International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 – 5161

Available at http://inpressco.com/category/ijcet

Research Article

Analysis of all Local Pairwise Sequence Alignment Algorithms - Survey

Suchindra Suchindra† and Preetam Nagaraj†

†1Department of Engineering, National Institute of Mental Health and Neurosciences, Karnataka State Govt, Bangalore, India
‡2Department of Engineering, IBM, Bangalore, India

25 March 2023, Accepted 10 April 2023, Available online 13 April 2023, **Vol.13, No.2 (March/April 2023)**

Abstract

Biological sequence alignment is common today and are used in a variety of fields ranging from Bioinformatics, Computational Biology, Genome analysis, Cancer research, Stem Research and many more fields. Most of these fields use the sequence alignment to find the 'similar' regions or similarities between organisms. Since, this step is computational heavy, today, there are specialized hardware to help speed up and techniques and strategies to help speed up or improve the sensitivity (quality) of the alignment in general. The early successful algorithms in sequence alignment were focused on quality, and it produced an optimal algorithm called SmithWaterman algorithm, which we will discuss in detail later using a technique called 'Dynamic Programming'. The time complexity of this algorithms was O (mn). Later, to speedup, heuristic algorithms were developed. Heuristic algorithms gave up a little bit on the quality for speed, by calculating the near-optimal alignment rather than optimal algorithm. In this paper, we will analyze various computational approaches for local sequence alignments.

Keywords: *Bioinformatics, Computational, Dynamic, Heuristic.*

1. Introduction

A sequence alignment is one of the important tasks in certain biological solutions or tasks we mentioned earlier. A sequence in general could be an RNA, DNA, or a protein sequence. A sequence is represented by a sequence of characters that represent their amino acids. For example, DNA (A, C, G, T) , RNA (A, C, G, U) and protein molecules (A, R, N, D, C, Q, E, G, H, I, L, K, M, F, P, S, T, W, Y, V) can be re represented as strings of letters from their alphabet set (Reddy, 2009), (Haque, et al, 2009), (Haque, et al, 2008), (Reddy and Fields, 2020).

A sequence alignment is primarily a task of finding similar regions of match between them. These matches throw major information between them, these include the evolutionary relationship - their common ancestor, the conserved region – which could throw light on their functional, structural, and visual relationships.

If the sequence characters match at the same position in both the sequences, them they are represented by a straight line shown in Fig 1. If they differ, then they are represented by a character ' - ', called indel. An indel represents a missing same character and signals a biologist a divergence from the other sequence. Two given sequences could be of different length although they might be homologous sequences. By homologous sequences, we mean, they are sequences from a same genome E.g., cat family, lion, and Cheetah. So, a letter can be matched with either an indel or a same character in an alignment.

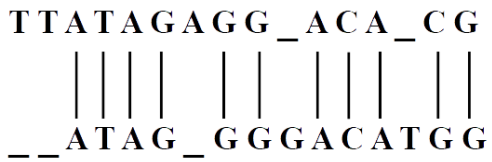

Fig 1 Example of a sequence alignment. (Reddy and Fields, 2020)

In Fig 1, we see regions, where the two sequences are aligned perfectly, these regions are what a biologist call 'similar region. In some regions, indels '-', are present (Reddy and Fields, 2020).

All sequence alignment algorithms can be classified into pairwise and multiple sequences alignment algorithms. Pairwise sequence alignment as the name says, is an alignment between two sequences (Reddy, 2009), (Haque, et al, 2009), (Haque, et al, 2008), (Reddy and Fields, 2020). The objective of a Pairwise sequence alignment is used to find conserved regions between them or a divergence between them depending on the sequence's perspective. Fig 1 shows a pairwise sequence alignment.

Multiple sequence alignments on the other hand, find common regions of similarity between multiple sequences. Here, an alignment between 3 or more is considered multiple sequencing. Some of these multiple sequence alignment algorithms use pairwise sequence alignment as a first step in many bioinformatics solutions. Hence the importance of a pairwise sequencing is quite pivotal from both the quality (sensitivity) and speed perspective.

*Corresponding author's ORCID ID: 0000-0000-0000-0000
DOI: https://doi.org/10.14741/ijcet/v.13.2.11

Most multiple sequence alignment algorithms are heuristic based and ClustalW family of programs by Higgins (Higgins, 1988), (Higgins, 1992) (Higgins, 1994) uses a version of the earlier solution by (Feng and Dolittle, 1987). Others are based off Hidden Markov method proposed first by Krogh, 1994). Others are based on a technique called 'iteration based'. A progressive iteration-based algorithm is called so because they all start with 2 sequences, align them, and progressively add more sequences to the already alignment sequences and ultimately arrive at a final alignment. A popular iteration-based algorithm is called MUSCLE (multiple sequence alignment by log-expectation). This algorithm improves on other previous progressive methods by accurately measuring how distantly they measure to assess the relatedness of two sequences (Edgar, 2004).

