ProbaBLAST: A modified BLAST algorithm for querying nucleotide sequences on probabilistic target databases

Alexander Bertrand
McGill University

December 10, 2021

Abstract

Basic Local Alignment Search Tool (BLAST) is an extremely prevalent family of algorithms for aligning and comparing biological sequences. BLAST uses a heuristic approach which allows it to run substantially faster than deterministic algorithms while maintaining comparable accuracy. However, BLAST programs require the database sequence to be deterministic rather than probabilistic. This presents a disconnect between common types of sequence databases and the tools needed to analyze them; a popular strategy for predicting ancestral genomes is to compare sequences of related extant species and use phylogenetic methods (e.g., maximum likelihood) to reconstruct ancestral sequences. Due to their probabilistic nature, these methods come with inherent uncertainty in their predictions. Current BLAST algorithms can't utilize these uncertainties, meaning potentially valuable information for finding biologically likely alignments of query sequences is thrown out. Here, I propose and implement a modification of the BLAST algorithm that allows alignments between non-probabilistic queries and a probabilistic nucleotide database using an efficient confidence-based score coefficient. The algorithm achieved upwards of 95% alignment accuracy under biologically realistic conditions, and performance remained respectable even under highly inflated mutation rates. Its speed is acceptable for aligning short to medium length queries (a few thousand base pairs), even on consumer-grade hardware.

Introduction

In its most basic form, the problem of sequence alignment seeks to find similar regions between two or more biological sequences. Good alignments can suggest that genes or proteins may serve a similar function or indicate a shared ancestry between species. Deterministic algorithms for solving sequence alignment have been known for decades, but early methods have time complexities that grow rapidly with the length of the sequences being aligned (Needleman and Wunsch, 1970). Modern advances, such as relatively fast and cheap genome sequencing, are causing exponential growth in the amount of available biological data that makes these slow approaches unfeasible (Li and Chen, 2014).

Basic Local Alignment Search Tool (BLAST) was first introduced in 1990, quickly becoming one of the most widely-cited scientific papers ever (Altschul et al., 1990; Van Noorden et al., 2014). Its heuristic alignment strategy allows it to achieve speeds orders of magnitude faster than its deterministic cousins with only minor compromises in accuracy. This efficiency has opened many opportunities for researchers in nearly all biological disciplines, particularly when working with very large sequence datasets. Initially designed to compare nucleotide queries to nucleotide sequence libraries or protein queries to protein sequence libraries, BLAST variants have been developed to automatically translate between nucleic acid and protein sequences, consider multiple possible reading frames, and more.

BLAST's defining feature is a preconstructed database (or "library") that is generated before any query is submitted. This database contains all of the subsequences ("words") of a set length w that appear in the database sequence(s), and a list of the locations at which they appear. When a query sequence is given, all of its w-length subsequences can be looked up in this database to find locally-identical "hits" extremely quickly. Next, the program performs an ungapped extension phase in which each hit is extended in either direction to ensure that the similarity is statistically significant. The quality of the alignment is scored using a substitution matrix, which quantifies the similarity of the sequences; matches in the sequence increase the score while mismatches are penalized. The best high-scoring segment pairs (HSPs) found in this phase can either be returned as-is, or further extended by a gapped alignment algorithm to increase the precision of the output.

A common type of database that biologists may need to query are inferred genomes of now-extinct ancestral species. These genomes can be estimated by comparing the DNA of related species that exist today through methods such as maximum parsimony or maximum likelihood (Thornton, 2004; Randall et al., 2016). However, the resulting genomes are just that — estimates. Each of the predicted nucleotides will have some degree of confidence associated with it.

Unfortunately, no algorithms in the BLAST family currently support querying probabilistic genomes. Therefore, attempts to use BLAST for queries on a probabilistic genome necessarily ignore any information provided by the genome's uncertainties. ProbaBLAST aims to remedy that.

Methodology

Defining "Optimal Alignment"

The first problem that must be tackled is to establish a definition for what the "best" alignment of a query sequence to a probabilistic database is. Intuitively, a good match should be one that has both a high rate of matches between the query and the database

(the same as normal BLAST) and also a high degree of certainty that the matches are "real" — not caused by prediction errors in database. Furthermore, if a mismatch occurs in a location with low confidence, it is conceivable that it was caused by a mispredicted nucleotide in the database instead of a mutation in the actual DNA. With this in mind, we want the scoring function to produce scores such that:

1. Matches in locations of the database with *high* confidence should be rewarded more than matches in locations with *low* confidence.

2. Mismatches in locations of the database with *high* confidence should be penalized more harshly than mismatches in locations with *low* confidence.

