

Divide and Conquer Approach to Long Genomic Sequence Alignment

Mahmoud Naghibzadeh, Samira Babaei

Computer Engineering Dept.
Ferdowsi University of Mashhad
Mashhad, Iran
naghibzadeh@um.ac.ir
samira.babaei@mail.um.ac.ir

Behshid Behkmal*, Mojtaba Hatami

Computer Engineering Dept.
Ferdowsi University of Mashhad
Mashhad, Iran
behkamal@um.ac.ir
hatami.mojtaba@mail.um.ac.ir

Abstract— optimal alignment of DNA sequences is a reliable approach to discover mutations in one sequence in comparison to the other or to discover the differences between two sequences. Needleman-Wunsch is the most applicable software for optimal alignment of the sequences and Smith-Waterman is the most applicable one for local optimal alignment. Their performances are excellent with short sequences, but as the sequences become longer their performance degeneration grow exponentially to the point that it is practically impossible to align two compete human DNAs. Alignment process is essential in diagnosis of genome related diseases. Therefore, many researches are done or being conducted to find ways of performing the alignment with tolerable time and memory consumptions. One such effort is breaking the sequences into same number of parts and align corresponding parts together to produce the overall alignment. With this, there are three achievements simultaneously: run time reduction, main memory utilization reduction, and the possibility to better utilize multiprocessors, multicores and General Purpose Graphic Processing Units (GPGPUs). In this research, the method for breaking long sequences into smaller parts is based on the divide and conquer approach. The breaking points are selected along the longest common subsequence of the current sequences. The method is demonstrated to be very efficient with respect to both time and main memory utilization.

Keywords—*genome sequence alignment; divide and conquer; longest common subsequence; big genome data*

I. INTRODUCTION

Genomic sequence analysis gained extra momentum since the beginning of human genome project [1]. Nowadays, with the advances in the genome sequencing technologies enormous amount of genomic data is continuously being produced which makes it an explosive big data area. There are variety of purposes such as revealing the relations between genes or proteins, understanding their homology and functionality, deciphering sequences to disclose biological aspects, diagnosing diseases, and producing drugs, in the analysis of this big data. Many diseases are caused by variations in the genome sequences such as many kinds of cancers [2] and variety of disorders related to nervous system's degeneration such as Huntington's disease [3]. Even a repeat polymorphism in one of the genes, i.e., IL-1ra, is

shown to be associated with an increased possibility of osteoporotic fractures [4][5]. Comparing a subject sequence against a reference one to find the differences is commonly used in these analysis. The scientific meaning of comparison here is alignment of sequences and revealing their differences.

Alignment of two sequences is the arranging of these sequences such that similarities and differences are shown in the best possible way while the order of elements of each sequence is preserved. Often, a scoring function is proposed and the quantitative optimal alignment of the two sequences becomes equivalent to minimizing or maximizing this function, depending on the problem being solved and the formulation of the scoring function. Considering the countless number of ways that two sequences can be aligned, when gaps are allowed, the Dynamic Programming (DP) approach is an extremely innovative way of optimally aligning sequences [6]. DP is a Mathematics-Computer optimization problem. A problem has to have the *principle of optimality* [6] property to be considered for being solved using the DP approach. The novelty of the DP idea is in systematic solution of all possible sizes of problems starting from the smallest one and going toward the biggest, which is actually the given problem to be solved. It stores the solution results of smaller problems and in solving a bigger problem it can use the solution results of any of the previously solved problems in an optimized manner. The optimal sequence alignment has the principle of optimality and it is successfully solved by the DP approach. The two most utilized approaches are the Needleman-Wunsch algorithm and the Smith-Waterman algorithm.