A. Pairwise Sequence Alignment Classification

All Pairwise sequence alignment algorithms can be classified into local and global sequence alignment algorithms. A local sequence alignment aims at finding regions or best subsequence between the two sequences in hand. A global sequence finds regions of similarity between the entire length of the two sequences (Reddy and Fields, 2020). Of the two, local sequence alignment algorithms are faster than global sequence algorithms as they are trying to find subregions and not regions or sections of similarity in entirety (Reddy and Fields, 2020).

Popular local sequence algorithms are optimal algorithm Smith-Waterman (Smith and Waterman, 1981), FASTA (Lipman and Pearson, 1985), BLAST (Altschul, et al, 1990), GappedBLAST (Altschul, et al, 1997), BLASTZ (Schwartz et al, 2000), PatternHunter[16], YASS (No and Kucherov, 2005), (Singer and Lambda, 2004), USearch (Edgar, 2010), LAST (Kiełbasa, 2011), and ALLAlign (Wachtel, 2016). This paper focuses on analyzing the local sequence alignment algorithms in detail.

Famous global sequence alignment algorithms icnclude Optimal and Heuristic based are Needleman and Wunsch (Needleman and Wunsch, 2007), GLASS (Batzoglou, 2000) , WABA (Kent, and Zahler, 2000) , AVID (Bray, 2002), AlignMe (Stamm,et al, 2013), MUMmer (Delcher, et al, 1999), LAGAN & MultiLAGAN (Brudno and Morgenstern, 1999), respectively.

Some of the pairwise algorithms find their usefulness in multiple sequence alignment (Reddy and Fields, 2022) and with the advent of AI in major fields today, we find similar techniques used in modern pairwise sequence alignments. However, both multiple sequence alignment and AI based sequence alignment algorithms are beyond the scope of this paper.

2. Background

In this section we will talk about the popular algorithms in local sequence alignment. We first start the section with some basic terminology used in the literature and we then analyze the algorithms in detail.

B. Sequence algingment terminology

A sequence is a set of characters written left to right such that each character in the sequence occupy a unique position of the sequence (Reddy, 2009). Many algorithms use a scoring function to quantify the alignment (Reddy, 2009), (Waqar, et al, 2009), (Waqar, et al, 2008), (Reddy and Fields, 2020). This scoring function set a score for all match pair between the sequences, score for mismatches and score for insertion or deletion (indels). Therefore, when one says the alignment score is 'a', then 'a' is a sum of all matches, mismatches and indels.

There are many scoring functions described in the literature. The simplest being the constant function. Meaning there is a constant score 'σ' for mismatches and matches, no matter where the matches or mismatches appear in the length of the two sequences in hand. A scoring function is represented in the form of a matrix with each cell having a score which corresponds to the base pair match or mismatch (online, 2007). PAM (Percentage of Acceptable point Mutations per 108 years) series of matrices (State, et al, 1991) (Dayhoff, 1978) and BLOSUM (BLOcks SUbstitution Matrix) series of matrices (Henikof and Henikof, 1992) are widely used scoring matrices.

Some Biologists believe that mutations are concentrated in the sequence, so much so, there is always a contiguous region after the mutation begins where the characters are either mismatches or deleted. This theory led to another scoring function called gap open score and gap extension score. A gap open score is a score which is fixed irrespective of the location in the aligned sequences, where the first difference shows between the 2 sequences. A gap extension is the indels are gaps inserted to the sequences after the initial mutation is identified. Since a mutation location is more important to the biologists a larger score is associated with the gap open penalty than the gap extension penalty.

When it comes to measuring the performance of each algorithm in the literature, some of the measure used are the maximum score of the alignment (Gusfield, 1997) and percent similarity score (Gusfield, 1997), total column matched score (Needleman and Wunsch, 2007) and score of the filtered region (Bray, 2002).

