Probablistic BLAST Algorithm

Riley Ballachay
riley.ballachay@mail.mcgill.ca

Xueyang Zhang
xueyang.zhang@mail.mcgill.ca

Haowei Qiu
haowei.qiu@mail.mcgill.ca

Abstract

With the recent biotechnological advancement, aligning a query to a probabilistic genome database becomes an immediate need. By proposing modifications including a new substitution scoring function, we modified the BLAST algorithm that works on this new setting. By evaluating local alignments and comparing different scoring functions, we verified that our proposed algorithm can properly do local alignments with similar amount of time and memory requirement as BLAST.

1 Introduction

Owing to recent biotechnology advancements in sequencing, there has been an explosion of sequencing in different omics, including DNA, RNA, and ATAC, as we can see in Heather & Chain (2016), Wang et al. (2009), Buenrostro et al. (2015), Jovic et al. (2022). While far more cost effective and magnitudes higher in throughput, current sequencing methods are still error-prone and limited to sequencing of length a few hundred bp. Computational biology has become indispensable in ameliorating ordinary laboratory workflows and gaining maximum insight from biological data. One core task is sequence alignment, like in Chao et al. (2022). It captures the shared evolutionary origin of two sequences S and T. We had some common assumptions based on evolution, including that mutations are rare, and indels are rarer than substitutions. We encode these into scores so that one likely alignment corresponds to a high score. We can find the optimal solution for this task.

However, finding the optimal solution is too slow. We typically have S as the genome of all species, which is matched for thousands of different T's. We can use BLAST, but it is heuristic-based and does not guarantee optimality. Meanwhile, it is more precise to treat genome as probabilistic because the genome of a species is actually a pool. To incorporate this change, we modified BLAST for this setting.

2 Literature Review

For the global optimal alignment problem, we have the Needleman-Wunch(NW) algorithm by Needleman & Wunsch (1970). It inputs a substitution matrix M and a gap penalty. Then we fill the table X in DP style as $\max(X_{i-1,j-1} + M(S_i, T_j), X_{i-1,j} + c, X_{i,j-1} + c)$. Then, we trace back from the bottom right to get the optimal alignment. The time and space complexity is $O(\text{len(S)} * \text{len(T)})$. For local alignment, we have the Smith-Waterman algorithm by Smith & Waterman (1981), which is just a few adjustments to the NW algorithm, including having an option of 0 in table filling and tracing back from the maximum value in the table.

The above two algorithms are too slow in real-life scenarios. One much faster but heuristic way is BLAST by Silva e Santos (2012). First, we do a sliding window of w = 11 in S and record the indices in a dictionary of w-mer index lists. Second, for each w-mer in T and each index in the corresponding index list, we do an ungapped extension to filter out low-score ones. Third, we do a gapped extension just like NW to allow gaps to potentially make the match longer. We will do a modification on this method to accommodate the new problem that S is probabilistic.

3 Method

3.1 Task Description

Though the genome of an individual is deterministic, the genome of a population is not deterministic. First, for a certain time point, there is population diversity. Second, we cannot match to an ancestral genome which can only be probabilistic. Only taking the most likely nucleotide at each position of the genome ignores the other sometimes non-negligible possibilities which makes the match inconclusive.

The task is to find aligned sub-sequences. S is the genome(G) of a species, typically having a length of millions or billions. T(q/query) is a query DNA, meaning it can be a DNA sequence of any species. It typically has a length of a few hundred. We want to find the highest-score local alignment of S and query.

3.2 Dataset

We now treat G as a probability table of size len(G) by 4. Similar to allele, we can define major nucleotide as the most likely one, and minor the other three. In this particular case, all 3 minor nucleotides have an equal probability for an arbitrary position, but the method described below works in the general case. The dataset we are working on is chromosome 22 of the predicted BoreoEutherian ancestor genome, discussed in Muffato et al. (2023). G has a length of 604466, and the query can be anything. We simulated the query as a subsequence of G of certain length with a probabilistic number of indels and mismatches.