Needleman-Wunsch algorithm [7] is often used to compare biological sequences with the goal of computing the similarity score of sequences. It does so by arranging the sequences in such a way that the scoring function is maximized (in some cases the goal is to minimize the overall penalty). In this paper we are interested in global alignments (not necessarily global optimal) of pairs of sequences of DNA, RNA, or other similar genomic sequences composed of the four nucleotides A(adenine), T(thymine), G(guanine), and C(cytosine). The optimization problem is organized as filling an m by n matrix S of scores, where m is the length of one of the sequences and n is the length of the other. $S_{i,j}$ is the maximum score of aligning the i first characters of the first sequence and the j first characters of the second one. This is the clue to the DP nature of breaking the

optimization of a large problem into optimization of smaller problems. At the end, the bottom right corner of that matrix, *i.e.*, $S_{m,n}$, gives the overall score of the alignment. To find the actual alignment, a technique for going from the bottom right corner to the top left corner of the matrix has to be followed. For long sequences, the time complexity of Needleman-Wunsch algorithm which is $O(mn)$ prolongs its execution time, hence continuous efforts are made to make it more practical [8][9][10]. For example, for two human genomes of size 3.2 Giga Nucleotides each, and considering at least 10 operations to fill each cell of the 3.2Giga by 3.2Giga matrix, the number of operations needed to fill the whole matrix is 10^{20} . Using a single processor computer with the power of 10 million instructions per second, it takes approximately 317,000 years to complete the matrix.

Smith-Waterman algorithm [11] is another variant of global sequence alignment. An important aspect of this alignment is that if the scoring value of a cell of the scoring matrix is calculated to be negative, its value is set to zero. This means, in such situations, the partial alignment score of sequences up to this point should not affect the alignment score of the rest of the sequences. The net consequence of this assumption is that Smith-Waterman algorithm can better demonstrate the score of local alignments. As a matter of fact, it better shows the local alignment scores and hence it is often used for finding similar regions of sequences. Smith-Waterman algorithm has the same problem as Needleman-Wunsch algorithm has, *i.e.*, its time complexity and hence its high execution time for aligning long sequences. In addition, its space complexity is also high such that some later researches have concentrated on lowering its space complexity only [12].

One way to reduce the alignment execution time is to break the two sequences into many pairs of smaller sequences, when, possible, and align each pair separately. For example, suppose the length of each of the original two sequences is 1000,000 base pairs, *i.e.*, characters. Using the Needleman-Wunsch alignment algorithm, the number of operations would be proportional to 10^{12} . Let's assume we can break the two sequences into 10 pairs of sequences such that the length of each of the resulting new sequences is approximately 100,000 base pairs, *i.e.*, bps for short. The number of operations for aligning all these 10 pairs is proportional to $10 \times 10^{10} = 10^{11}$ which would need 10 times less execution time compared to the case when the original sequences are aligned. On global alignment of related long genome sequences, many attempts have been made to break pairs of such long sequences into many smaller pairs [8][13]. The current research is in the same direction to higher the efficiency and the quality of globally aligning two genome sequences. It uses the Longest Common Subsequence (LCS) technology to find anchor points, Divide and Conquer (DaC) approach to recursively break pairs of longer sequences, and Needleman-Wunsch algorithm to align every short pair of sequences.

Finding the LCS of two genome sequences has many applications such as phylogenetic construction and analysis, quick search in genome sequences big data, and identification of motifs. The natural approach to solve LCS of a pair of sequences is to formulate it as a dynamic programming problem similar to

Needleman-Wunsch algorithm. However, there are quite newer methods with lower time complexities [14][15].

In this paper, a new divide and conquer approach to genome sequence alignment is proposed. The division is along the LCS of the two sequences which is located approximately in the middle of the two sequences. Only a section in the middle of the two sequences are selected in which the LCS is sought. If the discovered LCS is not long enough the sections length is enlarged until a reasonable length LCS is obtained. The novelties of this approach is highlighted in the following.

- It is much faster than either of Needleman-Wunsch and Smith-Waterman algorithms.
- The LCS subsequence is removed from both sequences, and hence from further processing, which further saves both execution time and memory space.
- It is faster than the state of the art anchor-based methods which also use some kind of division.
- The space requirement of the method is extremely low.
- It has the potential to be implemented in parallel in three different levels, division of the long sequences, alignment of all short pairs of sequences, and utilization of General Purpose Graphic Processing Units (GPGPU) within each alignment of short sequences.

The structure of the rest of the paper is as follows. In Section 2 a short review of related work is presented. Section 3 is a brief clarification of the problem being solved. Section 4 details the implementation of the proposed method's solution approach. Section 5 is the evaluation section and finally a short summary and future work is documented in Section 6.