3. Optimal Pairwise Local Sequence Alignment Algorithm

Smith-Waterman Algorithm is an optimal local sequence alignment algorithm. The algorithm employs a technique called 'Dynamic *Programming',* where a problem is broken into smaller problems and solving these smaller problems recursively (Smith and Waterman, 1981). In this algorithm, all characters of the two sequences are matches to find the optimal score between them. If S and T are the two sequences, then the algorithm builds the optimal alignment between S and T by using the following formula,

$$V[i;j] = \max\begin{cases} V[i-1;j] + \sigma(S[i-1];-)\ i > 0; j \geq 0 \\ V[i-1;j-1] + \sigma(S[i];T[j])i; j > 0 \\ V[i;j-1] + (-;T[j-1])i \geq 0; j > 0 \end{cases}$$

Bear in mind, there is a score associated with each character match, and ultimately, an optimal score alignment or a best score is calculated by using a procedure called 'back tracing' to find the optimal path (Smith and Waterman, 1981).

4. Heuristic Algorithms

Optimal algorithms are very slow. To overcome this short comings, Heuristic algorithms were developed. All heuristics algorithms produce a near optimal algorithm and are based on the inference or observation that there are regions which are conserved in both the sequences which have high alignment score. So, these optimal algorithms are first finding these conserved regions quickly and building the final alignment arounds these conserved regions.

A. Common Strategies

Most of the heuristic algorithms find regions of similarities which they call 'seed'. So, a 'seed' is a conserved region in the sequence of length 'l'. A seed could be a maximum subsequence in both the sequences. This is shown in Fig 2.

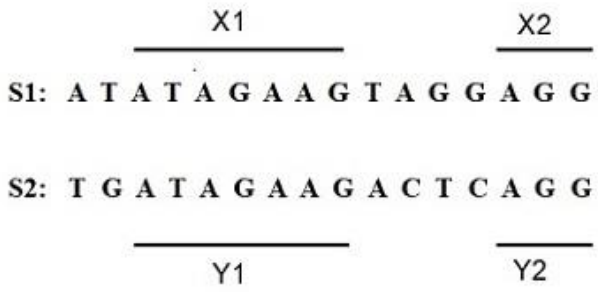

Fig 2. Seed or conserved regions in S1 and S2.

In some cases, Heuristic algorithms are based on a seed with some mismatches in them. Let look at Fig 3. In this figure, we see that, there is a mismatch between X1 and Y1, the number of mismatches in this seed is one. In the literature, later, we will talk about these mismatch seeds with 'x' mismatches of length 'l', where 'l' is the length of the seed including both the conserved and mismatch characters.

Some algorithms consider seeds which have a score above a threshold 'd'. This score is based off any of the scoring matrices we talked about in the previous sections. An example of a seed above a threshold is shown in Figure 4.

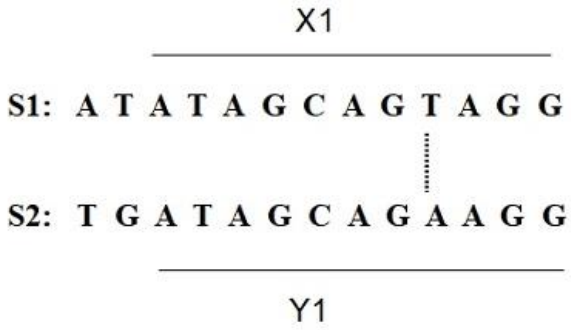

Fig 3: Mismatch seed with one mismatch.

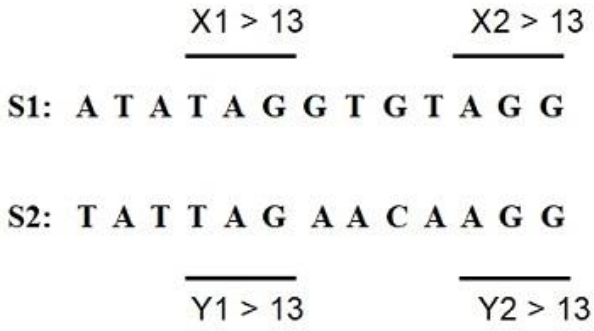

Fig 4: Seed above a threshold score 13.

In figure 4, we see two seeds who score is above a threshold score of 13. These two instances are taken into consideration in some heuristic algorithm for the final alignment. Bear in mind, this a score associated with a seed.

A seed can also be represented by 1's and 0's. In fig 4, beginning from position 4, or the 4th character, a seed making up for (TAGGTGTAGG) in sequence S1 and (TAGAACAAGG) in sequence S2 can be represented as 111000011 with four mismatches. Such a seed is called spaced seed with weight 5. Meaning 5 matches and 4 mismatches with a total length of 9. Another example is shown below in fig 5.

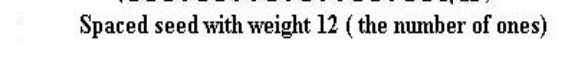

Fig 5: Spaced seed with weight 12 and length 18.