3.3 Indexing the Genome

In BLAST, we had the observation that "if q has a good alignment with G, then q and G are likely to contain at least one perfect match of length at least w, where we often take w = 11". There are counter-examples, but it covers the most cases. Because the underlying problem remains unchanged, the idea that "they should have perfect/very good match" should remain the same. Note that, the probabilistic setting is a more general setting for the previous deterministic setting. As a result, we expect that, in the degenerate case for the probabilistic genome, the behaviour and the result of this step should be the same.

The different part is that the notion of perfect match needs to translate into the probabilistic setting. In this setting, a pure perfect match means every nucleotide in q has a probability of 1 in the corresponding position of G. If we keep this definition, many sequences of the genome will be ignored. Naturally, we can extend this notion to a "probabilistic match". To find out the closeness of a query to a subsequence of G, we can measure its joint probability at the query. Note that it can still be the most likely subsequence if its joint probability is less than 0.5; that's because there are 4^{11} sequences to choose from, not 2. However, we only have their respective probabilities, not joint or conditional probabilities. We may search online for LD for this species, which can be used to calculate the conditional probabilities. However, that is not enough, because we want the joint probability of 11 positions, not 2. As a result, we can only assume that genome nucleotides are independent and compute their joint probability as $\Pi_{i=j}^{j+11} P(G[i] = q[i])$.

Note that it may not be a very accurate estimation of joint probability when there is a strong correlation. Consider the extreme case that

$$P(G[i], A) = 0.5, P(G[i], C) = 0.5, P(G[i+1], A) = 0.5, P(G[i+1], C) = 0.5$$

but actually, we observe that it is either AA or CC in the $[i : i+2]$ of the population with equal probability, which would produce the above result. In other words, there is a perfect linear correlation. We should have

$$P(G[i : i+2], AA) = P(G[i : i+2], CC) = 0.5$$
$$P(G[i : i+2], AC) = P(G[i : i+2], CA) = 0$$

However, from our calculation, we would have

$$P(G[i : i+2], AA) = P(G[i : i+2], CC)$$
$$= P(G[i : i+2], AC) = P(G[i : i+2], CA) = 0.25$$

which differs from the truth considerably. However, since we do not have all the population genotypes and we only have one statistic on that, we cannot approximate the joint probability accurately.

Now every sequence at every sliding window has a probability. We need to only record the indices of those with a significant probability, so we keep ones with a joint probability higher than a threshold $\tau = 0.6^{11}$. A higher τ keeps some actually possible sequences out, and a lower τ takes more time, RAM, and disk space for potentially unrelated sequences. In practice, about 80% of the G positions have a major nucleotide probability of more than 0.95. Hence it on average allows one minor selection with probability $0.6^{11}/0.95^{10} = 0.006$, or have two minor selections with probability $\sqrt{0.6^{11}/0.95^{9}} = 0.076$, which is pretty inclusive and lenient in "minor-selections". As a result, this is a good threshold in practice. In the end, we have a list of positions in the G for every w-mer, which will be used in the next step.

3.4 Ungapped Extension

The method for obtaining high-scoring pairs (HSPs) starts by initializing an empty dictionary to store the HSPs. The process then use sliding window on query sequence and compare the window with genome seeds. The positions of the genome seeds provide the initial locations from which the seeds can be extended.

To compute the score at each position in sequences, we propose scoring function: score $= \log_4 p + 1$ where p is the probability of the nucleotide at this position. The base 4 is because of 4 types of nucleotides: {A, T, C and G}. This scoring function is a key feature, as it provides a quantitative measure of the match between the query and the genome. The use of probabilities in the scoring function allows for a reasonable evaluation of alignment, taking into account the probabilistic property of the genome sequence. We use the proposed scoring function to find seeds.

The right and left extensions are performed by iterating over the query sequence from the end of the seed to the end of genome or query sequences and from the start of the seed to the beginning of the genome or query sequences, respectively. In right and left extensions, we continue to use our proposed $score = \log_4 p + 1$. The extension stops when the score drops more than a predefined threshold from the maximum score achieved during the extension.