The most straightforward way to quantify this relationship is by using a coefficient to modify the score provided by the substitution matrix. For simplicity, we will only consider the most likely residue at position i, with its confidence denoted c_i; every other nucleotide at that location is assumed to be equally likely.

In the case where we are perfectly confident for a given nucleotide ($c_i = 1$), the original score should be weighted fully. Any difference between the query sequence and the ancestral sequence must have arisen from a biological mutation, rather than from a chance that the prediction for the nucleotide is incorrect. In the other extreme case, when our prediction for the identity of the nucleotide is completely random ($c_i = 0.25$), the match/mismatch score should have no effect on the alignment's score; it does not make sense to reward or penalize one nucleotide in the query sequence differently from another if there is no evidence that any nucleotide from the ancestral sequence is more likely than another. Based on these two criteria, the function to generate the score coefficient, $f(c_i)$, should satisfy two properties:

$$\begin{cases} f(c_i) = 0 & \text{when } c_i = 0.25 \\ f(c_i) = 1 & \text{when } c_i = 1 \end{cases}$$

The simplest function with these properties is the linear function:

$$f(c_i) = c_i - \tfrac{1-c_i}{3} = \tfrac{4}{3}c_i - \tfrac{1}{3}$$

This is the coefficient function that was used during implementation and testing of the algorithm.[1] The alignment that maximizes the weighted score is defined to be the optimal alignment.

ProbaBLAST Algorithm

ProbaBLAST closely resembles traditional BLAST. The word library is built identically to normal BLAST, except that only words that have confidences (the product of the confidence at each letter in the word) above a user-definable threshold are added. This provides a coarse way to limit the words that must be searched during runtime, and reduces the disk space used to store the library.

When a query sequence is given, every word it contains is looked up in the library. The nucleotides at each instance of that word in the database are scored using the weighted scoring system described above. Each of these hits then undergoes ungapped extension in both directions, which continues until the score in that direction falls a pre-defined amount below the maximum reached (indicating that the alignment is no longer improving); the maximum scoring alignments of the left and right extension are added to the initial word's alignment.

If the overall alignment has a score above a certain threshold, it is considered an HSP and

[1] More functions satisfying these conditions exist, some of which may have useful properties (see Discussion)

the algorithm performs deterministic gapped extension between it and a local portion of the database sequence. This was done using the Needleman-Wunsch algorithm, slightly modified to incorporate the weighted scoring system. As a time-saving measure, gapped extension is only done on HSPs that aren't fully contained by the gapped extension found for a previous HSP. This lowers the chance of finding the highest-scoring alignment slightly, but very significantly reduces computation time. Finally, a customizable number of the top-scoring alignments and their scores are returned to the user.

Testing Procedure

To test the algorithm, a database with word length $w = 7$ and word confidence threshold 0.8 was generated from a portion of the predicted chromosome 22 of an ancestral Boreoeutherian (a sub-type of placental mammals, which includes humans) (Kuhn et al., 2018). The genome had a total length of 604,466 residues.

Subsequences with lengths between 50 and 500 residues were randomly selected from the database. These sequences were then mutated with both point mutations and insertions/deletions (indels) to create the query sequence. Unless otherwise specified, the probability of a point mutation at a given nucleotide was $1 - c_i$ (i.e., the probability of the database's prediction being incorrect), and the average rate of indels was 4% (that is, a 100 nucleotide sequence would have 4 residues inserted and/or deleted on average); this is significantly higher than estimated biological indel rates (Boschiero et al., 2015), but was needed to have them appear at an appreciable frequency (and to challenge the algorithm). The length of each indel was generated from a Poisson distribution with mean length = 3, since evidence suggests that most indels are only one to five base pairs long (Boschiero et al., 2015; Cartwright, 2009). Insertions and deletions were treated as equally likely.

Scoring was done using the default values for NCBI nucleotide BLAST: a match score of +1, a mismatch score of -2, and a linear gap penalty of -2 (Madden, 2003).

The performance of the algorithm was evaluated with a metric called "coverage", which measures the proportion of overlap between the true sequence (the subsequence of the database from which the query was generated) and probaBLAST's predicted alignment. A perfect alignment will have coverage = 1, while an alignment to a completely different location in the database will have coverage = 0 (even if that alignment's score is high).

Results

Accuracy

When run on 100 randomized sequences with default parameters, 97% of queries were matched at least partially to their source (Fig. 1). The average coverage over all queries was 95.6%, and 55/100 of all queries were perfectly matched (with 100% coverage).

