International Journal of Advanced Technology in Engineering and Science
Vol. No.3, Special Issue No. 01, September 2015
www.ijates.com
ijates
ISSN 2348 - 7550

A RANDOMIZED ALGORITHM FOR FAST SEQUENCE ALIGNMENT

Akash Nag[1], Dr. Sunil Karforma[2]

[1]Research Scholar, [2]Associate Professor, Department of Computer Science,
The University of Burdwan (India)

ABSTRACT

Sequence alignment is one of the most important tools available to molecular biologists in order to find similarities between two or more DNA or protein sequences. However, despite several advances in this field, multiple sequence alignment (MSA) is often a slow process, and still, optimum results cannot be guaranteed. MSA is an important prerequisite for constructing phylogenetic trees, which are often the underlying goal for biologists to trace the evolution of closely related species. In this paper, we propose a randomized algorithm that can produce a sufficiently good MSA, for huge datasets, in a fraction of the time taken by contemporary MSA algorithms.

Keywords: Bioinformatics, Multiple Sequence Alignment, Randomized Algorithms, Sequence Analysis

I. INTRODUCTION

Ever since sequence alignment gained significance, a large number of algorithms have been published. Most of these can be divided into two categories: pairwise sequence alignment algorithms and multiple sequence alignment algorithms. Pairwise sequence alignment is probably more of a theoretical interest only, as most problems deal with multiple sequences. However, we need to understand how pairwise sequence alignment works in order to proceed to multiple sequence alignment. Compared to MSA, pairwise sequence alignment algorithms can produce optimum results because of the small size of their input sequences. Several dynamic programming algorithms are available for this: the most famous being that of Needleman and Wunsch [1] and that of Smith and Waterman [2]. The former deals with global alignments while the latter with local alignments. Both of these produce the best possible alignment for the two given sequences. Several improvements in time complexity of these algorithms were made in later times, most notably by Gotoh [3] and then by Altschul and Erickson [4].

As DNA sequencing techniques improved, sequence databases started growing exponentially. A new problem was introduced: database search. Given a query sequence, an algorithm was required that could give out the most closely matching sequence in the database. A naïve solution would be to perform a pairwise sequence alignment of the input sequence to every sequence in the database, and then reporting the entry for which the alignment score was the highest. The alignment score is the sum of matches of nucleotides (or amino-acids in case of proteins) at each position of the aligned pair, reduced by the penalty score for mismatches and gaps. However, the time complexity of the dynamic programming algorithms, as well as the huge size of the databases

prevented this solution from being feasible. A new type of algorithm was required that could scale well to huge datasets, and the FASTA algorithm was introduced by Lipman and Pearson [5]. It was based on heuristics, and as such, it did not produce the most accurate alignments. However, it was sufficiently accurate to be used for database search. A few years later, BLAST [6], an extremely fast heuristic algorithm, replaced FASTA as the workhorse of the biological community, for database search.

II. THE PROPOSED ALGORITHM

2.1 Pairwise Sequence Alignment

Before proceeding to discuss MSA, we need to explore a randomized pairwise sequence alignment algorithm, on which our proposed MSA is based. This pairwise sequence alignment algorithm, as presented by Nag and Karforma [7], starts off by chopping the pair of input sequences into sub-sequences. Each such pair of sub-sequences are then aligned, and the final alignment is thus simply the concatenation of the aligned subsequences. The randomness in the algorithm comes from the fact that the size of the subsequence chosen is completely random. Moreover, the lengths of the subsequences in any pair may be unequal. The alignment phase proceeds by sliding a subsequence against its counterpart, and calculating the alignment score for each such sliding configuration. The alignment pertaining to the configuration with the highest score is selected, and the process continues to the next subsequence-pair. However, with each configuration, some amount of residue from any one of the pair of subsequences may be returned back to the original, as yet unscanned, portion of the sequences. In Fig. 1, we see the various possible sliding alignments between the two DNA sub-sequences: ATTAGC and TTCGAA, which are possibly part of a pair of much larger DNA sequences. The best possible alignment has a match score of 3, and the thin line shows the residue. Anything to the right of that line will not be included in the current alignment. This constitutes the nucleotide A of the 2nd string in this case, which will be returned to the unsliced portion of the second string, and reconsidered for the next slicing into two substrings.