After both extensions, the method checks if the HSP has score higher than a predifined minimum HSP score threshold. If it is, then the HSP is added to the dictionary. This step ensures that only HSPs with a certain minimum score are considered, which helps to filter out insignificant HSPs, reducing the computational load of our algorithm.

In addition to the logarithmic scoring scheme that we proposed, another scoring function, the linear scoring function, was also taken for comparison in our study. In linear scoring function, the score is defined as $score = p - 0.25$. This means that for an unlikely event, the score, as a penalty, is approximately -0.25, while for a likely event, the score, as a reward, is approximately 0.75. This scoring function provides a straightforward method to quantify the match or mismatch based on the probability.

3.5 Gapped Extension

As mentioned in the introduction and literature review, given an appropriate scoring function, local alignment algorithm Smith-Waterman is guaranteed optimal. To this point, we have generated HSP's which allow us to target matching regions of our query and database which are likely to produce optimal alignments. The next step in our procedure will be to evaluate the optimal local alignment of these sections to produce close-to-optimal local alignments to the global problem. The prior un-gapped extension step produces many quality candidates for full sequence alignment, however, among these are many short matches that may have arisen just from chance. To determine which HSP's merit further attention, we must first get an expectation of what score is expected to represent a high-quality match. To achieve this, we use the Gumbel distribution to estimate the likelihood of seeing a certain score S by chance, and only retaining those matches which are very likely to be good matches. For a further description of how the Gumbel distribution is created, see section 3.6. Based on empirical results, a cutoff probability p=1e-10 was selected, which corresponds to a cutoff score of 20 for a query length of 500 (note that the cutoff score is a a function of the query length).

The original implementation of BLAST Altschul et al. (1990) states that candidate HSP's ought to be extended using a local alignment until a lower threshold is met. To achieve this, the entire query string corresponding to each HSP is aligned to the database sequence, and the corresponding bases in the database, in addition 50 bases prior to and following the query sequence are extracted and the two sequences are aligned using the Smith-Waterman local alignment algorithm. The gap penalty used is -2 and the 'proposed' scoring scheme from section 4.2.1 is used to evaluate base match/mismatch. Once the scoring is complete, the significance of each match is evaluated using the scoring scheme described in the next section.

3.6 Evaluating Match Significance

The BLAST algorithm described up to this point has provided meaningful candidate alignments from our query to a database sequence. As the size of the query shrinks and the size of the database grows, however, it becomes more likely that high-scoring alignments occur by chance rather than indicate a phylogenetic relationship. In order to evaluate this significance, we use the Gumbel extreme value distribution shown below.

$$p\left(S \geq x\right) = 1 - \exp\left(-e^{-\lambda(x-\mu)}\right), where$$

where

$$\mu = \frac{\log\left(Km'n'\right)}{\lambda}$$

The parameters K and μ are estimated by fitting the local alignment scores of a valid query with shuffled versions of the target genome, while m' and n' are the lengths

of the query and database sequences. The value x is a given ungapped or gapped alignment score and p is the probability density. Given a score and length of query and database being matched, we are able to assess the likelihood that this is a true match resulting from phylogeny and not chance. For the methods used to fit the distribution, see result section 4.4.

4 Result

4.1 Indexing the Genome Result

By Section 3.3, we get a json of "{one-w-mer: [idx's] for all w-mers}". With the default threshold, it has 13.8 million indices, which averages for $13.8m/604466 = 22.8$ subsequences per sliding window. In traditional BLAST, there is one sequence per sliding window, and now there are about 20. However, this step is a preprocess that is run only once, so this increase in time and memory is not a big problem, especially given that we have a lot more information in the input genome.

4.2 Ungapped Extension Result

For the Ungapped Extension, we tried to test on our proposed logarithm scoring function, and the linear one. The running time is the same as BLAST, which is linear in length. For each scoring function, we tested on 3 different query length: 50, 200, 500. For each selected query length, we design the number of random inserts and deletes: no more than 20% of query sequence length. This means that given query sequence length of 50, 200, 500, the number of insertion and deletion will be no more than 20, 40, 100 respectively. The delta of the ungapped alignment we chose for our experiments in this step is: 30. We also set a cutoff score of 5 across all experiments to filter out HSPs that have too low scores. Figure 1 shows an example of a high scoring pair extension for query length 1000.