Since the seed selection is an important step in the heuristic algorithms, the way these seeds or subregion or conserved regions are found is also important. The way these seeds are found directly affects the speed of the algorithm. From the literature there are two ways of finding these regions, first is called a look-up table data structure based and second being a tree data structure.
A look-up table is a data structure which is usually an array, vector, or a linked list. The idea is all seeds which are present in the sequence and their position is first known and hashed using a hashing function. Once the look-up table is established for sequence S1, then one

can linearly move across sequence S2 and find all the seeds of length 'l' and their position while moving across sequence S2.

The second approach is using a data structure called tree, most specifically a version of tree called suffix tree. Suffix tree represents the "internal structure of a string in a comprehensive manner" (Gusfield, 1997). The main advantage of using this tree is that it can find exact matching strings in linear time O(n), where n is the length of the string. Weiner developed the first linear time suffix tree back in 1973 (Weiner, 1973) and was improved later by (McCreight, 1976)(Ukkonen, 1995).

B. FASTA

FASTA (Lipman and Pearson, 1985), which stands for FAST-ALL was the first algorithm to use a lookup table to find all seeds. Also, the seeds it found were perfect match seeds of length 'k' also known as k-mer seeds. Initially, the algorithm finds all k-mers in both the sequences and stores their positions in a look up table.

In the next step, the algorithm uses a 'diagonal' method, where all seeds along the diagonal between the two sequences are identified. FASTA stores 10 best diagonal seeds along the diagonal. In the subsequent step, groups of seeds with high scores along the diagonal are saved. There could be many diagonal paths.

All such diagonals are then combined into a single alignment allowing spaces by constructing a directed weighted graph around the seeds (Reddy, 2009), (Haque, et al, 2009), (Haque, et al, 2008), (Reddy and Fields, 2020) (Lipman and Pearson, 1985). The maximum weighted graph is then selected, and the best alignment found is then marked as 'initn' (Reddy, 2009), (Haque, et al, 2009), (Haque, et al, 2008), (Reddy and Fields, 2020) (Lipman and Pearson, 1985).

FASTA then proceeds to build a narrow band of width 'k' centered along the diagonal. FASTA then computes an optimal local alignment in this band by using Smith-Waterman algorithm (Lipman and Pearson, 1985).

C. BLAST

Basic Local Alignment Search Tool BLAST (Altschul, et al, 1990), also uses a look-up table to identify seeds of length 'k' and is faster than FASTA (Lipman and Pearson, 1985). BLAST differs from FASTA on how the seeds are identified. A Sliding window technique is employed to find all good neighbors seeds for every k-mer seed in either direction (Altschul, et al, 1990).

When all seeds and their neighborhood seeds are found, it then extends the seeds in either direction without introducing any gaps. Now, in this step, the alignment score can increase or decrease. When the alignment drops below a threshold 't', the extension is stopped. Such a segment pair is called a high scoring segment pair (HSP).

When all HSP are found, BLAST now extends with gaps (indels) around these HSP's. using Smith-Waterman algorithm until the resultant score falls again below a threshold 'r'. The best HSP is taken then and is the output the two sequences.

When all seeds are found, it then proceeds to find the seeds (HSP, high scoring pairs), and extend them until they fall under a threshold score 'k'. These HSP are then stitched using a restricted dynamic programming which is a version of Smith-Waterman Algorithm (Smith and Waterman, 1981).

BLAST 2 [], is an interactive solution of BLAST and the only difference here is that it produces a gapped alignment by using dynamic programming versus the ungapped alignment in the earlier version to extend the HSPs. BLAST2 is mostly used to compare two sequences that are homologous.

D. Gapped BLAST

Gapped BLAST or PSI-BLAST (Altschul, 1997) is an upgraded version of BLAST which is 3 times faster than BLAST. It employs a method which convert all statistically significant alignments into a position-based scoring matrix. To speed up, this algorithm a '2 hit method', where two non-overlapping words along the diagonal are chosen and they both need to be within a distance 'k' from each other.

E. BLASTZ

BLASTZ (Schwartz et al, 2000), is the fastest among the BLAST family of algorithms, and it employs a different method. All repeat seeds in the sequence are removed [15]. It then looks for smaller seeds of length 'l' with almost one-character transition or one mismatch (Reddy, 2009), (Haque, et al, 2009), (Haque, et al, 2008), (Reddy and Fields, 2020) (Schwartz et al, 2000).

All seeds are then extended on both sides. For regions in between the seeds, it employs smaller seeds and uses optimal alignment to stitch these seeds to form the final alignment (Reddy, 2009), (Haque, et al, 2009), (Haque, et al, 2008), (Reddy and Fields, 2020) (Schwartz et al, 2000).

