Modified Edit Distance on Global SARS-CoV-2 Analysis

Jenny Fan

Princeton Day School

January 2021

No major assistance was received for this research project, but I greatly appreciate Dr. Andrew Vershon and Dr. Janet Mead at the Waksman Institute, Dr. Matthew Nielpielko at Kean University, and Dr. Jeffrey Heinz at Stony Brook University for giving me the knowledge and tools that this research project uses.

Modified Edit Distance on Global SARS-CoV-2 Analysis

Abstract

In bioinformatics, sequence comparison is used to determine how closely related two sequences of nucleotides or amino acids are by finding the best alignment and taking into consideration insertions, deletions, and substitutions that can happen between two sequences, which is remarkably similar to the Edit Distance algorithm. To improve the accuracy of the algorithm applied to bioinformatics sequence comparison, many researchers have used different weight matrices to change the cost of various edits. A shortcoming of this approach is that it does not reflect random errors that may occur during genetic duplication. Thus, this study proposes and evaluates a new cost function that takes into account the number of consecutive differences. The data used to perform analyses are SARS-CoV-2 sequences from NCBI from several places around the world to determine their relationships. Accuracy is measured by comparing the results from the algorithm with analyses from BLASTP on NCBI and phylogeny trees drawn by MEGAX. The results show that by implementing this cost function, distances between related sequences are more consistent with analyses given by established tools. This suggests that the cost function is an effective alternative approach to improve the accuracy of Edit Distance as applied to bioinformatics sequence comparison.

Introduction

Edit distance is a measure of how far two strings are from each other, and it encompasses a family of different distances that have different rules for the edits allowed between two strings. The most common distance is the Levenshtein distance. It has three types of edits: insertion, deletion, and substitution, all of a single symbol. Typically, each edit has a cost of 1, so the Levenshtein distance can be considered the number of single symbol edits between two strings, but the cost can also be changed to be applied more generally. The most common algorithm implementing the Levenshtein distance uses a computer science method called dynamic programming, and it involves making a table where all of the prefixes of the first string make the row headers, and all of the prefixes of the second string make the column headers. The first column and row can be filled in with however many letters the prefix has because the other string is empty. Then the program goes back to the first empty cell and takes the minimum of the cost of three different operations: insertion (cell to the left + 1), deletion (cell above + 1), and substitution (cell diagonally left, + 1 if the two new letters are not the same). The program does this for each cell, until the final cell gives the desired minimum edit distance. What this essentially does is find the best alignment of the two strings to give the smallest number of edits, which is a brute force way of how humans would solve it by hand. (Kleinberg & Tardos, 2014)

Edit distance is similar to sequence comparison in bioinformatics, as both are finding the best alignment and calculating distances between strings. The algorithms behind tools like BLAST that are commonly used today to search for similar sequences are based off these algorithms that align sequences (Berger, Waterman & Yu, 2020). Thus, research has been done to apply Edit Distance algorithm to bioinformatics by making certain modifications that attempt to encompass the complex factors in biology that go into determining how two sequences are related. A very

common method is using substitution matrices, or scoring matrices, which in the simplest nature, change the weight of substituting between different amino acids according to how likely they are to be aligned with each other. Research has been done to find suitable matrices to find similarities across evolutionary distances (Pearson, 2013).

In DNA replication, random errors may happen during transcription and translation, but they may or may not affect the actual function of the protein. Thus, finding a single character difference between two sequences may not be significant, but longer chains of differences most likely indicates a significant deviation. A substitution matrix does not take into account this factor, so the idea behind this research was to modify the Edit Distance algorithm from this angle instead. This research focuses on finding a mathematical function where given the number of consecutive edits, the function would give a factor to multiply the cost by, and that distance is what is added to the total distance between two sequences. This function is termed as “cost function.” Thus, the research aims to answer the following question: Are cost functions a viable alternative way to modify the edit distance algorithm for the purpose of sequence comparison? This is done by finding a suitable function and then testing to see if it gives results that match those of current bioinformatics tools.