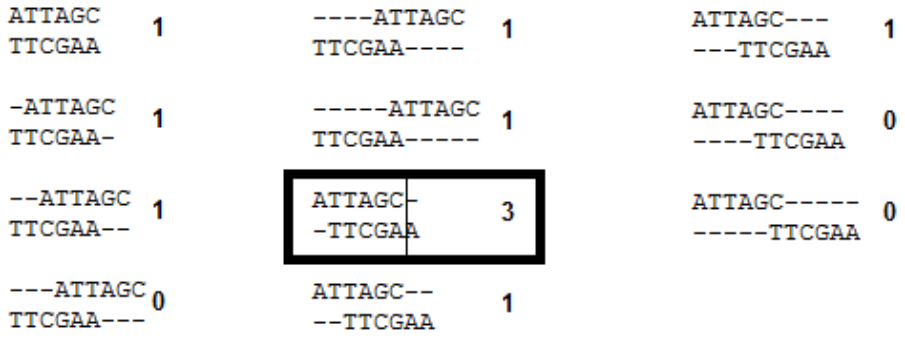

Fig. 1. Possible Sliding Alignments Between Two DNA Subsequences Along with Their Match Scores. The best sliding configuration is highlighted.

The pairwise sequence alignment discussed here is important because of two reasons:

1. It has the best space complexity of any alignment algorithm. Space complexity may not be an issue in pairwise alignment, but is very important in MSA because of the large number of sequences that needs to be aligned.
2. It forms the basis for our proposed multiple sequence alignment algorithm

2.2 The Proposed Multiple Sequence Alignment Algorithm

2.2.1 Random Shuffling

The algorithm begins by shuffling the given input sequences using the Durstenfeld-Fisher-Yates shuffle [8] algorithm, aligning each configuration, evaluating the alignment score for that particular combination, and then returning the aligned sequences with the highest score. This process is customizable in that, the user can select the number of shuffling rounds to be performed. By default, the algorithm performs the shuffling five times.

2.2.2 MSA Scoring

The scoring system used in our algorithm is the Sum-of-Pairs Score for each pairwise alignment. The score takes into account three types of penalties: periphery-gap-penalty, gap-opening-penalty, and the gap-extension-penalty. Usually, the periphery-gap-penalty is set to zero, as we are interested in local alignments only.

$$PairwiseScore(S_1, S_2) = \sum_{i=1}^{n} B(S_1^i, S_2^i) - Penalty$$

$$MSAScore = \sum PairwiseScore(S_i, S_j) \forall 1 \le i \le n, i < j \le n$$

2.2.3The Alignment Process

The alignment process follows a divide-and-conquer (DAC) approach where at each step, the set of sequences to be aligned is divided into two halves, and the MSA algorithm is applied on each set. The division into subsets continues till the number of sequences in a set reaches two. After that, the heuristic algorithm for pair-wise sequence alignment due to Nag and Karforma [7] is used for aligning the pair.

During the merging phase of DAC, a representative string is computed from each of the 2 sets – x_0 and x_1. A representative of a set of sequences is the string containing the character occurring with the highest frequency at each column position. These two representatives are then aligned multiple times. The number of rounds can be customized by the user; by default, it is set to five. The best sliding-configuration or alignment is selected for which the score is the highest. It is clear that one of the shift-value (of the selected alignment) in the pair must be zero.