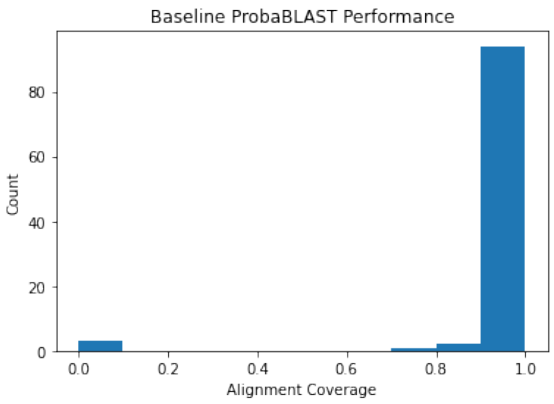

Figure 1: Performance of probaBLAST on $n = 100$ random sequences using default algorithm parameters.

To investigate how the algorithm's performance would change as the query sequence became more distantly related to the database, the sequences were randomized with increasing mutation rates (Fig. 2). The new mutation probabilities were calculated as $(1 - c_i) + p_{mut}$, where p_{mut} is the added mutation probability. Below 0.2, the quality of alignments remained relatively stable. At +0.3, 58% of queries were partially matched to their source, but only 10% had very good alignments (above 95%). With a p_{mut} of 0.5, less than half of nucleotides from the source sequence remained unchanged. Predictably, such abysmal similarity to the source sequence meant that finding a meaningful alignment was essentially impossible.

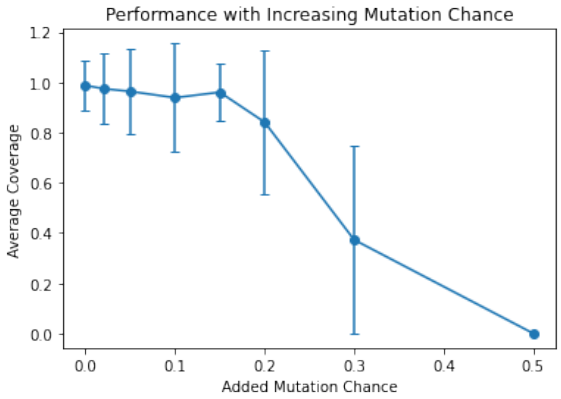

Figure 2: Performance with increasing substitution mutation rates. Each data point is based on $n = 100$ randomized subsequences. Error bars represent one standard deviation around the mean coverage. No indels were added during randomization.

Next, I looked at how the rate of indels affected the alignment coverage (Fig. 3). Due to increasing noise from insertions and lost information from deletions, the performance suffered as indels increased. Luckily, for any biologically-relevant indel rate (recall that the default value of 0.04 is already higher than most estimates of indel occurrence), the accuracy only decreased slightly as the rate increased.

Together, figures 2 and 3 both show stable performance up to around 20% dissimilarity (0.2 added mutation chance or 0.2 indel rate), followed by rapid declines. It is likely that changing the parameters would allow for better accuracy, such as by reducing the gap cost when increasing the rate of indels.

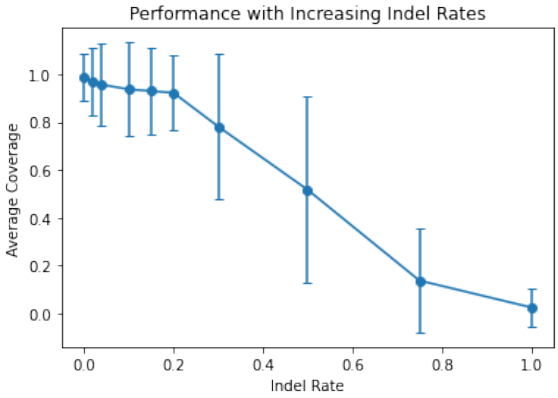

Figure 3: Performance with increasing indel rates. "Indel rate" measures the average proportion of nucleotides involved in indels compared to the query length. The mean indel length was 3. Each data point is based on $n = 100$ randomized subsequences. Error bars represent one standard deviation around the mean coverage.

While very short indels are the most common type of indel, large indels (which can be hundreds or thousand of base pairs long) can also occur many times within genomes (Chen and Zhang, 2015). It is therefore beneficial for alignment algorithms to be robust to long regions of dissimilarity. To gauge probaBLAST's performance on queries with long indels, the average indel length was increased (Fig. 4). Note that to ensure a similar number of indels per sequence, the indel rate (0.04, by default) had to be adjusted proportionally to the indel length. This meant that the indel rate grew similar to what was

shown in Fig. 3 (for example, the indel rate when the average indel length was 30 was $0.04 \times 30 = 1.2$).