Methods

First, the traditional Edit Distance algorithm was written using Python, setting all the weights to 1 and using a for loop to fill in the values of the table to find the distance between two strings. Then coronavirus sequences were obtained from NCBI, one of SARS-CoV-2 from China, one of SARS, and one of a bat coronavirus. For the purposes of comparison, only the sequences of the ORF1ab polyprotein, the largest gene of the coronavirus, were used. In addition, the amino acid sequence was used instead of the nucleotide sequence because the algorithm compares letter by letter, but multiple nucleotide sequences could code for the same amino acid, so using the amino acid sequence allows for more accurate calculation of distance between two proteins.

The Edit Distance algorithm was then run on the China sequence with the SARS sequence and the bat sequence to give the percent identity. These results were compared with the results given from the BLASTP tool on NCBI. As shown in Figure 1, the two methods give very similar percent identities for China and SARS differing by only 0.27, but they give very different percent identities for China and bat, differing by 7.17. Thus, it was clear that modifications had to be made to the Edit Distance algorithm to improve the accuracy of the algorithm at predicting the closeness of two protein sequences.

Figure 1: Percent Identities between sequences as given by the Edit Distance algorithm and BLASTP

	CHN and SARS	CHN and bat
Edit Distance	86.39%	39.64%
BLASTP	86.12%	46.81%

There were two modifications pursued from here: the substitution weight and a cost function based on the number of consecutive edits as discussed in the introduction. The substitution weight accommodation just involved adding an extra variable to hold that value. To add the cost function, however, required making another table to hold the different edits that could happen in each step (substitution, insertion, and/or deletion). Once that table was filled, the algorithm would

then have to backtrack through the table, going through the steps of the best alignment while counting the number of consecutive edits and using the cost function to appropriately add to the total distance. The cost function multiplies the original distance of the edits (d_0) by a factor based on the number of consecutive edits (x) and the number of consecutive edits allowed (N) to give the final distance (d_f).

Figure 2: Table of percent identities for the quadratic cost function; top value is CHN and SARS, bottom value is CHN and bat

Substitution Weight \ N	1	2	3	4
1	86.387 39.642	91.745 46.269	94.648 53.086	96.256 59.005
1.5	80.510 19.419	89.034 31.759	93.247 43.831	95.628 54.955
2	74.789 3.664	85.950 18.799	91.399 33.753	94.404 47.070

The first function tried was

$$d_f = \begin{cases} d_0 \left(\frac{x}{N}\right)^2 & x < N \\ d_0 & x \geq N \end{cases},$$

a parametric function where the first part quadratically increases from 0 to 1 over the span of N, and after N is a constant of 1. The results of this function are shown in Figure 2, with different substitution weights also factored in. The closest values are when the substitution weight is 1 and the N value is 2, with a value of 91.75 for CHN and SARS and 46.27 for CHN and bat, showing some improvement

The next function tried was a similar parametric function, but with a linear instead of quadratic part:

$$d_f = \begin{cases} d_0 \left(\frac{x}{N}\right) & x < N \\ d_0 & x \geq N \end{cases}.$$

As shown in the results in

Figure 3: Table of percent identities for the linear cost function; top value is CHN and SARS, bottom value is CHN and bat

Substitution Weight \ N	1	2	3	4
1	86.387 39.642	89.959 44.060	92.296 48.887	93.803 53.319
1.5	80.510 19.419	86.193 27.646	89.668 36.260	91.923 44.509
2	74.789 3.664	82.229 13.754	86.739 24.408	89.621 34.354

Figure 3, the closest values are also when the substitution weight is 1 and the N value is 2. This time the values are 89.96 for CHN and SARS and 44.060 for CHN and bat, which are better results than the quadratic function because the distribution of error is more even between the two instead of one being very accurate and the other being very off.

Figure 4: Table of percent identities for the exponential cost function with the exponent constant of 3; top value is CHN and SARS, bottom value is CHN and bat

Substitution Weight \ N	1	2	3	4
1	86.751 40.108	88.169 42.227	89.564 44.852	90.720 47.518
1.5	81.088 20.284	83.307 24.176	85.453 28.921	87.204 33.624
2	75.545 4.725	78.443 9.497	81.237 15.267	83.510 20.918

Given that the linear function performed better than the quadratic function, the next function tried was the exponential function $d_f = d_0\left(1 - e^{-\frac{3x}{N}}\right)$, which is a function that first rises very quickly but then slows down to an asymptote of 1. The constant value of 3 was chosen so as to make the value of the function close to 1 when x is equal to N and align the function with the meaning of N, which is the number of consecutive edits allowed. The results are shown in figure 4, and this time the closest values are when the substitution weight is 1 and the N value is 3. The values are 89.564 for CHN and SARS and 44.852 for CHN and bat, which are both slightly closer to the BLASTP values than those given by the linear function.