4.2.1 The proposed scoring function

As mentioned in 3.4, our proposed scoring function is: $\log_4 p + 1$. The experiment results of ungapped alignment using this method is shown in table 1 below.

Query Length	50	200	500
HSP count	7	11	46
Average HSP Length	11.71	14.63	11.65

Table 1: HSPs results using proposed scoring function

4.2.2 linear scoring function

A different scoring function that we chose to compare is linear, $p - 0.25$. The experiment results of ungapped alignment using this is shown in table 2 below.

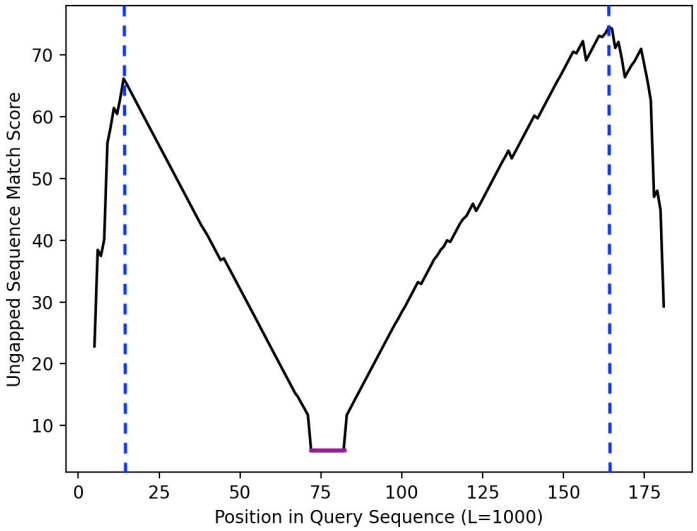

Figure 1: Example of ungapped alignment for query length 1000 with proposed scoring scheme. Blue dashed line is the end of the HSP, purple line is the alignment seed.

Query Length	50	200	500
total HSP count	144	912	1104
Average HSP Length	32.52	131.16	314.97

Table 2: HSPs results using linear scoring function

4.3 Gapped Extension Result

The threshold of p=1e-10 is used to select only HSPs which are likely to be real matches. For a query length of 500, this reduced the number of candidate pairings from 2500 to 25. The smith-waterman algorithm is then used to align the best candidate high scoring pairs. Figure 2 shows an example of an optimal local alignment created from a high-scoring pair. The traceback starts at the maximum value and stops once a lower threshold is met. In the figure, this value is 5, however different values were explored to maximize the quality of matches. Two scoring functions were evaluated as a way to increase or decrease the number of high scoring matches in section 4.2. Comparison of the number and length of HSP's produced from each scoring function in the prior section helped to contrast the pros and cons of each scoring function in the context of HSP generation. The final scores and local optimal alignment length evaluated using each scoring function will not help to gain insight into the final quality of the matches, however, because the scores are subjective to the scoring function being used. In order to properly evaluate the quality, we may take a cue from the original BLAST, and use the Gumbel distribution to model the quality of matches from each scoring function. The running time of this step is the same as BLAST, which is about $O(\text{len}(\text{query}))$

