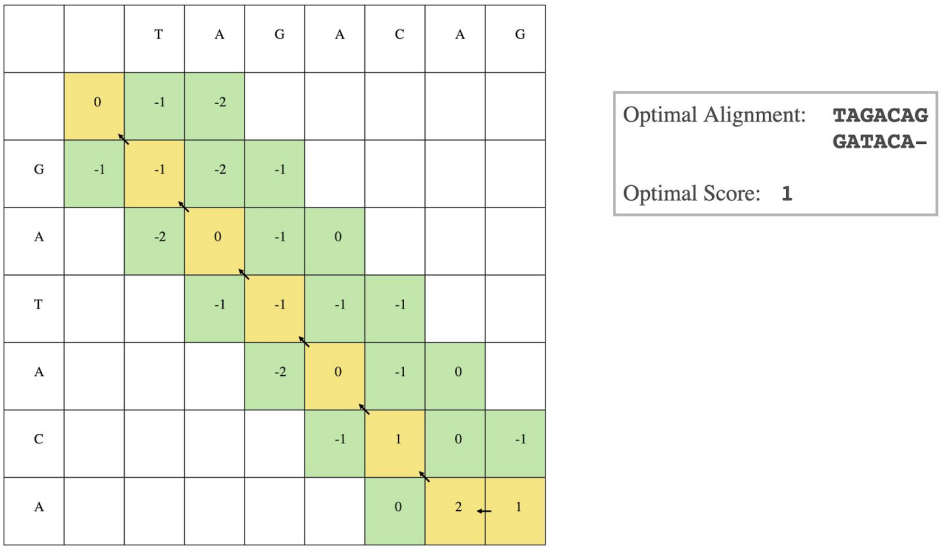

Fig.3: Visualization of our algorithm run with bandwidth = 2, match score = 1, mismatch / gap score = -1

As seen in Fig.3, the alignment matrix is displayed as a grid labeled with the input sequences along the axes. The cells along the band defined by our optimization are highlighted in green, while the final optimal alignment within that band is highlighted in yellow. Each cell in the green band contains the highest score of aligning the subsequences up to that cell. Each cell along the

yellow optimal path contains an arrow identifying which backtrace pointer was optimal. Finally, the visualization outputs a text box which shows the score of the optimal alignment, as well as the pairwise alignment itself.

Validating Results

We verified our algorithm by creating a few of our own test cases, that is, we manually computed the alignment matrices for a few pairs of sequences and compared each with its corresponding alignment matrix outputted by the algorithm. We then compared the output of the algorithm to what was displayed on the page, making sure that the alignment matrix was correctly printed, the width of the band was right, and the optimal path highlighted in yellow accurately matched the optimal alignment of the sequences. We also made sure the direction of the arrows displayed correctly reflected the backtrace pointer matrix. There were several off-by-one errors to fix due to the extra row and column in the grid to account for the sequences themselves. Additional errors occurred while calculating the search space restricted by the bandwidth, due to differing lengths of the sequences as the alignment matrix wasn't always a square with a perfectly diagonal band. However, these issues were easy to track down and debug.

Future Work

Future work for the visualization includes possibly purchasing a domain name to host it, as well as a general cleanup. For the cleanup, the page can be made more user friendly by formatting the text boxes better so that they aren't all in a vertical line at the top of the page. Furthermore, the text box containing the optimal alignment score and the alignment strings could be cleaned up to create a more cohesive layout with the other text boxes. Other possible work includes adding more optimization options to global alignment, so that users can compare how they affect the runtime as well as the final alignment.

Conclusion

In conclusion, our algorithm analysis shows that the banded alignment optimization clearly speeds up the running time of global alignment. This is especially useful when aligning similar, large sequences, as those are less likely to have an optimal alignment outside of the bandwidth, and will hence have a significantly decreased time complexity with a smaller bandwidth. The bandwidth vs runtime graphs, generated in the Jupyter Notebook, justify the benefits mentioned earlier, as they clearly depict a positive correlation between bandwidth and time taken for the algorithm to run. Furthermore, our visualization provides a clear representation of the inner workings of our algorithm. This allows the user to better understand the benefits and drawbacks of using the banded alignment optimization, since it demonstrates how many cells in the alignment matrix were actually used in finding the optimal alignment. Finally, our comparative analysis of protein sequences of different coronaviruses gives an interesting insight into the similarity and differences of their genomic structures. Work like this is increasingly relevant during the ongoing pandemic, and studying the genetic similarity of the SARS CoV-2 virus to other known viruses will provide a better understanding of the virus, enabling us to work towards solutions to resolve this global crisis.