Due to the better results using the exponential function, more tweaks were performed with this function by changing the constant coefficient in the exponent to try to raise the CHN and bat value while keeping the CHN and SARS value from increasing too much. The closest attempt is shown in Figure 5, where the substitution weight of 1 and N value of 3 give a value of 90.286 for CHN and SARS and 46.459 for CHN and bat. This shows the limits of this exponential cost

function, as lowering the constant certainly raises the value for CHN and bat as desired, but also raises the CHN and SARS value significantly. The tables for other constant values can be found in Appendix.B.

Figure 5: Table of percent identities for the exponential cost function with the exponent constant of 2.5; top value is CHN and SARS, bottom value is CHN and bat

Substitution Weight \ N	1	2	3	4
1	86.997 40.440	88.754 43.255	90.286 46.459	91.487 49.582
1.5	81.476 20.898	84.211 26.046	86.550 31.772	88.353 37.165
2	76.053 5.479	79.620 11.778	82.661 18.700	84.999 25.133

Thus, the goal was to find a function that would raise the value of CHN and bat while simultaneously keeping the CHN and SARS value low. Following the trend set by previous functions, this was done by creating functions that had the same or higher cost values for low values of x while increasing much more gradually so that larger x would have lower cost. After playing around with different functions, the final family of functions all follow the rational form: $d_f = d_0\left(1 - \frac{a}{(x+b)^c}\right)$ with a, b, and c being constants adjusted to form the desired shape or make the function pass through desired values.

The first function tried was one with a = 0.74, b = 1, and c = 1. The numerator was found by choosing a constant to make the value of the cost function at 1 about the same as that of the first exponential function with N = 3 because that function is close to the values wanted and thus a good starting point. Figure 6 shows the results from this function, the best column is with a substitution weight of 1, with values of 90.34 for CHN and SARS and 50.39 for CHN and bat.

Figure 6: Table of percent identities for the rational function with a = 0.74, b = 1, and c = 1

	Substitution Weight		
	1	1.5	2
CHN and SARS	90.335	86.443	82.497
CHN and bat	50.389	36.523	24.131

Both values are too high, but compared to the original exponential function, this shows that this function is able raise the CHN and bat value while barely raising the CHN and SARS, which is desirable. Thus, the next functions were aimed at raising the cost in some way to lower both values.

One direction was to maintain the cost at 1 but raise the cost for higher x values. For example, one function tried a = 1, b = 0.65, c = 2. The b value was found by shifting the graph until the value at 1 was about the same as the previous function. As shown in Figure 7, the best results were with a substitution weight of 1, with a value of 89.657 for CHN and SARS and a value of 45.75. These values showed that the changes made lowered the CHN and bat value much more than the CHN and SARS value, which was undesirable.

Figure 7: Table of percent identities for the rational function with a = 1, b = 0.65, and c = 2

	Substitution Weight		
	1	1.5	2
CHN and SARS	89.657	85.566	81.379
CHN and bat	45.749	30.143	16.652

The other direction was to raise the cost at 1 but lower the cost for higher x by making the graph raise even more gradually. This was done by lowering the exponent in the denominator and shifting the graph over. An example of a function tried was with a = 1, b = 7, and c = 0.8. The results are shown in Figure 8, with the best column from substitution weight 1 and with values of 88.77 for CHN and SARS and 48.51 for CHN and bat. These results show that this direction is indeed promising because this function managed to lower both

Figure 8: Table of percent identities for the rational function with a = 1, b = 7, and c = 0.8

	Substitution Weight		
	1	1.5	2
CHN and SARS	88.775	83.993	79.301
CHN and bat	48.513	32.228	18.917

values by a similar amount instead of disproportionately affecting one but not the other.

Thus, the exponent value was adjusted more, and the results of all of those functions can be found in Appendix.C. Ultimately, the function settled for was with a = 1, b = 75, and c = 0.5. The results of this function are shown in Figure 9, the best values in the first column with substitution weight. This function gives a value of 87.94 for CHN and SARS, and a value of 46.41 for CHN and bat. These values are very close to those given by BLASTP, the CHN and SARS off by 1.82 and the CHN and bat value off by only 0.43. These values are clearly much closer than those given by the original Edit Distance algorithm.