F. PatternHunter

PatternHunter (Ma, et al, 2002) introduced a seed called spaced seed which we talked about in the previous strategies section. This seed strategy was used to improve the sensitivity and speed. It uses a combination of priority queues variation of red-black tree, queue and hash table to achieve speed (Ma, et al, 2002).

Once all spaced speed of K-mer with weight 'w' are found. It then finds all these spaced seeds in the diagonal between 2 sequences as in FASTA to find the final alignment (Ma, et al, 2002). The algorithm is written in JAVA, and encounters memory problems for long sequences (Reddy, 2009), (Waqar, et al, 2009), (Haque, et al, 2008), (Reddy and Fields, 2020) (Ma, et al, 2002).

G. SOAP, SeqMAP and MAQ

SOAP (Li, 2008) also makes use of the spaced seed like PatternHunter (Ma, et al, 2002). Here, SOAP (Li, et al,

2008) allows for certain number of mismatches or a continuous gap for aligning a sequence. The best hit with minimal mismatches or smaller gaps is reported and then the rest of the algorithm follows the same technique of FASTA to build the final alignment.

SeqMAP (Jiang and Wong, 2008) and MAQ (Li, et al, 2008) extend the matches to allow k-mismatches in then. On these two, SeqMAP allows up to 5 mismatches and is considerably faster than MAQ.

H. BLAT

BLAT – BLAST like alignment tool (Kent, 2002) (States, et al, 1991) is much faster than BLAST. BLAT is like FASTA and BLAST in that, it searches for K-mer seed. BLAT differs from previous algorithms in the way sequences are indexed. One of the sequences S1 is already preprocessed in the database and all the seeds in that sequences are already known. We then proceed to find all "non-overlapping seeds of S2 are run linearly. If BLAST builds an index of S1 and then scans linearly through the S2 (Altschul, et al, 1990), BLAT has a preprocessed sequence with it seeds and their position in the database, this saves time. After this stage, it then searches for seeds with some mismatch's 'n' in them around the seeds it found earlier (Reddy, 2009), (Haque, et al, 2009), (Haque, et al, 2008), (Reddy and Fields, 2020) (Altschul, et al, 1990) (Altschul, 1997). Then the HSPs of seeds like BLAST and mismatch seeds are extended like BLAST and BLAST2 to form a final alignment.

I. SSAHA

SSAHA (Ning, 2001) which stands for Sequence Search and Alignment by hashing algorithm is new algorithm that performs fast searches on the database containing many gigabytes of data. It achieves this by organizing the database into hash table data structure. This is only possible because SSAHA are now able to exploit faster bigger machines with huge RAM, which enables it to store a hash table that describes the database containing these sequences.

Since it can store such an enormous amount of data and has hashed it, the sequence 1 is already preprocessed in BLAT can now be searched across sequence 2 in hand at a much higher speed than BLAT. To improve sensitivity, all it should do to decrease the k-mer to over wit BLAT.

J. UBLAST

UBLAST (Edgar, 2010) uses a different technique – by finding fewer long subsequences in both the sequences. These subsequences should not only be unique, found least amount of time between the sequences but also long. The rest of the algorithm is very similar to BLAST in the way that, a diagonal is found, and best diagonal subsequence is found.

Later the subsequence is extended on either side until the score falls under a threshold and then the final alignment is produced (Edgar, 2010). The sole aim of UBLAST is the outperform BLAST (Altschul, et al, 1990) and MEGABLAST (Zhang, et al, 2000) which is algorithm from BLAST family.

K. LAST

LAST (Kiełbasa et al, 2011) is recent algorithm. It uses a new type of seed called adaptive alignment seeds; these adaptive seeds vary in length and the number of indels in them so they can be considered as dynamic seeds. These adaptive seeds can be of different lengths and weight (Kiełbasa et al, 2011).

By weight, a score associated with the seed if they have a match and mismatches. The rest of the algorithm is very similar to BLAST. To improve the sensitivity one can, reduce the k-mer in the adaptive seed and decrease the number of indels to zero. If one wants speed, then one can increase the k-mer length and indels in the seed. This is truly a dynamic seed in that, there is no prescribed length here, just a group of adaptive ranging from a – b in length with m – n number of indels in them (Kiełbasa et al, 2011).

ALLAlign (Wachtel, 2016) is a new algorithm developed, however literature of this AWS based web algorithm is very limited.