Each sequence in the set x_i is then shifted by s places (i.e. gaps are inserted to the left) – s being the non-zero shift-value of the selected alignment, and for each sequence in the set $x_{1-i,}$ gaps are inserted to the right to equalize the lengths of the two sets. This gap insertion process is repeated for each *k*-mer in each set separately and not on the whole sequence. When all *k*-mers have been accounted for, the resulting shifted sequence sets are merged to form the complete set, whose MSA score is then calculated. The best complete-set is returned by the merging phase whose score is the maximum.

2.2.4 User Parameters and Scoring Matrix

The algorithm accepts two tunable parameters from the user: *ROUNDS* and *RANDOM-ROUNDS*. The *ROUNDS* parameter is used to tune up the pair-wise alignment algorithm, while the *RANDOM-ROUNDS* parameter determines the number of shuffling rounds to be used. For protein sequences, BLOSUM-62 was used as the scoring matrix.

III. RESULTS AND DISCUSSION

Since the proposed algorithm is a randomized one, performance can vary with the particular sequences to be aligned. The average *k*-mer length chosen at each subdivision step, being random – it has a visible effect on the

quality of the alignments produced. When implemented, the algorithm successfully aligned 5000 sequences, each of average length 1000, in 1 minute on a dual-core 2.3GHz CPU.

The recurrence relation for the running time of the sequence alignment algorithm is given by:

$O(X.T_{MSA}(n))$, where,

$$T_{MSA}(n) = \begin{cases} R.m.W & n = 2 \\ 2T_{MSA}\left(\frac{n}{2}\right) + n.m + R\left\{m.W + n\left(1 + \frac{m}{W}\right)\right\} & n > 2 \end{cases}$$

Where,

n is the number of sequences to be aligned, *m* is the average sequence length,

R is the user-specified *ROUNDS* parameter, *X* is the user-specified *RANDOM-ROUNDS* parameter,

W is the average *k*-mer length (during subdivision)

IV. CONCLUSION

The MSA algorithm proposed in this paper can be used when fast approximate alignments are required. Our algorithm outperforms the MUSCLE [9] algorithm in terms of speed, but is not as accurate. However, in this paper, our goal is not to produce an accurate and biologically meaningful alignment. Rather, the objective of the proposed MSA algorithm is to only aid in the phylogenetic tree construction in the fastest way possible, specially for huge datasets. Such an approximate tree can then serve as a guide tree for yet another round of MSA. Since the guide tree affects the quality of MSA, it is essential that the guide-tree be sufficiently accurate, yet be fast to construct, as it is not the goal itself but an aid to help generate quality alignments. In such situations, the proposed algorithm can be used to construct a rough alignment, generate a guide-tree using that alignment, and then using that guide-tree to construct the actual MSA.

REFERENCES

[1] Needleman, Saul B., 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 (1970): 443-453.

[2] Smith, Temple F., and Michael S. Waterman. "Identification of common molecular subsequences." Journal of molecular biology 147.1 (1981): 195-197.

[3] Gotoh, Osamu. "An improved algorithm for matching biological sequences." Journal of molecular biology 162.3 (1982): 705-708.

[4] Altschul, Stephen F., and Bruce W. Erickson. "Optimal sequence alignment using affine gap costs." Bulletin of mathematical biology 48.5-6 (1986): 603-616.

[5] Lipman, David J., and William R. Pearson. "Rapid and sensitive protein similarity searches." Science 227.4693 (1985): 1435-1441.

[6] Altschul, Stephen F., et al. "Basic local alignment search tool." Journal of molecular biology 215.3 (1990): 403-410.

[7] Nag, Akash, and Sunil Karforma. "A Space-Efficient Approach towards Distantly Homologous Protein Similarity Searches." International Journal of Advanced Research in Computer Science 6.2 (2015): 19-22.

[8] Durstenfeld, Richard. "Algorithm 235: random permutation." Communications of the ACM 7.7 (1964): 420.

[9] Edgar, Robert C. "MUSCLE: multiple sequence alignment with high accuracy and high throughput." Nucleic acids research 32.5 (2004): 1792-1797.