Figure 9: Table of percent identities for the final rational function with a = 1, b = 75, and c = 0.5

	Substitution Weight		
	1	1.5	2
CHN and SARS	87.938	82.736	77.668
CHN and bat	46.407	28.530	14.547

Thus, the final function settled on was $d_f = d_0\left(1 - \frac{1}{(x+75)^{0.5}}\right)$. Multiple tests were then performed to test the accuracy of the modified Edit Distance algorithm on other sequences. To do this, SARS-CoV-2 sequences were retrieved from NCBI from Italy, California, New York, Texas, Australia, Russia, and Brazil (Accession numbers can be found in Appendix.A). These sequences, along with the China sequences, were compared with each other using the modified Edit Distance algorithm. These sequences with the bat sequence were also run on MEGAX (the sequences that were identical were only put in once), a bioinformatics tool that aligns sequences and can draw phylogeny trees. The results from the modified Edit Distance algorithm were then compared with the phylogeny trees (rooted with the bat sequence) to see if they matched.

Results

The table in Figure 10 shows the distances calculated using the modified Edit Distance algorithm. The sequences from New York and Texas were identical, as were Russia and Italy, so only New York and Italy out of the four were included for MEGAX to draw phylogeny trees.

Figure 10: Table of distances between SARS-COV-2 sequences around the world

	CHN	ITA	CA	NY	TX	AUS	RUS	BRA
CHN		0.885	1.771	1.771	1.771	3.541	0.885	1.771
ITA	0.885		2.656	0.885	0.885	4.426	0	2.656
CA	1.771	2.656		3.541	3.541	5.312	2.656	3.541
NY	1.771	0.885	3.541		0	5.312	0.885	3.541
TX	1.771	0.885	3.541	0		5.312	0.885	3.541
AUS	3.541	4.426	5.312	5.312	5.312		4.426	5.312
RUS	0.885	0	2.656	0.885	0.885	4.426		2.656
BRA	1.771	2.656	3.541	3.541	3.541	5.312	2.656	

The trees drawn by MEGAX are shown in Figures 11.a and 11.b, the first one using the Maximum Likelihood Method and the second using the Maximum Parsimony Method.

Figure 11.a: Phylogeny tree drawn by MEGAX using Maximum Likelihood Method

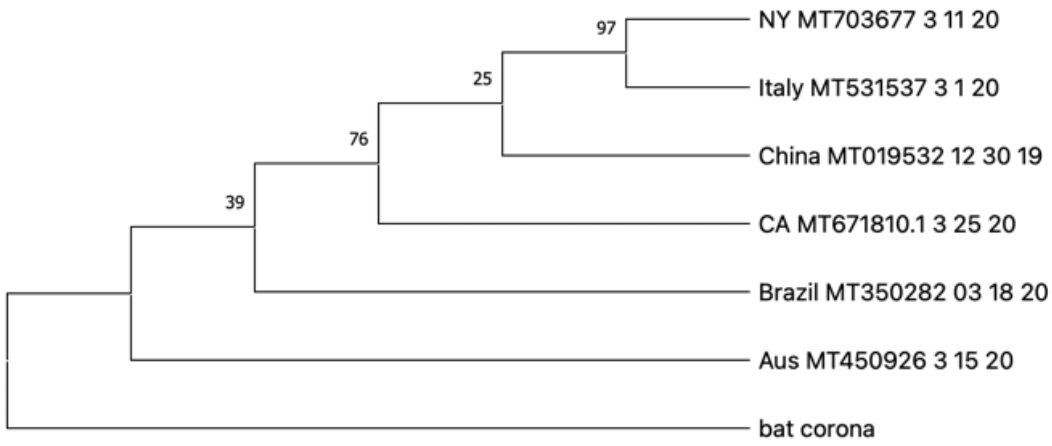
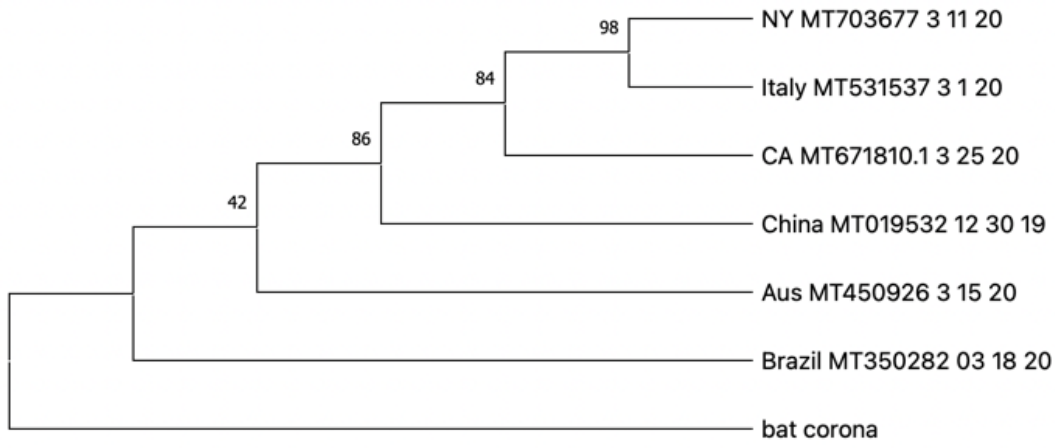