Interestingly, the algorithm performed better with longer indels than shorter ones at similar overall indel rates, indicating that the algorithm can more easily align a sequence with a few large indels than a sequence with many small indels. The reason for this may be explained by BLAST's word library strategy; many short indels would be likely to interrupt query words used in the initial library search, while few long indels would leave wider stretches of the query intact.

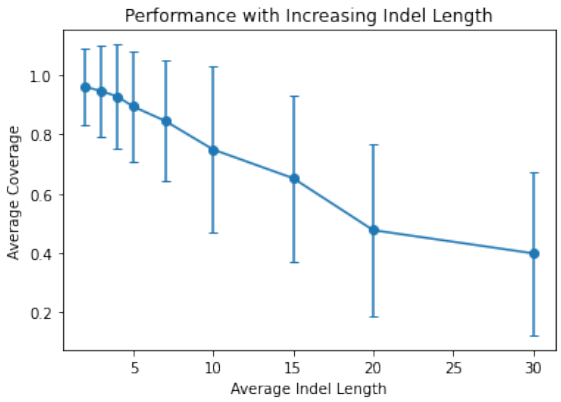

Figure 4: Performance with increasing average indel length. Each data point is based on $n = 100$ randomized subsequences, and error bars represent one standard deviation from the mean coverage.

Speed

The only significant difference between the probaBLAST and canonical BLAST algorithms is probaBLAST's calculation for the score coefficient, which can be computed in constant time. Therefore, the asymptotic time complexities of the two are the same (when not counting various assumptions/optimizations that are used in some implementations of BLAST). However, the speed of a BLAST query is highly dependent on the input sequences and parameters used in the algorithm, making a useful Big-O running time analysis difficult.

That being said, practical estimates of the speed can be made by aligning test queries. On an Intel Xeon CPU running at a relatively modest 2.20 GHz, a set of 100 50 to 500 base pair sequences could all be aligned in 59.7 seconds, and a single 1000 base pair query took under four seconds. Computation times for very long sequences began to blow up, with a 5000 base pair query taking two minutes and ten seconds. This was dominated by the gapped extension (Needleman-Wunsch) phase of the algorithm, whose running time is quadratic with sequence length. Strategies exist to address this issue, but they were not implemented in the initial version of probaBLAST (see Discussion).

Discussion

ProbaBLAST provides a strong proof of concept for integrating genome uncertainty into the BLAST framework. I believe the score coefficient method for incorporating genome confidence information into alignments is a very efficient and logical approach to the problem. That being said, the linear function used during testing may not be the best possible one. Taking the linear function to any positive power creates a polynomial that satisfies the same criteria, but with a different shape between zero and one (Fig. 5). These shapes would allow fine-tuning the response of the algorithm to the confidence. For example, using the square root of the linear coefficient function would require better matching in low-confidence areas to achieve a good score. Another possible function is $1-\log_4 c_i$, whose form is similar to the concept of Shannon entropy used in information theory (Shannon, 1948). It could be extremely interesting to link probaBLAST's coefficient

function to a well-studied mathematical concept.

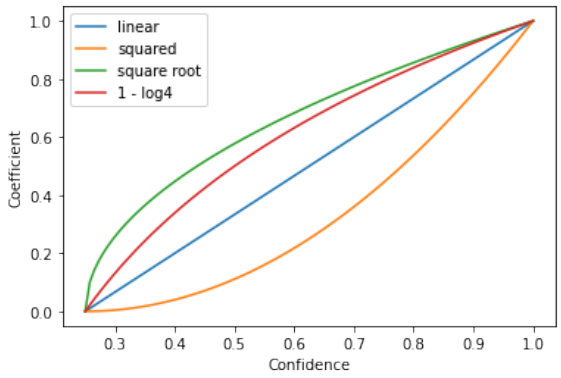

Figure 5: Possible alternative coefficient-generating functions. All graphs are plotted on the range [0.25, 1].

In general, the values for the many parameters of the algorithm have not been optimized. Of course, good values for most of the parameters (e.g., the substitution matrix and gap penalty) depend highly on the application and desired specificity, so an "optimal" parameter may not be meaningful. That being said, choices for some of them (such as the threshold score needed to continue from an HSP to gapped extension) were made arbitrarily, which most likely leaves performance on the table.

A significant weakness in the current implementation is the slow computation times on very long sequences during the gapped extension phase. A way around this is to intelligently consider the HSPs found before doing any gapped extensions. If two or more HSPs are close to one another, those areas of the alignment can be considered "fixed" and gapped extension only needs to be done on the regions between them. A good alignment will have most of the alignment be part of at least one HSP, so this would drastically reduce the time needed for gapped extension.