II. RELATED WORK

Sequence alignment is the canonical point of most tools of DNA and other genome sequences alignments. Human genome sequencing and analysis [1] formally started in 1990 and with it, computational methods, especially alignment, became an important part of any genome analysis activity. Undoubtedly, with the invention of its dynamic programming implementation and also after that, there has been great progress in making the alignment algorithms efficient and up to date. It is worth mentioning that although we have focused on genome sequence alignment, alignment is widely used in other domains such as protein sequences alignment, protein networks alignment [16] and all kinds of text alignment [17].

Needleman-Wunsch algorithm is the basic method for global sequence alignments [7]. Smith-Waterman algorithm is a variant of Needleman-Wunsch algorithm which is also a global aligner but it is tailored to find local similar regions of sequences being aligned [11]. The problem with these algorithms is their high time complexity which is $O(mn)$, or $O(n^2)$ where the length of the two sequences are the same and it is equal to n . In addition, the space complexity of these algorithms is also $O(mn)$ which can be problematic for large sequences. In such cases the solution would be to have part of the scoring matrix in the secondary storage which will further worsen the execution time requirement. Many improvements are reported which we will concentrate on the most recent ones, here.

BLAST is heuristic algorithm developed to search a short sequence in a large volume of data. Based on a hashing mechanism and a local alignment method, it is capable of finding sequences in the database that are similar to the search sequence with some degree of similarity [18]. BLAST is not originally developed for alignment of genomic sequences and it has the potential to be used in any kind of text data with any kind of alphabet. Later, specific program versions were developed for this purpose. BLASTZ and LASTZ are recent versions of the program that are widely used for local optimal alignment of genomic and DNA data. It is much faster than Smith-Waterman method especially for long sequences.

Leimeister et al. [8] proposed a new anchor point finding method called filtered spaced word match. Anchor points are short subsequences in the two sequences which will be matched in the final alignment of these sequences. Subsequences between consecutive anchor points of the two sequences are aligned using known alignment algorithms. They claim that their superiority is in finding better anchor points. However, their comparison with that of Mugsy pipeline [9] did not lead to similar quality for closely related sequences but they claim it is superior in alignment of distal sequences. Neither time nor space complexity of the method is reported. Our guess is that its time complexity would be in the level of that of Needleman-Wunsch but there may be improvements in its space complexity.

Another recent development in the field of long genome sequence alignment is MUMmer4 [10] which is the fourth generation of MUMmer. It is based on a 48-bit suffix array data structure. It is capable of using multicores of the host computer however, this is applicable for the case of aligning many sequences to the reference genome; e.g., aligning many short reads to the human reference genome. In such cases, it can handle very large input size up to 141 Tera bps. Although the most important aspect of an algorithmic computational method is its time and space complexity, these are not reported in the paper.

The research reported by Sun et al. [12] is an effort towards space requirement reduction of Smith-Waterman. Similar to that of Smith-Waterman, its input is a pair of sequences and it performs the optimal local alignment of the sequences. It is capable of aligning long sequences up to 100 million bps. It claims that the space requirement is tremendously reduced but the order of reduction is not reported. It also claims its time complexity is the same as that of Smith-Waterman. However, because of extra computations required to reduce space, one would expect its time requirement would be higher than that of Smith-Waterman.

A recent method called GSAAlign is the last method studied here. It is specifically designed for semi-optimal alignment of long Genome and DNA sequences [19]. Fundamentally, it is composed of three phases: seed identification and pairing of the two sequences' seeds, similar region identification by chaining seed pairs, and finally the local aligning of regions. To produce the overall alignment, local alignment of regions are joined together. It is capable to implement the alignment phase of the process in parallel using a multithreading technique. The authors claim that GSAAlign is the fastest semi optimal aligner of long sequences. They also claim that the developed program

produces perfect or nearly perfect precision and recalls on the identification of sequence variations in the dataset.