Figure 11.b: Phylogeny tree drawn by MEGAX using Maximum Parsimony Method



The phylogeny trees show the relationships between the sequences, meaning the further away two sequences are on a tree, the less related those two sequences are, or the earlier they branched away from a common ancestor. Thus, two sequences that are closely branched together, such as New York and Italy in this case, should have a relatively small distance between the two sequences compared with, say, New York and Australia. The numbers on each node represent the confidence of that branching drew by the algorithm. The bat sequence was run with the human sequences in order to root the tree at a sequence that is far away from the rest so that sensible relationships between the other sequences can be drawn.

By looking at the distances between the sequences, one can see if the data given by the algorithm matches either of the trees. For New York, the order of sequences from closest to furthest is Italy, China, California = Brazil, and Australia (ignoring the duplicates that aren't in the trees), which exactly matches the order of sequences for Maximum Likelihood. The distances for other sequences also match this. In addition, the distances for California and Brazil are the same down the column despite them being different from each other. On the Maximum Likelihood tree, the confidence value between the split of California and Brazil was only 39, which means that the branch was not drawn with high confidence.

Discussion and Conclusions

The results show that the modified Edit Distance algorithm indeed gives results that significantly match those given by other tools. The order of sequences from closest to furthest from New York on the Maximum Likelihood tree exactly match the ordering of distances in increasing order. The low confidence of the split between California and Brazil parallel the equal distances of the two from all other sequences, as a similar distance might make it hard to distinguish which sequence came first. All of this suggests that the algorithm is similar to the Maximum Likelihood method of determining sequence distance and relationships, and thus the cost function is a valid alternative approach to improve the accuracy of the Edit Distance algorithm as applied to bioinformatics.

It is clear that the algorithm does not entirely match the results given by MEGAX. The results given by the algorithm do not explain the low confidence between Italy and China, and they also do not give any insight into the variation of confidence values. This is likely because the cost function focuses on one factor, the number of consecutive edits, when there are many other factors that go into determining the distance between sequences. In addition, the end results for making the cost function itself could be improved, as the percent identity between China and SARS was further from the percentage given by NCBI than the percent identity given by the original algorithm was. The numbers could be tweaked to be more precise, or perhaps an entirely new function shape would be needed in order for the algorithm based on a cost function to be more accurate.

Further Work

If this research were to be continued, several directions could be taken.

Firstly, as previously mentioned, the cost function could be researched further to give more accurate results. An idea for a new cost function could be to lower the cost of a single edit on its own, raise the cost for a certain number of consecutive edits, but after a certain point the cost wouldn't change much. The reasoning behind this is that for lower amounts of edits, for example 5 edits versus 10 edits, the difference is much more significant than, say, 60 edits and 65 edits, so perhaps a function that pays attention to that nuance might fare better, especially with the China and SARS percent identity.

In addition, the cost function only takes into account the number of consecutive edits, which is an oversimplified model of how relationships between sequences are actually determined. If instead, it combined weight matrices containing different substitution weights for between different amino acids with the current cost function, that would likely give more accurate results. Other factors could also be researched and incorporated to increase the accuracy of the algorithm.