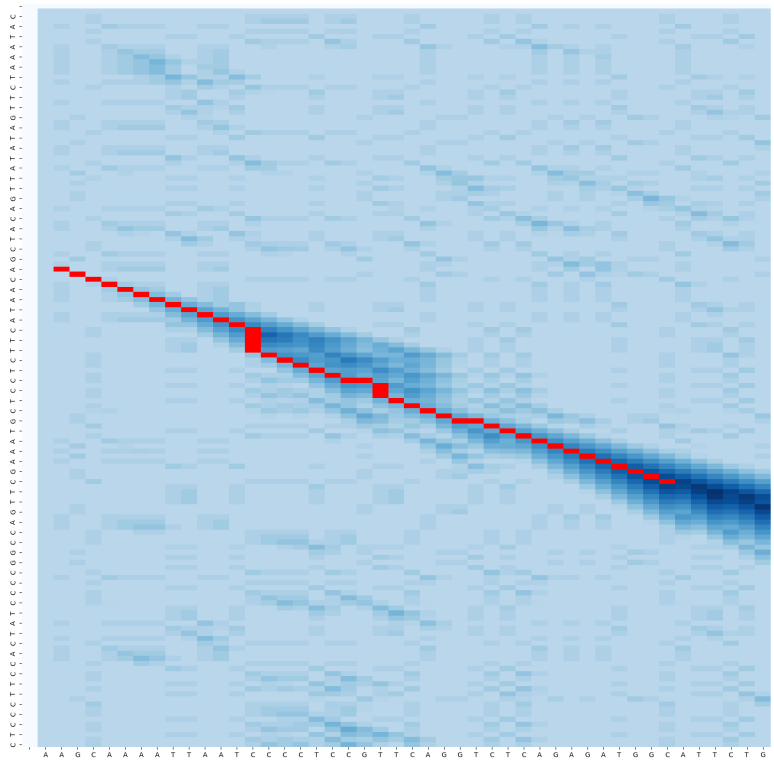

Figure 2: Example of smith-waterman alignment between query on x-axis and database on y-axis. Blue shade indicates score, red indicates traceback.

4.4 Evaluating Local Alignments

The gumbel distribution allows us to estimate the significance of an alignment score relative to the scoring scheme. A separate distribution is fit for each scoring scheme using Monte Carlo simulation. In each monte carlo simulation, a valid query string Q and shuffled database G are fit and the score is reported. λ and μ are fit over 100 simulations for a different query and database length. The query lengths 50, 100, 250, 500 and 1000 are fit a distribution (varying m+n), and then K and λ are estimated from the following linear regression:

$$e^{\mu * \lambda} = Km'n'$$

A Gumbel distribution of expected scores using the proposed scoring function is shown in Figure 3. The parameters for each scoring function are shown below:

Scoring	K	λ
Proposed Method	5.24e-6	0.786
Linear Method	6.39e-4	1.54

Table 3: Parameters for scoring functions

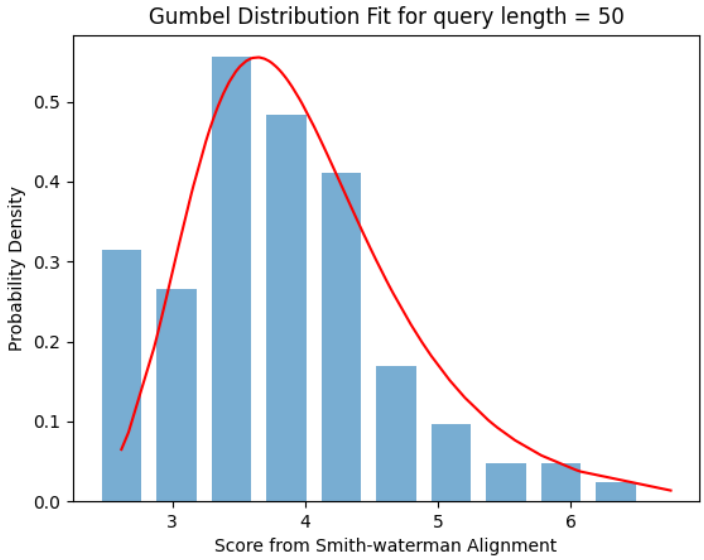

Figure 3: Distribution of scores for original scoring scheme. Bars show the observed values over 100 simulations and the red line the fit distribution.

4.5 Comparing Scoring

Now that we have a way to evaluate the quality of our score, we can proceed to comparing our two scoring metrics. First, the proposed scoring function $\log_4 p + 1$. Table 4 shows the BLAST results using a threshold of 5 for local alignment.

Query Length	50	200	500
Num Matches	2	8	13
Avg Length	50	162	403
Avg Score	44.4	65.1	109.0
Avg p-value	2.45e-11	6.66e-13	1.22e-25

Table 4: Results for BLAST with proposed scoring

The second scoring function evaluated is p-0.25. The same threshold for local alignment of 5 is used. The run time for this scoring scheme ranged from 50-100 times longer than the prior scoring scheme and the p-value cutoff had to be reduced from 1e-10 to 1e-2 in order to produce results. The results from the linear scoring scheme can be seen in table 5.