III. PROBLEM DEFINITION

Two genome sequences S1 and S2 are inputs to the longest common subsequence divide and conquer approach which is developed here. The length of the sequences are considered to be m and n , respectively. The output would be the global alignment of the two sequences in which differences and similarities of the sequences are clearly recognizable. It is assumed that the sequences are composed of characters A, C, G, and T. The length of the sequences could range from couple of nucleotides up to the length of a whole human genome. For very short sequences, up to 100 bps, the Needleman-Wunsch will directly be used and hence there would be no improvements on these sizes. As the length become longer the efficiency of the presented method with respect to both time and space increases.

IV. LCSDAC IMPLEMENTATION

In this section details of the implementation of the proposed Longest Common Subsequence Divide and Conquer (LCSDaC) alignment of two sequences is explained. The system will be able to align similar sequences of any sizes up to the length of a human genome, i.e., 3.2 Giga bps. If both sequences are short, i.e., less than 100 bps is assumed here, the Needleman-Wunsch will directly be utilized. Otherwise, the dividing and conquer process will systematically break them into many pairs of short sequences. Figure 1 illustrates how this breaking process works. From the middle of each of the sequences a subsequence of length equal to the minimum of 1000 and one third of the current sequence's length, i.e., $\text{Minimum}(1000, n_1/3)$, is distinguished which become the input to the LCS procedure. This procedure will find their longest common subsequence. If this subsequence is long enough the division is successful, otherwise the length of the distinguished subsequences are doubled and the LCS is called again. There is a maximum which is set to 3 for this step and in the worst case the longest common subsequence obtained in the third iteration is accepted as the breaking point. Figure 1 also shows the maximum number of times the LCS is called in each level of the tree.

Figure 2 has illustrated this procedure on a miniature pair of sequences. It is assumed that the first iteration of the procedure gives an acceptable result. In this example, the length of each of the sequences is 50. Although the length of sequences should be more than 100 to perform the division, for this example the division is applied. The LCS procedure will find the longest common subsequence to be GGAGCATGAGCTGG. It is located in Locations 17 to 30 of the first sequence and 19 to 32 of the second sequence. These places are assumed to be aligned in the final alignment of the two sequences and hence they are exempted from further processing. The algorithm remembers this alignment and includes it in the final alignment. In the second level of the tree of Figure 1 we will have to deal with two pairs of sequences to be processed. The first pair is in locations 1 to 16 of the first sequence and Locations 1 to 18 of the second sequence. The second pair is in locations 31 to 50 of the first sequence and Locations 33 to 50 of the second sequence. The division is not continued because the sequences are short. For these two pairs Needleman-Wunsch is directly applied.

The overall pseudocode of the algorithm of the method presented here is shown in Algorithm 1. In Line 2, two new sequences which are the final results of the alignment are obtained. Their length is not necessarily equal to the length of the original sequences and may be a bit longer. The whole process, Lines 3 to 19, is declared as a recursive procedure. In this procedure, whenever one of the two inputs is short, *i.e.*, less than 100 base pairs, Needleman-Wunsch is called to align the

two sequences (Lines 5 to 8). The aligned results would be placed in output sequences *SA1* and *SA2*. In Lines 9 and 10, the SELECT procedure selects a subsequence from the middle of each of the sequences and then the longest common subsequence of the selected sequences are computed (Line 11). A similar procedure has to be applied to either sides of the anchor segments with the LCS being placed in the middle (Lines 12 to 18).