ProbaBLAST currently only supports linear gap penalties. Research has found that affine gap penalties generally provide better results (Liu et al., 2009). Affine gap penalties can be implemented without increasing the asymptotic time complexity of the alignment algorithm, meaning the efficiency of the algorithm would remain high (Altschul and Erickson, 1986).

The idea behind probaBLAST can be extended beyond just queries on probabilistic genomes. There is no reason that the concepts described here don't apply to other types of sequences, like proteins. However, probabilistic protein sequences are far less common than probabilistic nucleotide sequences. It would also be relatively simple to apply the same methodology to query sequences, which would allow probabilistic queries on both deterministic and probabilistic databases.

References

S. F. Altschul and B. W. Erickson. Optimal sequence alignment using affine gap costs. *Bulletin of Mathematical Biology*, 48(5):603–616, Sept. 1986. ISSN 1522-9602. doi: 10.1007/BF02462326. URL `https://doi.org/10.1007/BF02462326`.

S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman. Basic local alignment search tool. *Journal of Molecular Biology*, 215(3):403–410, Oct. 1990. ISSN 0022-2836. doi: 10.1016/S0022-2836(05)80360-2.

C. Boschiero, A. A. Gheyas, H. K. Ralph, L. Eory, B. Paton, R. Kuo, J. Fulton, R. Preisinger, P. Kaiser, and D. W. Burt. Detection and characterization of small insertion and deletion

genetic variants in modern layer chicken genomes. *BMC Genomics*, 16:562, July 2015. ISSN 1471-2164. doi: 10.1186/s12864-015-1711-1. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4563830/.

R. A. Cartwright. Problems and solutions for estimating indel rates and length distributions. *Molecular Biology and Evolution*, 26(2):473–480, Feb. 2009. ISSN 0737-4038. doi: 10.1093/molbev/msn275. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2734402/.

W. Chen and L. Zhang. The pattern of DNA cleavage intensity around indels. *Scientific Reports*, 5(1):8333, Feb. 2015. ISSN 2045-2322. doi: 10.1038/srep08333. URL https://www.nature.com/articles/srep08333.

H. Kuhn, L. Humeniuk, N. Kozlov, S. Roigas, S. Adel, and D. Heydeck. The evolutionary hypothesis of reaction specificity of mammalian ALOX15 orthologs. *Progress in Lipid Research*, 72:55–74, Oct. 2018. ISSN 0163-7827. doi: 10.1016/j.plipres.2018.09.002. URL https://www.sciencedirect.com/science/article/pii/S0163782718300018.

Y. Li and L. Chen. Big biological data: challenges and opportunities. *Genomics, Proteomics & Bioinformatics*, 12(5):187–189, Oct. 2014. ISSN 1672-0229. doi: 10.1016/j.gpb.2014.10.001. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4411415/.

K. Liu, S. Nelesen, S. Raghavan, C. R. Linder, and T. Warnow. Barking up the wrong tree-length: the impact of gap penalty on alignment and tree accuracy. *IEEE/ACM transactions on computational biology and bioinformatics*, 6(1):7–21, Mar. 2009. ISSN 1557-9964. doi: 10.1109/TCBB.2008.63.

T. Madden. *The blast sequence analysis tool*. National Center for Biotechnology Information (US), Aug. 2003. URL https://www.ncbi.nlm.nih.gov/books/NBK21097/.

S. B. Needleman and C. 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, Mar. 1970. ISSN 0022-2836. doi: 10.1016/0022-2836(70)90057-4. URL https://www.sciencedirect.com/science/article/pii/0022283670900574.

R. N. Randall, C. E. Radford, K. A. Roof, D. K. Natarajan, and E. A. Gaucher. An experimental phylogeny to benchmark ancestral sequence reconstruction. *Nature Communications*, 7(1):12847, Sept. 2016. ISSN 2041-1723. doi: 10.1038/ncomms12847. URL https://www.nature.com/articles/ncomms12847.

C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27(3):379–423, July 1948. ISSN 00058580. doi: 10.1002/j.1538-7305.1948.tb01338.x. URL https://ieeexplore.ieee.org/document/6773024.

J. W. Thornton. Resurrecting ancient genes: experimental analysis of extinct molecules. *Nature Reviews Genetics*, 5(5):366–375, May 2004. ISSN 1471-0064. doi: 10.1038/nrg1324. URL https://www.nature.com/articles/nrg1324.

R. Van Noorden, B. Maher, and R. Nuzzo. The top 100 papers. *Nature News*, 514 (7524):550, Oct. 2014. doi: 10.1038/514550a. URL http://www.nature.com/news/the-top-100-papers-1.16224.