5 Discussion and future work

In Ungapped alignment, we tested on 2 scoring function: our proposed logarithm method, and the linear method for comparison. We found that when set with same parameters, our proposed method has better HSPs output. This is because

Query Length	50	200	500
Num Matches	58	753	1143
Avg Length	39	162	471
Avg Score	17.6	30.3	50.6
Avg p-value	1.71e-5	4.22e-8	1.67e-11

Table 5: Results for BLAST with linear scoring

the amount of HSPs and average HSP length are at reasonable value, but, for linear method, there are too many HSPs outputs, and the computation load for the next gapped alignment step would be high. To evaluate each scoring function, an extreme value distribution is fit to scores from random sequence alignments. Each scoring function is then used to run a complete BLAST workflow, evaluating the full number of candidates produced using each method. The p-values obtained using the proposed scoring scheme are far better than those obtained using the linear scoring - in most cases, more than 5 orders of magnitude. The run time of linear scoring is much longer, and produces far too many matches to be useful. One problem that was observed with this version of BLAST is that very similar sequences with minor adjustments all show up as matches. It would be worthwhile to investigate these matches and see if there are multiple sequences with the same length at the same location in the genome, and try to reduce redundancy.

For future work, We shall improve our study by designing a dataset of query sequences and subsequently testing our algorithm on this dataset. Meanwhile, we shall further explore more types of scoring functions for our algorithm.

References

Stephen F. Altschul, Warren Gish, Webb Miller, Eugene W. Myers, and David J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, 1990. ISSN 0022-2836. doi: https://doi.org/10.1016/S0022-2836(05)80360-2. URL https://www.sciencedirect.com/science/article/pii/S0022283605803602.

Jason D Buenrostro, Beijing Wu, Howard Y Chang, and William J Greenleaf. ATAC-seq: A method for assaying chromatin accessibility genome-wide. *Curr. Protoc. Mol. Biol.*, 109(1):21.29.1–21.29.9, January 2015. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4374986/.

Jiannan Chao, Furong Tang, and Lei Xu. Developments in algorithms for sequence alignment: A review. *Biomolecules*, 12(4):546, April 2022. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9024764/.

James M Heather and Benjamin Chain. The sequence of sequencers: The history of sequencing DNA. *Genomics*, 107(1):1–8, January 2016. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4727787/.

Dragomirka Jovic, Xue Liang, Hua Zeng, Lin Lin, Fengping Xu, and Yonglun Luo. Single-cell RNA sequencing technologies and applications: A brief overview. *Clin. Transl. Med.*, 12(3):e694, March 2022. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC8964935/.

Matthieu Muffato, Alexandra Louis, Nga Thi Thuy Nguyen, Joseph Lucas, Camille Berthelot, and Hugues Roest Crollius. Reconstruction of hundreds of reference ancestral genomes across the eukaryotic kingdom. *Nature Ecology & Evolution*, 7 (3):355–366, January 2023. ISSN 2397-334X. doi: 10.1038/s41559-022-01956-z. URL http://dx.doi.org/10.1038/s41559-022-01956-z.

Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970. ISSN 0022-2836. doi: https://doi.org/10.1016/0022-2836(70)90057-4. URL https://www.sciencedirect.com/science/article/pii/0022283670900574.

Marcello Silva e Santos. The PhOCoe model–ergonomic pattern mapping in participatory design processes. *Work*, 41 Suppl 1:2643–2650, 2012. URL https://pubmed.ncbi.nlm.nih.gov/2231712/.

T.F. Smith and M.S. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981. ISSN 0022-2836. doi: https://doi.org/10.1016/0022-2836(81)90087-5. URL https://www.sciencedirect.com/science/article/pii/0022283681900875.

Zhong Wang, Mark Gerstein, and Michael Snyder. RNA-Seq: a revolutionary tool for transcriptomics. *Nat. Rev. Genet.*, 10(1):57–63, January 2009. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2949280/.