Finally, a different Edit Distance could be considered. This program uses the Levenshtein distance, where the only edits are insertion, deletion, and substitution, but a modified version of this, called the Damerau–Levenshtein distance also adds transposition as another type of edit. This could be implemented instead to test if the algorithm gives better results.

Literature Cited

[1] Kleinberg, Jon, and Eva Tardos. Algorithm Design. Pearson Education, 2014.

[2] Berger, Bonnie & Waterman, Michael & Yu, Yun. "Levenshtein Distance, Sequence Comparison and Biological Database Search." IEEE Transactions on Information Theory (2020): PP. 1-1. doi:10.1109/TIT.2020.2996543.

[3] Pearson, William R. "Selecting the Right Similarity-Scoring Matrix." Current protocols in bioinformatics vol. 43 (2013): 3.5.1-3.5.9. doi:10.1002/0471250953.bi0305s43

Appendix

A. Accession Numbers of Sequences

Made sure all sequences didn't have X's (unknowns) in them

Accession Number	Location	Accession Number	Location
MT019532	China	MT614507	USA, TX
MT531537	Italy	MT450932	Australia
MT671810	USA, CA	MT510643	Russia
MT703677	USA, NY	MT350282	Brazil

Bat: MG916903

SARS: P0C6X7.1

B. Exponential Function, Extra Data Tables

Constant = 4

Substitution Weight \ N	1	2	3	4
1	86.519 39.807	87.420 41.045	88.538 42.864	89.564 44.852
1.5	80.720 19.727	82.141 22.012	83.878 25.336	85.453 28.921
2	75.063 4.041	76.921 6.846	79.187 10.914	81.237 15.267

Modified Edit Distance on Global SARS-CoV-2 Analysis

Constant = 3.25

Substitution Weight \ N	1	2	3	4
1	86.669 40.000	87.938 41.847	89.263 44.234	90.389 46.704
1.5	80.958 20.085	82.949 23.482	84.993 27.813	86.706 32.204
2	75.375 4.481	77.976 8.648	80.638 13.925	82.864 19.218

Constant = 2.75

Substitution Weight \ N	1	2	3	4
1	86.858 40.250	88.438 42.688	89.903 45.583	91.084 48.467
1.5	81.257 20.547	83.724 25.018	85.970 30.225	87.751 35.263
2	75.766 5.049	78.986 10.524	81.907 16.840	84.219 22.872

C. Rational Functions, Extra Data Tables

1 - 1.11/(x+2)

	1	1.5	2
SARS	90.549	86.713	82.840
bat	52.052	38.706	26.687

1 - 0.37/x

	1	1.5	2
SARS	89.958	85.956	81.876
bat	47.844	32.979	19.974

Modified Edit Distance on Global SARS-CoV-2 Analysis

1 - 1/(x+0.4)^3

	1	1.5	2
SARS	89.286	85.060	80.733
bat	43.963	27.308	13.304

1 - 0.5/(x+1)

	1	1.5	2
SARS	89.055	84.519	79.997
bat	46.904	30.976	17.493

1 - 1/(x+5)^0.8

	1	1.5	2
SARS	89.335	84.827	80.383
bat	50.182	34.849	22.040

1 - 1/(x+10)^0.8

	1	1.5	2
SARS	88.271	83.248	78.334
bat	46.901	29.754	15.974

1 - 1.5/(x+15)^0.8

	1	1.5	2
SARS	88.516	83.592	78.778
bat	48.139	31.341	17.870

1 - 1/(x+12)^0.75

	1	1.5	2
SARS	88.281	83.256	78.344
bat	47.119	29.958	16.220

Modified Edit Distance on Global SARS-CoV-2 Analysis

1 - 1/(x+14)^0.7

	1	1.5	2
SARS	88.352	83.352	78.468
bat	47.545	30.467	16.831

1 - 1/(x+13)^0.79

	1	1.5	2
SARS	88.000	82.849	77.817
bat	46.017	28.403	14.367

1 - 1/(x+16)^0.75

	1	1.5	2
SARS	87.952	82.773	77.718
bat	45.957	28.238	14.174

1 - 1/(x+20)^0.7

	1	1.5	2
SARS	87.956	82.774	77.718
bat	46.101	28.357	14.321