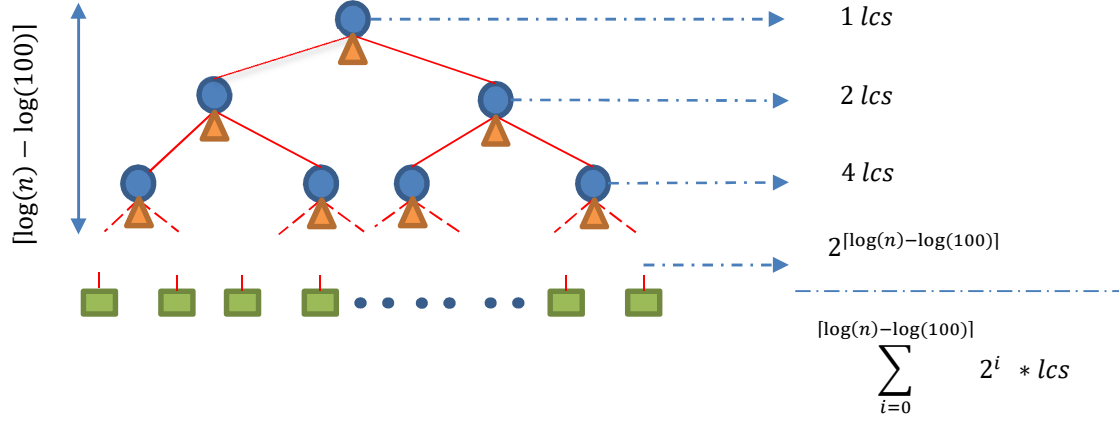


Figure 1. The division process of LCSDaC showing the number of LCS calls

1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50
CCTTTATCTAATCTT**TTGGAGCATGAGCTGG**CATAGTTTGGAAACCGCCCTCA
CCTTTATGTAATCT**TTGTGGAGCATGAGCTGGGAG**TTGGACACGCCCTCA

Figure 2. The effect of applying LCS on two sequences

V. EVALUATION

For the alignment of long sequences, which is the interest of this research, both time and space complexities are two major limitations. Otherwise, Needleman-Wunsch produces the optimal alignment. Accuracy is often scarified to be able to obtain approximate solution in tolerable amount of time and with the available storage capacity. We proceed first with the most important property of the algorithm which is time complexity.

Time complexity of Algorithm 1

The number of operations needed to find the LCS of two sequences of length k is proportional to k , *i.e.*, $C_l k$ where C_l is a constant [15]. On the other hand, the number of times we can divide an integer n by 2 before the final result becomes less than 100 is $\log_2(n) - \log_2(100)$ or approximately $\log_2(n) - 6$. This is the depth of the tree of Figure 1. Therefore, the number of times the LCS procedure is called is shown by Formula (1).

$$\text{Number of LCS calls} = \sum_{i=0}^{[\log(n)-\log(100)]} 2^i \quad (1)$$

Algorithm 1. The pseudocode for LCSDaC approach

```

1- Input Sequences  $S1$  and  $S2$ 
2- Global  $SA1, SA2$  // the two sequences after alignment
3- Recursive Procedure  $LCSDaC(S1, S2)$ 
4- {
5-   if ( $S1 < 100$  OR  $S2 < 100$ ) {
6-      $SA(S1, S2)$  // Short Align and place in  $SA1, SA2$ 
7-     Return
8-   }
9-    $SELECT(s1, S1)$  //  $s1$  is selected from the middle of  $S1$ 
10-   $SELECT(s2, S2)$ 
11-   $LCS(s1, s2)$  // find the LCS of  $s1$  and  $s2$ 
12-   $S1 = \text{left section of } S1 \text{ up to LCS start}$ 
13-   $S2 = \text{left section of } S2 \text{ up to LCS start}$ 
14-   $LCSDaC(S1, S2)$  // recursive call for new sequences
15-   $Append(LCS)$  // append LCS to current end of  $SA1, SA2$ 
16-   $S1 = \text{right section of } S1 \text{ from end of LCS}$ 
17-   $S2 = \text{right section of } S2 \text{ from end of LCS}$ 
18-   $LCSDaC(S1, S2)$ 
19- }

```

Assuming the maximum length of the pair of sequences which is sent to the LCS procedure to be k_1 , the number of operations needed to run the whole algorithm would satisfy Formula (2). In this formula $O(SA)$ represents the time complexity of the Short Align (SA) procedure.

$$T(n) \leq C_1 * k_1 \sum_{i=0}^{\lceil \log(n) \rceil - \log(100)} 2^i + 2^{\log(n)+1} O(SA) \quad (2)$$

Or,

$$T(n) \leq C_1 * k_1 n + n * O(SA). \quad (3)$$

Recall that even if the length of the original sequences to be aligned is as large as a whole genome or any other longer length, in the algorithm, alignment for only short sequences with length less than 100 bps is called. Let's assume that the number of operations needed to do this alignment is k . It is usually very small compared to the length of the input sequences and it is bounded by a constant. However, it is not too small to satisfy the definition of the big O notation of the time complexity to consider it a constant. On the average, for alignment of short sequences there would be $50*50=2500$ cells to fill. As this is represented by k , the time complexity of the method would be $O(kn)$. Of course, there is always an option to set the length of short sequences to be less than 100 bps.

