Popcorn: an alignment-free metric distance calculator for protein sequences

Abigaela Boroica[1], Lisa Malins[2]

[1]Department of Computer Science, University of California, Davis
[2]Department of Biotechnology, University of California, Davis

March 13, 2020

1 INTRODUCTION

Alignment of two protein sequences is a standard way to determine how similar they are. A scoring matrix such as the PAM or BLOSUM matrices evaluates pairing a single amino acid with itself or with another. In addition to scoring matrices, the score of an alignment depends on gap penalties. This may be a constant penalty for a gap of any length, a penalty per amino acid, or a larger penalty for opening a gap and a lesser penalty for each additional amino acid. These scoring schemes are arbitrary, which prevents alignment scores from satisfying the triangular inequality.

In 2012, Shen et al. introduced a more exact mathematical method of calculating the distance between two amino acid sequences based on kernels.[1] Our program "Popcorn" implements this method in a simple C++ program.

2 METHODS

2.1 Assumptions

The program assumes that the user inputs a positive real number for β; it will terminate if the user inputs a wrong data type or a negative number. For the purpose of this project, all three given protein sequences are stored in the program instead of read from an external file. The algorithm was designed for protein sequences written with capital letters. It also assumes that the values read from the BLOSUM62 matrix are positive decimal numbers.

2.2 Kernels

Shen et al. define a kernel K as a symmetric function $K : X \times X \rightarrow \mathbb{R}$ where X is a finite set. This means the kernel accepts two arguments from the same finite set, computes the same value with either order of the two arguments, and the result is a real number.

For amino acid sequence comparison, Shen et al. define kernels at three levels: individual amino acids, amino acid sequences of length k, and amino acid sequences of arbitrary length.

Kernel K^1 operates on two amino acids, x and y. It computes the value for (x, y) in the BLOSUM62-2 matrix raised to the parameter β, which Shen et al. state is typically set to $\frac{1}{8}$ or $\frac{1}{10}$.[1]

$$K^1(x,y) = ([BLOSUM62-2](x,y))^{\beta}$$

To avoid re-calculating this kernel, Popcorn computes this kernel for every amino acid pair and stores the result in a 20 by 20 matrix..

Kernel K^2 operates on two amino acid sequences u and v. These sequences must be the same length k so, they may also be called k-mers. Kernel K^2 calculates kernel K^1 at each position i along the two k-mers and multiplies the results together.

$$K_k^2(u,v) = \prod_{i=1}^{k} K^1(u_i,\ v_i)$$

Kernel K^3 operates on two amino acid sequences f and g of arbitrary length. To accommodate sequences of different lengths, K^3 considers subsequences u and v of length k, where k starts at 1 and grows to the length of the shorter sequence.

$$K^3(f,\ g) = \sum_{u \subset f,\ v \subset g} K_k^2(u,v)$$

$$\text{For } |u| = |v| = k,\ all\ k = 1,\ 2...$$

Popcorn implements a helper function that generates these substrings for both protein sequences and saves them in a vector. This function supports a limit on maximum value of k, which is an option to reduce the runtime on long protein sequences.

Finally, correlation kernel $\widehat{K^3}$ is a normalized form of kernel K^3.