L. LAMBDA

LAMBDA (Singer and Lambda, 2004) is new algorithm which is optimized for protein sequence alignment. It implements a technique where there are more than 1 protein sequences as the target sequences to be aligned with a pre indexed database set of all other know sequences [18]. It is optimized for big or large biological data and uses a Suffix tree to get the maximal common subsequences or maximal unique sequences (Singer and Lambda, 2004), (Reddy and Fields, 2020) and then goes about aligning these subsequences against a pre indexed database (pre indexed based off suffix array) (Singer and Lambda, 2004).

M. MASAA

MASAA (Reddy, 2009), (Haque, et al, 2008), introduced in 2008 is based on Ukkonen [38[suffix tree. The algorithm uses double indexing and back tracking and identifies maximum match subsequences (MMSS) (Haque, et al, 2008). In the subsequent stages, it finds perfect and near perfect seeds and stitches the local alignment to produce the final alignment.

MASAA – S (Reddy and Fields, 2020) was introduced in 2019 which is like MASAA but uses adaptive seeds in between the MMSS, In the later stages it uses perfect seeds to improve sensitivity (Reddy and Fields, 2020). The algorithm is also more sensitive than MASAA but comparable in speed to MASAA (Reddy and Fields, 2020).

4. Experimental Results

In the experimental results, we randomly generated sequences whose length is from 100k to 500k and compared the speed of alignment than others [Tatusova, 1999). For smaller sequences the speed is much faster

than BLASTZ and compares well with other previous algorithms. However, when the sequences grow, then the speed of algorithm get slower. We believe it will perform slower as the length of the sequence increases further the algorithms would continue to be to maintain the same trajectory. We did not compare get to compare the other algorithms in the literature, as it was sometimes difficult to get a copy of their algorithms and at times the difficulty of exercising the comparing. For example, AllAlign needed an index database first and for a randomly generated sequences it was challenging. For AllAlign, we could not find source code to download and compare, checking the performance of the sequence on a server was not clinical. Although we know LAMBDA is 500x faster than BLASTZ, here we assume that LAMBDA is faster than FASTA, BLAST and BLAT too.

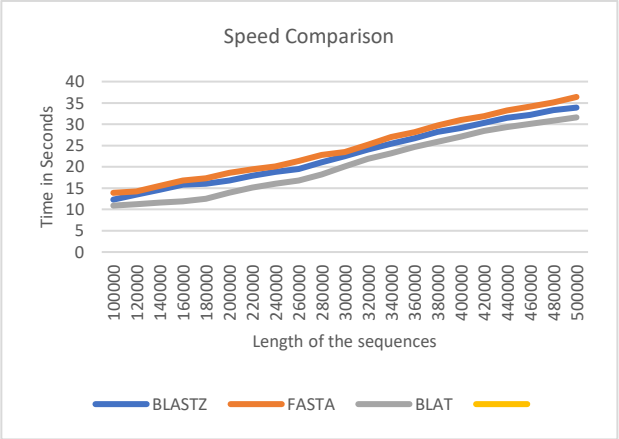

Fig 5: Speed Comparison on different sequence length

For sensitivity, we compared the exon coverage from all four algorithms. We compared the performance of three algorithms on the small dataset of genes, most of these genes are not longer than few thousands. The idea was to see if the algorithms can extract full genes or partial genes and compare the exon coverage. Table 1 shows the percentage of exon coverage, here we think BLASTZ and FASTA are more of less the same while BLAT performs poorly, and this is expected as BLAT is designed for speed and not sensitivity.

Table 1 Sensitivity comparison on genes

Algorithm	% of exon coverage		
	100 exon	***90 exon***	***70 exon***
BLASTZ	94	97	98
FASTA	94	99	99
BLAT	94	95	94

Conclusions

Pairwise sequence algorithms are very important and therefore has been an active field of research. It is also due to many private firms employing computational biology for commercial purposes. This and technological revolution in computer hardware are opened the gates where in previous older algorithms which were sensitive but slower now can be made faster using superior hardware.

Although most of the algorithms concentrate in seed finding techniques, database preprocessing or hardware improvement, we believe that we have not seen the end of any of the above 3 strategies yet. Therefore, we believe there is enough research to find new data structure to speed up the seed's identification and new seeds but also enough research in parallelizing above algorithms for better performance employing clusters in the future.

References

[1]. Reddy, B. G. (2009). *Multiple Anchor Staged Local Sequence Alignment Algorithm-MASAA*. University of Northern British Columbia.

[2]. Waqar Haque, Alex Aravind, and Bharath Reddy. 2009. Pairwise sequence alignment algorithms: a survey. In Proceedings of the 2009 conference on Information Science, Technology and Applications (ISTA '09). ACM, New York, NY, USA, 96-103. DOI=http://dx.doi.org/10.1145/1551950.1551980

[3]. Haque, W., Aravind, A.A., & Reddy, B. (2008). An efficient algorithm for local sequence alignment. 2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 1367-1372.

