CS 466 Project

Andrew Yang
ayang14@illinois.edu

Jiwon Kwak
jkwak12@illinois.edu

Jasmine Kwon
jhkwon3@illinois.edu

May 4, 2018

Abstract

Aligning two sequences of nucleotides or proteins is a common problem in bioinformatics. Since 1970, there have been several different algorithms that have been developed that use different approaches and have slightly different objectives. The four algorithms we will be examining are the Needleman-Wunsch algorithm for global alignment, the Smith-Waterman algorithm for local alignment, semi-global (glocal) alignment, and Hirschberg's algorithm. We derive our datasets—one that simulates mass mutation of nucleotides and one that simulates mass deletion of nucleotides—from the one provided by M. Burset and R. Guigó of the Centre for Genomic Regulation [1].

Introduction

Sequence alignment is at the basis of analyzing the evolutionary and functional relationships between molecular sequences. An alignment of two sequences is an arrangement of both sequences, one written below the other. There are matches, mismatches, insertions, deletions, and gaps in alignment. Depending on the context, matches/mismatches can have varying meanings. They can indicate evolutionary, structural, or other correlation.

This project was aimed to analyze different sequence alignment algorithms based on quality of alignment and speed. We derive two different datasets from the original dataset provided by the Centre for Genomic Regulation. For the first one, each nucleotide has a 5% chance of being mutated and a 5% chance of being deleted. For the second one, each nucleotide has a 2.5% chance of being mutated and a 47.5% of being deleted.

The overall similarity between two biological sequences is studied by performing an alignment between them. The alignments were global alignment, semi-global alignment, local alignment, and Hirschberg algorithm. We tested the dataset with multiple algorithms to see how each fare with a certain DNA sequences.

Methods

Needleman-Wunsch algorithm

The Needleman-Wunsch algorithm is used to globally align nucleotide or protein sequences. It was developed by Saul Needleman and Christian Wunsch and published in 1970. It uses dynamic programming and runs in $O(mn)$ time for two sequences of length m and n, respectively. The original algorithm specified a match score, a mismatch score, a gap starting penalty, and a gap extension penalty. Later versions of the algorithm would use a similarity matrices, such as the BLOSUM62 matrix for amino acids. In this paper, we will explore the use of the original algorithm without the matrix.

Smith-Waterman algorithm

The Smith-Waterman algorithm is used to locally align nucleotide or protein sequences. It was first proposed by Professors Temple Smith and Michael Waterman in 1981. Like the Needleman-Wunsch algorithm, it uses dynamic programming. The original algorithm ran in $O(m^2n)$ time, but in 1982, Osamu Gotoh optimized the algorithm to run in $O(mn)$ time. Gotoh also modified Smith and Waterman's original algorithm to specify affine gap penalties.

Semi-global/glocal alignment

The semiglobal or glocal alignment is a hybrid of the global and local alignment methods as described above. As the name suggests, it combines the global and local methods such that the algorithm finds the best possible alignment which includes the start and the end of one of the sequences. There is no gap penalty for the beginning of a sequence or the end of a sequence. In the dynamic programming table, the recurrence is the same as the Needleman-Wunsch algorithm's recurrence, with a different base case and different backtrack start position. This accounts for the no-penalty start gap and end gap. Semiglobal alignment is generally more useful than other alignments when one sequence is significantly shorter than another or when the objective is to find the overlap between two sequences (e.g. the overlap of the end of one sequence and the start of another).

Hirschberg's algorithm

The Hirschberg algorithm is named after its inventor, Dan Hirschberg. It is a dynamic programming algorithm that finds the optimal alignment between two sequences. It is considered as a modified version of the Needleman-Wunsch algorithm which is more space efficient and applies the divide and conquer strategy. This algorithm can find the optimal alignment in O(mn) time, using O(min$\{m, n\}$) space.

Procedure

In order to be able to use our algorithms to align two unique but similar-enough sequences, we built two different datasets- one in which the sequences being aligned were similar in length, and the other in which one sequence was a lot longer than the other. Doing so ensured a more unbiased and more varied dataset.

To build the first dataset, we took the FASTA-formatted dataset (under the "DNA Sequences" link) provided by Burset and Guigó [1] and ran the first twenty sequences in the file through a program to mutate approximately 10% of the bases. We then ran the mutated sequences through the same program to mutate 10% of those bases. By mutating both the original sequences and then the mutated sequences, we could be sure that the sequence pairs would be different, but similar enough to align. The mutated sequences that originated from the same original sequence were put into twenty tuples and the resulting tuples were printed into a separate file so that we could run an algorithm by passing in the tuple values.

To build the latter dataset, we took the first dataset and manually deleted parts of one sequence in each tuple to ensure that one sequence would be significantly different in length than the other. The resulting tuples were put into another separate file so that we could pass the values into the algorithms separately.

The Needleman-Wunsch, Smith-Waterman, semi-global, and Hirschberg algorithms were run with the same twenty sequence tuples, as well as the same penalty and match scores to maintain consistency. In the relevant algorithms, base-pair match added two points, a mismatch cost one point, and gaps cost two points.

We chose to test these algorithms for efficiency by time and max alignment score, so instead of printing out the aligned sequences, we printed out the runtime of the algorithm programs and the resulting scores. The runtimes were found by subtracting the time the programs concluded from the time the programs started.

In order to do so, the four algorithms were called on each tuple so that each algorithm printed out the corresponding score and runtime until all twenty tuples were aligned.

Results

The four algorithms were run, each with the dataset of sequence tuples of similar lengths, which we called the mutated dataset, and the dataset of sequence tuples of significantly different lenghts, which we called the short/long dataset. The tuples in each dataset were run in the same order for each algorithm, and as such, the data in the first row of one table corresponds to the same tuples of sequences as any other table's first row data.