$$\widehat{K^3}(x,\ y) = \frac{K^3(x,y)}{\sqrt{K^3(x,x),\ K^3(y,y)}}$$

2.3 Algorithm

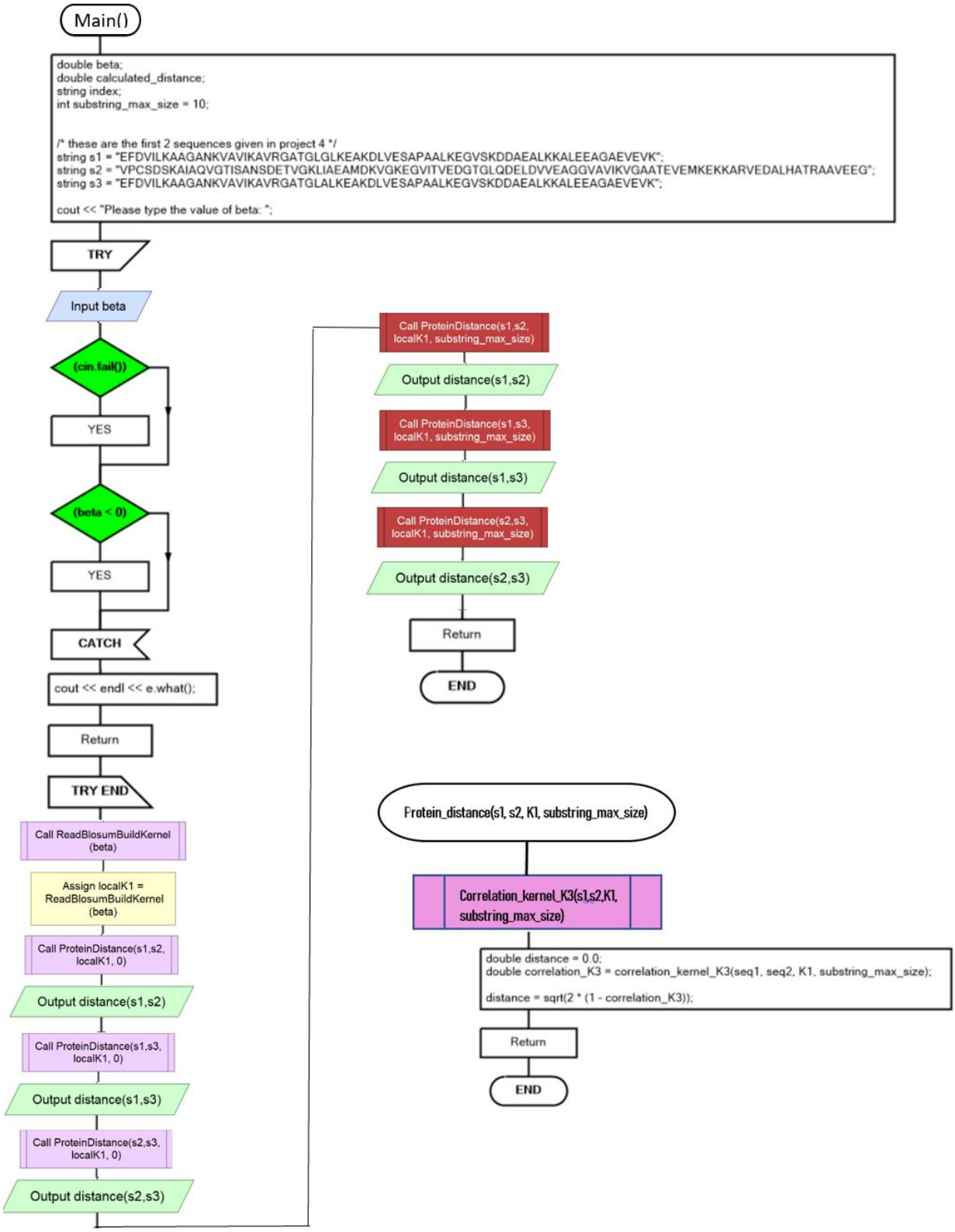

Figure 1. Graphic/logic representation of main() and protein_distance() methods

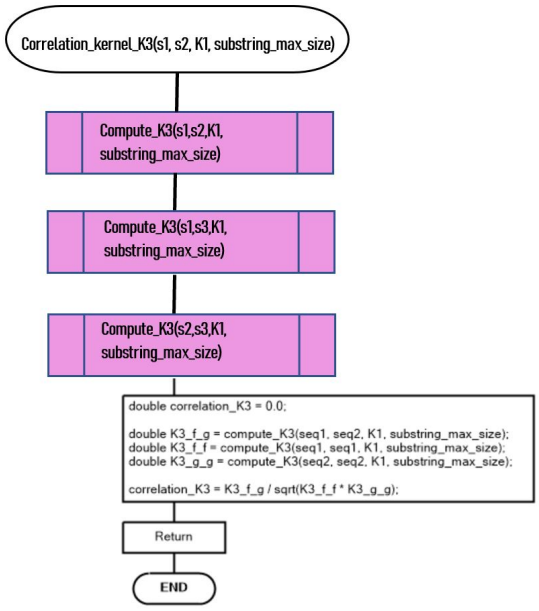

Figure 2. Graphic/logic representation of correlation_kernel_K3() method

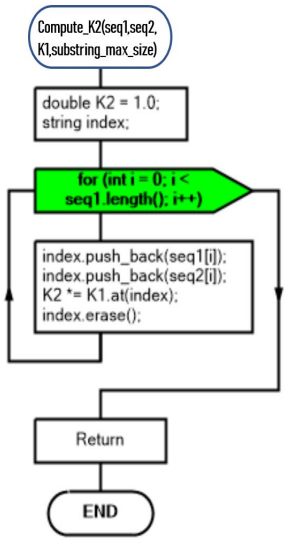

Figure 3. Graphic/logic representation of compute_K2() method