There is a hidden benefit in the proposed method which is the exemption of longest common subsequences of each pair of sequences from any further processing. In the calculation of the time complexity, this is ignored because it does not affect the time complexity itself but reduces the hidden constant

Space complexity of Algorithm 1

Space complexity of the alignment methods is as important as their time complexities. Some recent methods such as Sun et al. [12] have left the time complexity of the aligner untouched and have concentrated on reducing its space complexity. The actual space complexity of their proposed algorithm is not reported because, it is calculated for the worst case and in the worst case it would not be impressive. However, they claim the main memory of a "normal personal computer" would be able to align sequences as long as 100,000,000 nucleotides.

Here we explicitly express that the space complexity of our algorithm is $O(n)$ where n is the length of longer input sequence. There are two input sequences of length say n and two output sequences of approximate length n , too. LCSDaC would need $4n$ bytes to keep them all in the main memory. Within this algorithm, each Needleman-Wunsch execution will require at the most 10,000 cells and considering 4 bytes for each cell, adds up to 40,000 bytes. In the non-parallel version of LCSDaC, there would be only one running Needleman-Wunsch at any given time. Other minor memory requirements are ignorable. Therefore, the total memory space needed is expressed by Formula (4).

$$S(n) = 4n + 40000 \quad (4)$$

For sequences larger than 40000 bps, the space requirement would be

$$S(n) \leq 5n \quad (5)$$

Therefore, the space complexity is $S(n) \in O(n)$. Even for the case of parallel implementation of the LCSDaC using multicores, the number of multicores are limited and it is small, say 16, which still leaves the space complexity to be $O(n)$. This is another achievement of the current research.

The method presented here has a high potential to be implemented in parallel. The simplest section that can be paralleled is the "Short align" section of the algorithm which is responsible for alignment of short sequences. This section corresponds to the lowest level of the tree of Figure 1. Within each short align instance one can utilize General Purpose Graphic Processing Unit to elevate the degree of parallelism. Furthermore, the LCS recognition can become parallel. For example, in Level 2 (third level) of the tree of Figure 1 four LCS instances could run in parallel. In this paper, the whole idea is developed sequentially and the comparisons are with sequential competitors. Actually, we intended to put our idea into practice soon and evaluate the practicality of the method. The parallelization of the algorithm is left for future work.

For short sequences, there is no need for comparison, and the choice is definitely Needleman-Wunsch [7] with optimal alignment capability. For local optimal alignment, the choice is Smith-Waterman [11] for local optimal alignment. For medium length sequences it depends on the available computer, its number of cores, number of General Purpose Graphic Process Units (GPGPUs), and the capabilities of the employed software program. For large and very large sequences the choice is definitely not Needleman-Wunsch or Smith-Waterman. A practical choice is an efficient heuristic semi-optimal methods. Therefore, two such recent methods are selected for the comparison part: MUMmer [10], and GSAAlign [19]. The experiment includes both short genomes and long ones.

BRCA1 is a human gene responsible for suppressing tumors and repairing DNA, ATF6 is a human gene which acts as a transcription factor inside the nucleus, and CFTR gene that is the provider of instructions for making a kind of protein. The approximate sizes of these genes are expressed in thousands (K) of nucleotides in Table 1. These genes are taken from 1000 genomes project dataset. Escherichia coli (E.Coli), Shigella, and Salmonella are three bacteria with the approximate sizes that are expressed in millions (M) of nucleotides in this table. The bacteria sequences are taken from NCBI site. For each gene and bacteria two different variants are selected to be aligned.

The computer used for the experiments is Intel(R) core(TM) i7-353U CPU 2GHz, RAM 6GB, and Linux Ubuntu 18.0 operating system.