[4]. Reddy, B, Fields, R. 2020, Multiple Anchor Staged alignment algorithm – Sensitive, 2020, In proceedings with The International Conference on Information and Computer Technologies (ICICT), 2020, San Jose. USA.

[5]. Higgins, D.G. & Sharp P.M. (1988), "CLUSTAL: a package for performing multiple sequence alignment on a microcomputer", Gene, vol. 73,pp237-44. ACEIT Conference Proceeding 2016, IJCSIT-S96

[6]. Higgins, D.G et al. (1992), "ClustalV—improved software for multiple sequence alignment" Comput. Appl. Biosci., vol. 8, pp. 189-91.

[7]. Higgins, D.G et al. (1994), "ClustalW—improving the sensitivity of progressive multiple sequence alignment through sequence weighting, position-specific gap penalties and weight matrix choice", Nucl. Acid Res., vol. 22, no. 22, pp 4673-80.

[8]. Feng D. & Doolittle R. F (1987), "Progressive sequence alignment as a prerequisite to correct phylogenetic trees"), J. Mol. Evol., vol. 60, pp 351-360.

[9]. Krogh, A. et al. (1994), "Hidden Markov models in computational biology: applications to protein modeling", J. Mol. Biol., vol. 235, pp. 1501-1531.

[10]. Edgar C Robert (2004), " MUSCLE: multiple sequence alignment with high accuracy and high throughput" Oxford Journals Science & Mathematics Nucleic Acids Research Volume 32, Issue 5Pp. 1792-1797.

[11]. Smith, T.F. and Waterman, M.S. 1981. Identification of common molecular subsequences. J. Mol. Biol. 147:195-197.

[12]. Lipman D. and Pearson W, 1985. Rapid and sensitive protein similarity searches. Science, 227: 1435, 1441.

[13]. [Altschul, S.F.,Gish, W., Miller, W., Myers, E. W., and Lipman, D. J. 1990. Basic local alignment search tool. J. Mol. Biol. 215: 403-410.

[14]. Altschul, S.F., Madden. T. L, Schaer , A. A., J. Zhang, Z. Zhang, Miller W., and Lipman D. J, 1997. Gapped BLAST

and PSI-BLAST: a new generation of protein database search programs. Nucleic Acids Research,25(17): 3389, 3392.

[15]. Schwartz, S., Zhang, Z., Frazer, K.A., Smit, A., Riemer, C., Bouck, J., Gibbs, R., Hardison, R., and Miller, W. 2000. PipMaker-A web server for aligning two genomic DNA sequences. Genome Res. 10: 577-586.

[16]. Ma, B., J. Tromp, and M. Li, (2002). Patternhunter: Faster and more sensitive homology search. Bioinformatics 18, 440-445.

[17]. [No, Laurent and Kucherov, Gregory, 2005. YASS: enhancing the sensitivity of DNA similarity search. Nucleic Acids Res.

[18]. Singer, H. Hauswedell, J. &, Lambda. K. Reinert, (2004), the local aligner for massive biological data. Bioinformatics 30, i349–i355 .

[19]. Edgar R.C (2010). Search and clustering orders of magnitude faster than BLAST. Bioinformatics. 2010;26(19):2460–2461. doi: 10.1093/bioinformatics/btq461.

[20]. Kiełbasa SM, Wan R, Sato K, Horton P, Frith MC. Adaptive seeds tame genomic sequence comparison. Genome Res. 2011 Mar;21(3):487-93. doi: 10.1101/gr.113985.110. Epub 2011 Jan 5. PMID: 21209072; PMCID: PMC3044862.

[21]. Stamm M, Staritzbichler R, Khafizov K, Forrest LR (2013) Alignment of Helical Membrane Protein Sequences Using AlignMe. PLOS ONE 8(3): e57731. https://doi.org/10.1371/journal.pone.0057731.

[22]. Wachtel. E 2016. All ALL Local Alignment. http://www.allalign.com/

[23]. Zhang Z, et al, 2000. A greedy algorithm for aligning DNAsequences, J. Comp. Biol , 2000, vol. 7 (pg. 203-214)

[24]. Kent. W. James, 2002, BLAT: The BLAST-Like Alignment Tool. Genome Research, Vol. 12, Issue 4, 656-664.

[25]. Needleman, S.B. and Wunsch, C.D. 1970. A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Mol. Biol. 48: 443-453.