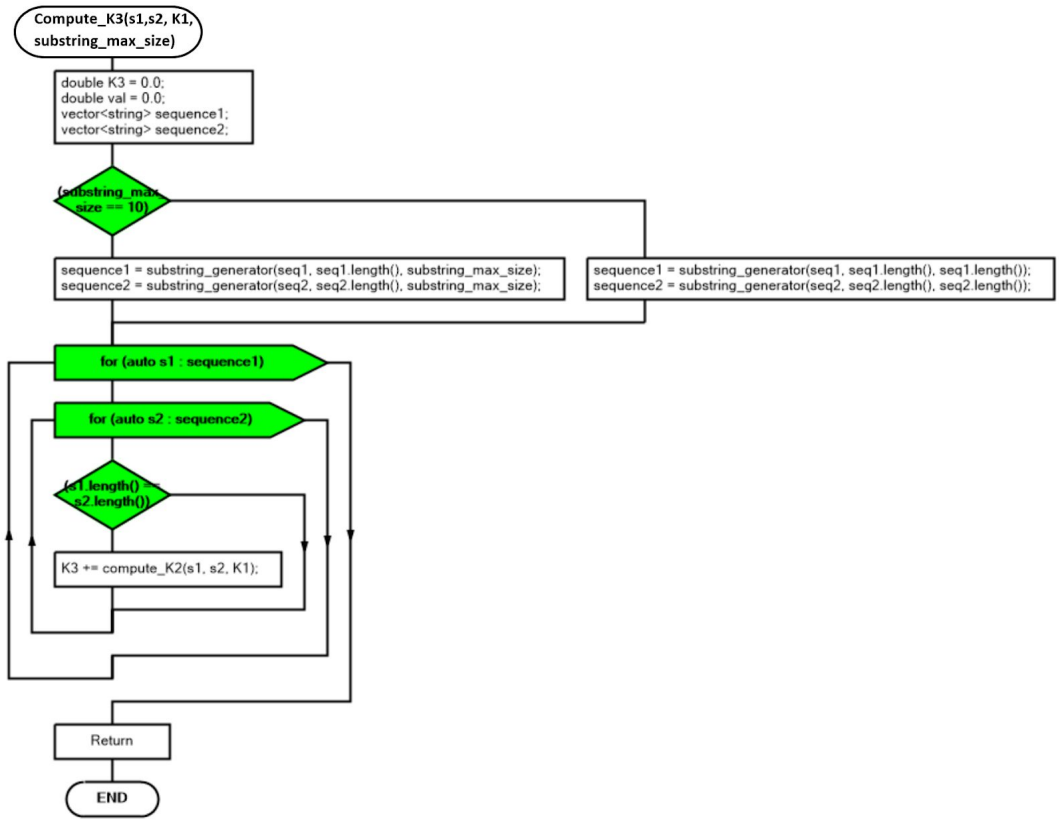

Figure 4. Graphic/logic representation of compute_K3() method

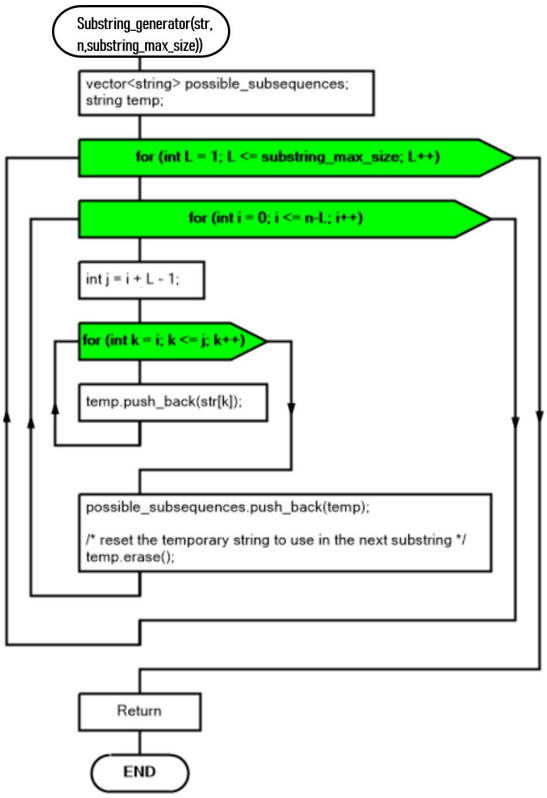

Figure 5. Graphic/logic representation of substring_generator() method

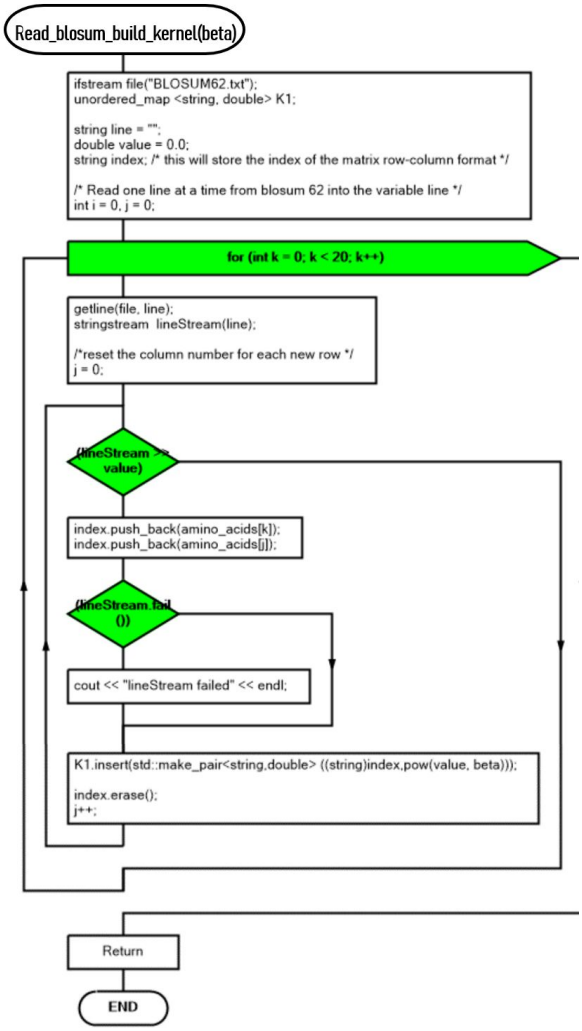

Figure 6. Graphic/logic representation of read_blosum_build_kernel() method

3 RESULTS

We tested Popcorn on three example sequences. Sequence 1 is from the 50S ribosomal protein L7/L12 in *E. coli*.[2] Sequence 3 differs from Sequence 1 by only one amino acid. Sequence 2 is from the chaperonin GroE from *E. coli*.[3]

Table 1 displays the distance results from Popcorn for each pair of sequences. Popcorn finds the same distance between sequences regardless of order, and finds the distance to be 0 when both sequences are the same.