TABLE I. SUMMARY OF THE TIMING COMPARISON RESULTS

| Sequence→ Method↓ | BRCA1 #127K | ATF6 #198K | CFTR #430K | E.coli #5M | Shigella #5M | Salmonella #4.9M |
|----------------------|----------------|---------------|---------------|---------------|-----------------|---------------------|
| MUMmer | 5s | 5s | 9s | 328s | 350s | 246s |
| GSAlign | 2s | 3s | 4s | 125s | 240s | 160s |
| LCSDaC | 1s | 2s | 4s | 138s | 218s | 127s |

The overall results of Table 1 for the six tested sequences, show that, on the average, LCSDaC 2.61 times faster than MUMmer, and 1.29 times faster than GSAlign. For example in comparing MUMmer and LCSDaC the following computation is used.

$$(5/1 + 5/2 + 9/4 + 328/138 + 330/218 + 246/127)/6 = 2.61$$

Another major area of comparison is the accuracy of the methods. It is obvious that Needleman-Wunsch is the most accurate one because it is an optimal aligner. The problem arises when the sequences are long and its time and space requirements is absolutely intolerable. Smith-Waterman is not an optimal aligner but, it is a locally optimal one. MUMmer, GSAlign, and LCSDaC fall into the category and none of them could be used as a fully correct one. In the absence of an optimal alignment the number of exact matches of the two sequences are taken to be a measure for the purpose of correctness evaluation. A higher value of this measure is interpreted as the method being more accurate. Table 2 shows that the method presented in this paper is more accurate than others, in all cases. Therefore, the Relative Accuracies (RA) of other methods are computed in comparison to LCSDaC.

Evaluating their accuracy in terms of score, precision, recall, and F-measure requires extensive experiments on numerous sequences which is left for the future work.

TABLE II. RELATIVE ACCURACY COMPARISON RESULTS

| Sequence→ Method↓ | BRCA1 #127K | ATF6 #198K | CFTR #430K | E.coli #5M | Shigella #5M | Salmonella #4.9M |
|----------------------|----------------|---------------|---------------|---------------|-----------------|---------------------|
| MUMmer | 103521 | 152501 | 366248 | 2814666 | 2505372 | 2494108 |
| GSAlign | 98531 | 128951 | 323587 | 2453267 | 2378563 | 2108569 |
| LCSDaC | 123087 | 197277 | 421446 | 2955849 | 2958760 | 2754703 |

Therefore, the average relative accuracies of other methods are computed in comparison to LCSDaC. Details of calculations for MUMmer is shown in the following.

$$RA_{MUMmer} = (103521/123087 + 152501/197277 + 366248/421446 + 2814666/2955849 + 2505372/2958760 + 2494108/2754703)/6 = 0.864$$

Performing a similar calculation for GSAlign evaluates its relative accuracy to be $RA_{GSAlign} = 0.6554$.

Therefore the accuracy of LCSDaC is the highest and that of MUMmer is 86 percent of LCSDaC. With respect to relative accuracy, the GSAlign is the lowest with 66 percent of the LCSDaC.

VI. CONCLUSION AND FUTURE WORK

In this paper, we introduced LCSDaC which is a novel long DNA sequence aligner base on divide and conquer approach in which, the division takes place along the longest common subsequence of the middle portions of the current sequences. The time complexity of the method is analyzed and it is shows to be superior to traditional aligners. The space complexity of the algorithm is also calculated to be $O(n)$ which is better than all classic aligners such as Needleman-Wunsch and Smith-Waterman methods. Further, it outperforms any state of the art method. The sequential version of the presented method is implemented and it is compared against two state of the art heuristic aligners called MUMmer, and GSAlign. It is shown that on the average, the proposed algorithm, LCSDaC, is 2.61 times faster than MUMmer, and 1.29 times faster than GSAlign. For the accuracy we show that the accuracy of MUMmer is 86 percent of LCSDaC and that of GSAlign is 66 percent of the LCSDaC.