Mutated dataset		Short/long dataset	
Score	Time (sec)	Score	Time (sec)
7780	51.75674	2884	39.53572
10716	91.56036	3938	67.33856
15881	205.0983	5770	138.2112
2452	4.729127	893	3.764387
2415	4.657262	882	3.736915
616	0.2755852	210	0.183903
5318	22.50623	1953	18.59676
7817	51.0283	2927	32.70329
14306	166.1343	5261	113.9965
973	0.728934	372	0.511086
3183	8.221857	1160	5.682126
2477	4.979044	899	3.138434
6930	38.39597	2582	26.10153
11700	110.8228	4367	80.46727
2466	4.798284	910	3.294865
2560	5.581948	958	3.887726
9951	79.63418	3686	55.09359
1538	1.88921	576	1.263348
7836	48.84422	2863	37.39115
16971	234.3374	6161	191.0623

Table 1: Needleman-Wunsch/Global Alignment

Mutated dataset		Short/long dataset	
Scores	Time (sec)	Scores	Time (sec)
7782	59.78286	2884	59.80204
10716	110.5763	3940	96.79851
15881	247.9014	5770	188.6072
2452	5.832099	893	3.893948
2415	5.816085	884	3.869944
616	0.3605411	213	0.2299612
5318	27.60173	1956	18.09327
7817	58.95970	2948	40.86593
14306	209.7700	5267	156.5345
973	0.9541759	372	0.6540661
3183	10.31493	1165	8.199042
2478	5.940399	899	5.776792
6932	47.62160	2592	35.95764
11700	140.2220	4371	106.9021
2466	6.084021	911	4.601081
2560	6.458268	981	4.850940
9952	100.3695	3687	74.79712
1538	2.344161	581	1.663815
7836	60.69216	2864	43.68111
16971	299.1123	6165	204.9724

Table 2: Smith-Waterman/Local Alignment

Mutated dataset		Short/Long dataset	
Scores	Time (sec)	Scores	Time (sec)
7245	50.69379	1101	26.19088
9962	96.01232	1465	47.99920
14856	213.6798	2229	106.8956
2254	5.002428	390	2.493702
2257	5.183051	390	2.485531
565	0.287277	96	0.1469829
4955	25.16890	760	11.74212
7294	46.90851	1180	25.9655
13314	158.2497	2029	88.00435
898	0.6870651	176	0.3807278
2920	7.836137	470	4.31660
2287	4.537370	363	2.509853
6413	36.57707	978	20.252089
10835	106.9029	1768	58.14615
2281	4.590682	373	2.518566
2370	6.395129	375	2.712013
9226	80.73999	1516	41.86259
1413	1.789564	225	0.9774088
7254	47.14019	1194	25.88635
15630	226.1576	2456	122.5287

Table 3: Semi-global/glocal Alignment

Mutated dataset		Short/long dataset	
Scores	Time (sec)	Scores	Time (sec)
7549	98.09898	574	46.06929
10386	180.2541	788	88.01848
15418	400.0483	1071	200.6849
2387	9.171484	160	4.863019
2329	8.535144	155	4.230324
596	0.523904	21	0.2663548
5162	42.2537	377	20.15030
7582	88.650585	624	43.32446
13839	305.05972	973	156.9024
941	1.28972	87	0.6694280
3079	14.45835	194	7.446842
2390	8.310436	163	4.336179
6732	74.15377	542	35.06627
11342	200.064785	905	107.6991
2399	10.4218	157	4.517944
2487	9.6502	208	4.927732
9656	154.8836	745	82.30119
1488	3.89178895	118	3.536799
7569	89.5955	534	5.251584
16434	442.54319	1075	266.6642

Table 4: Hirschberg Alignment

Conclusion

The scores between local and global alignments are the most similar for both datasets, although global alignment slightly outperforms local alignment. Hirschberg's algorithm notably outperforms both local and global alignments in terms of score with an average difference in scores of 206.35 and 206.05 respectively, but is the worst performing algorithm in terms of time in both datasets. The Hirschberg's algorithm scores higher than semiglobal alignment in the mutated dataset with an average difference of 276.8 points in favor of Hirschberg, but in the long-short dataset the opposite is true, with the semiglobal alignment having an average of 503.15 points more than the Hirschberg algorithm.

With the mutated dataset, the global and semiglobal alignments are very similar in terms of time performance: the average of differences in time is about .57203 seconds, the lowest difference found between all algorithms run, on both datasets.

Out of all the algorithms run with the mutated dataset, the local algorithm performed the best in terms of score with an average of 6694.6 points, and the semi-global algorithm performed the best in terms of time, with an average of

56.227 seconds. For score, the worst algorithm was the semi-global algorithm with an average of 6211.45 points, and the worst algorithm for time was the Hirschberg algorithm, with an average of 107.09 seconds.

With the long/short dataset, the best performing algorithm for score was the local algorithm again, with an average of 2467.15 points. The best time-performant algorithm was again the semi-global algorithm with an average of 29.701 seconds. The algorithm with both the worst time and worst score in the long/short dataset was the Hirschberg algorithm, with an average score of 473.55 points and an average time of 54.346 seconds.

Based on the data, we can conclude that the local algorithm is the best alignment algorithm in terms of scoring, and that the semi-global aglorithm is the best alignment algorithm for time efficiency, even with a varied pool of datasets.

References

[1] Burset, M., and R. Guigó. "Evaluation of Gene Structure Prediction Programs." *Genome Informatics Research Lab*, Centre for Genomic Regulation, 1996, `genome.crg.es/datasets/genomics96/`.