[26]. Batzoglou, S., Pachter, L., Mesirov, J.P., Berger, B., and Lander, E.S. 2000. Human and mouse gene structure: Comparative analysis and application to exon prediction. Genome Res. 10: 950-958.

[27]. Kent, J. and Zahler, M. 2000. The Intronerator: Exploring introns and alternative splicing in C. elegans genomic alignment. Genome Res. 10: 1115-1125.

[28]. Bray Nick, Dubchak Inna and Pachter Lior, Avid: A global alignment program, Genome Research. 2003 13: 97-102; 2002.

[29]. Delcher A.L., Kasif S., Fleischmann R.D., Peterson J., White O., and Salzberg S.L. 1999. Alignment of whole genomes. Nucleic Acids Res. 27: 2369-2376.

[30]. [Brudno, M. and Morgenstern, B. Fast and sensitive alignment of large genomic sequences, 2002.

[31]. Pittsburgh Supercomputing Center. (2007) [Online]. http://www.psc.edu/research/biomed/homologous/scoring_primer.html

[32]. States. D. J., W. Gish, and S. F. Altschul, "Improved Sensitivity of Nucleic Acid Database Search Using Application-Specific Scoring Matrices ," METHODS: A Companion to Methods in Enzymology, vol. 3, no. 1, pp. 66-70, 1991.

[33]. Dayhoff M. O, Schwartz R. M, and Orcutt. B. C, "A Model of Evolutionary Change in Proteins," in Atlas of Protein Sequence Structure Nat'l Biomedical Res., M. O. Dayhoff, Ed. 1978, ch. 5, pp. 345-352.

[34]. H=enikof. S and Henikof J. G, "Amino acid substitution matrices from protein blocks," in Proc Natl Acad Sci, 1992, pp. 10915-9.

[35]. Gusfield. D, 1997, Algorithms on Strings, Trees and Sequences: Computer Science and Computational Biology . Cambridge University Press.

[36]. Weiner. P, 1973. "Linear Pattern Matching Algorithms," in Proc. 14th IEEE Annual Symp. on Switching and Automata Theory, pp. 1-11.

[37]. McCreight. E. M, 1976 "A Space- Economical Suffix Tree Construction Algorithm," J. ACM, vol. 23, pp. 262-272.

[38]. Ukkonen. E. 1995, "On-Line Construction of Suffix Trees ," Algorithmica, vol. 14, pp. 249-260, 1995.

[39]. Altschul. S. F, Madden L T., Alejandro A. Schäffer1, Jinghui Zhang, heng Zhang, Webb Miller2 and David J. Lipman "Gapped BLAST and PSI-BLAST, 1997: A new generation of protein database search programs", Nucleic Acids Research, 1997, Vol. 25, No. 17 3389–3402.

[40]. Tatusova. A. Tatiana, Madden L. Thomas, 1999" BLAST 2 SEQUENCES, a new tool for comparing protein and nucleotide sequences", FEMS Microbiology Letters 174 (1999) 247250.

[41]. States. D. J., Gish.W., and Altschul S. F, 1991, "Improved Sensitivity of Nucleic Acid Database Search Using Application-Specific Scoring Matrices ," METHODS: A Companion to Methods in Enzymology, vol. 3, no. 1, pp. 66-70, 1991.

[42]. Ning Z, et al, 2001, SSAHA: a fast search method for large DNA databases, Genome Res., vol. 11 (pg. 1725-1729)

[43]. Li R, Li Y, Kristiansen K, etal, 2008, "SOAP: short oligonucleotide alignment program.", Bioinformatics 2008;24:713–4.

[44]. Jiang H, Wong WH, 2008 " SeqMap: mapping massive amount of oligonucleotides to the genome", Bioinformatics 2008;24: 2395–6.

[45]. Li H, Ruan J, Durbin R., 2008 " Mapping short DNA sequencing reads and calling variants using mapping quality scores", Genome Res 2008;18:1851–8.

[46]. Reddy, B., Fields, R. 2022. From past to present: a comprehensive technical review of rule-based expert systems from 1980 -- 2021. In Proceedings of the 2022 ACM Southeast Conference (ACM SE '22). Association for Computing Machinery, New York, NY, USA, 167–172. https://doi.org/10.1145/3476883.3520211

[47]. Reddy, B., Fields, R. (2022). Multiple Sequence Alignment Algorithms in Bioinformatics. In: Zhang, YD., Senjyu, T., So-In, C., Joshi, A. (eds) Smart Trends in Computing and Communications. Lecture Notes in Networks and Systems, vol 286. Springer, Singapore. https://doi.org/10.1007/978-981-16-4016-2_9