REFERENCES

- [1] H. Chial, "DNA sequencing technologies key to the Human Genome Project," *Nat. Educ.*, vol. 1, no. 1, p. 219, 2008.
- [2] J. Lever, E. Zhao, J. Grewal, M. Jones, and S. Jones, "CancerMine: a literature-mined resource for drivers, oncogenes and tumor suppressors in cancer," *Nat. Methods*, vol. 16, pp. 505–507, 2019.
- [3] I. Kovtun and C. McMurray, "Features of trinucleotide repeat instability in vivo," *Cell Res.*, vol. 18, pp. 198–213, 2008.
- [4] B. Langdahl, E. Løkke, M. Carstens, S. LL, and E. Eriksen, "Osteoporotic Fractures Are Associated with an 86-Base Pair Repeat Polymorphism in the Interleukin-1–Receptor Antagonist Gene But Not with polymorphisms in the Interleukin-1b Gene," *J Bone Min. Res.*, vol. 15, no. 3, pp. 402–414, 2000.
- [5] N. Saadati and R. Rajabian, "The effect of bisphosphonate on prevention of glucocorticoid-induced osteoporosis," *Iran. red crescent Med. J.*, vol. 10, no. 1, pp. 8–11, 2008.
- [6] R. E. Bellman, *Dynamic Programming*. New York: Dover Publications Inc, 2013.
- [7] S. Needleman and C. Wunsch, "A general method

- applicable to the search for similarities in the amino acid sequence of two proteins,” *J. Mol. Biol.*, vol. 48, no. 3, pp. 443–453, 1970.
- [8] C. Leimeister, T. Dencker, and B. Morgenstern, “Accurate multiple alignment of distantly related genome sequences using filtered spaced word matches as anchor points,” *Bioinformatics*, vol. 35, no. 2, pp. 211–218, 2019.
- [9] S. V Angiuoli and S. L. Salzberg, “Mugsy: fast multiple alignment of closely related whole genomes,” *Bioinformatics*, vol. 27, no. 3, pp. 334–342, 2011.
- [10] G. MarcÉais, A. L. Delcher, A. M. Phillippy, R. Coston, S. L. Salzberg, and A. Zimin, “MUMmer4: A fast and versatile genome alignment system,” *PLOS Comput. Biol.*, 2018.
- [11] T. Smith and M. Waterman, “Identification of Common Molecular Subsequences,” *J. Mol. Biol.*, vol. 145, no. 1, pp. 195–197, 1981.
- [12] J. Sun, K. Chen, and Z. Hao, “Pairwise alignment for very long nucleic acid sequences,” *Biochem Biophys Res Commun*, vol. 502, no. 3, pp. 313–317, 2018.
- [13] M. Brudno, C. B. . Do, G. M. Cooper, M. F. Kim, and E. et al. Davydov, “LAGAN and Multi-LAGAN: Efficient Tools for Large-Scale Multiple Alignment of Genomic DNA,” *Genome Res.*, vol. 13, no. 4, pp. 721–731, 2003.
- [14] Q. Wang, D. Korkin, and Y. Shang, “A Fast Multiple Longest Common Subsequence (MLCS) Algorithm,” *IEEE Trans. Knowl. Data Eng.*, vol. 23, no. 3, pp. 321–334, 2011.
- [15] M. Sazvar, M. Naghibzadeh, and N. Saadati, “Quick-MLCS: A new algorithm for the multiple longest common subsequence problem,” in *Proceedings of the Fifth International C* Conference on Computer Science and Software Engineering*, 2012, pp. 61–66.
- [16] A. Mir, M. Naghibzadeh, and N. Saadati, “INDEX: Incremental depth extension approach for protein–protein interaction networks alignment,” *BioSystems*, vol. 162, pp. 24–34, 2017.
- [17] Y. Deng, S. Kumar, and W. Byrne, “Segmentation and alignment of parallel text for statistical machine translation,” *Nat. Lang. Eng.*, vol. 13, no. 3, pp. 235–260, 2007.
- [18] S. F. Altschul, W. Gish, W. Miller, E. W. Myers, and D. J. Lipman, “Basic local alignment search tool,” *J. Mol. Biol.*, vol. 215, no. 3, pp. 403–410, 1990.
- [19] H. N. Lin, & W. L. Hsu. “GSAlign: an efficient sequence alignment tool for intra-species genomes.” *BMC genomics*, vol. 21, no. 1, pp. 1-10. 2020.