Table 1. Popcorn distance calculations between 3 example amino acid sequences

	seq1.fa	seq2.fa	seq3.fa
seq1.fa	0	0.278569	0.0245591
seq2.fa	0.278569	0	0.279908
seq3.fa	0.0245591	0.279908	0

For long amino acid sequences, the full Popcorn algorithm is prohibitively slow. To reduce the runtime, the algorithm can be halted after considering a maximum subsequence length. Since modification reduces the number of kernel K^2 results that are summed, the final distances are smaller. However, the proportion of the distances to each other is mostly maintained, with some distortion as the max substring length decreases.

Table 2. Popcorn distance results with different maximum substring lengths

Maximum substring length	seq1–seq2		seq1–seq3		seq2–seq3	
	Distance	Ratio	Distance	Ratio	Distance	Ratio
Unlimited	0.27857	0.48	0.02456	0.04	0.27991	0.48
100	0.27857	0.48	0.02456	0.04	0.27991	0.48
90	0.27819	0.48	0.02456	0.04	0.27953	0.48
60	0.19910	0.47	0.02399	0.06	0.20104	0.47
30	0.09442	0.46	0.01526	0.07	0.09730	0.47
10	0.04265	0.45	0.00696	0.07	0.04454	0.47

Notes: The ratio column is the proportion of that pair's distance to the sum of all three distances, to show the effect of max substring length on the relative distances.

4 DISCUSSION

Developing a method that calculates a true metric distance between protein sequences is useful to compare protein structures and find an optimal match in terms of shape without relying on sequence alignment or considering alignment gaps. The metric distance provides more precise information on protein folding, increasing the accuracy of protein function prediction.

For the purpose of this project we used a coefficient $\beta = 0.01$; according to Saghi Nojoomi and Dr. Patrice Koehl, using small β coefficients yields a satisfactory outcome of this algorithm independent of the k-mers length and in the context of remote homologs. However, in the context of protein folding recognition, the results depend on β coefficients much larger than 0.01 and are optimal when using k-mers of maximum length of 10.[4] Our algorithm supports changes in both parameters.

One possible limitation of our program is storing the sequences in a string data structure which has a limit of 2^{32} characters. However, based on the context, it is unlikely that a protein sequence would exceed this length. Also, the program assumes each letter represents one of the standard 20 amino acids. Therefore, the program does not support unusual amino acids or modified amino acids.

An advantage that protein aligners have over our program is that they display results visually. For example, BLAST outputs a pairwise alignment showing identities and positives between the query and subject sequences. Humans tend to understand data more easily if they can visualize it. In the future, Popcorn results could be sent to a different program that visually represents distances between proteins and their structures.

Another limitation is that Popcorn is unsuitable for finding matches between a query and a large database of protein sequences. The time complexity would be too great. An additional feature would be to support database queries by first shallowly comparing a query to database sequences. If the distance is immediately large, the pair could be discarded, but if the preliminary distance is under a certain threshold, the full algorithm could more accurately calculate the distance.

The efficiency of our program could be improved by storing previously seen K^2 scores between k-mers in a database. When calculating K^3 using the same β on k-mers already in the database, retrieving previously calculated K^2 scores would reduce the runtime from quadratic to linear time complexity. However, this would increase the auxiliary space used by the program and would require execution of the algorithm on many sequences to create an extensive database.

REFERENCES

1. Shen WJ, Wong HS, Xiao QW, Guo X, Smale S. Towards a mathematical foundation of immunology and amino acid chains. arXiv preprint arXiv:1205.6031. 2012 May 28.
2. Tyson GH, Li C, Hsu CH, Bodeis-Jones S, McDermott PF. Diverse Fluoroquinolone Resistance Plasmids From Retail Meat E. coli in the United States. *Frontiers in Microbiology*. 2019;10:2826.
3. Walter S. Structure and function of the GroE chaperone. *Cellular and Molecular Life Sciences CMLS*. 2002 Oct 1;59(10):1589-97.
4. Nojoomi S, Koehl P. A weighted string kernel for protein fold recognition. *BMC bioinformatics*. 2017 Dec;18(1):378.