Journal of Distributed Computing and Systems(JDCS), Vol. 4, Issue 2, 2021, Pages 65-72

Local Sequence Alignment with a Parallel Hash based Model

Seyed Ehsan Parsaeian[1], Nasrin Aghaee-Maybodi*[2]
[1]*Department of Computer Engineering, Islamic Azad University, Yazd Branch,Yazd, Iran.*
[2]*Department of Computer Engineering, Islamic Azad University, Maybod Branch, Maybod, Iran.*

Article History:

Received: 28 November 2021
Received in revised form: 27 January 2022
Accepted: 15 February 2022
Available online: 17 March 2022

Abstract

The two main alignment problems in genetic sequencing mapping are mapping speed and storage space. In recent decades, effective methods are required to speed up the alignment process and reduce storage space due to increasing DNA sequence data. This study is conducted to provide a hash-based search algorithm to reduce data storage space, which is one of the major issues in hash bases, in addition to increased accuracy and speed. Like hash algorithms, the proposed method uses a hash table to store data, but changes the basis of its work to them, especially the Blast family, and uses a method other than Seed-and-Extend. This increases the speed several times. This method uses the compression technique to reduce the storage space so that the data is stored compressed in the hash table and used during the alignment process, there is no need to decrypt the data in the search and mapping steps, and memory saved is preserved until the end of the process. Besides, unlike most hash methods, this method does not include all possible subsequences of the required length in the hash table when creating the table, and the table is created by passing and recording the position of the existing subsequences of the reference sequence. Another feature of the algorithm is the use of the multiprocessor technique to increase the algorithm execution speed so that the two main and time-consuming steps of the alignment process, one is creating a hash table and the other mapping, are done concurrently.

Keywords: Indexing, Sequence Alignment, Mapping, OpenMP.

I. INTRODUCTION

Bioinformatics refers to the knowledge of the application of computer science in the field of biology. Recent advances in molecular biology and related equipment have rapidly increased sequencing in organisms [1]. Sequence alignment is one of the applications of bioinformatics that aims to identify similarities between imported sequences. The first two-sequence algorithm was proposed by Needleman and Wunsch in 1970 [2, 3]. This algorithm is based on dynamic programming and can map two sequences of length m and n in O (mn) time. Although such algorithms guarantee an optimal solution, they are not used in most alignment methods due to their low speed and are replaced by heuristic algorithms. The first heuristic algorithm was introduced by Wilbur and Lipman in the FASTA program [4].

There are many different tools for sequencing alignment, but next-generation sequencing (NGS) experiments suggest that the technology regularly generates exomes or complete genomic sequences of several hundred thousand specimens in a short time. This can help to gradually increase the length of query sequences to reduce errors. So, faster, more sensitive, and more accurate tools are required for short and long query mapping [5]. In this regard, various techniques such as data compression and parallelization have been proposed to help these tools to achieve their goals. The tools can perform the alignment process in two or more sequences. Moreover, the process can be done locally or globally. The global alignment process follows similarities between all two strings, but in local alignment, the goal is to identify similarities between the two sequences. A group of alignment tools can only work with short queries, but some can also work with long queries.

Most tools use an index-based mapping strategy to deal with storage space and mapping speed. This strategy can be based on the Burrows-Wheeler Aligner (BWA) or hash table [6]. BWA-based tools use the FM2 index structure and perform the alignment process using the concept of suffix arrays. One of the characteristics of extension arrays is the lack of efficient management of the problem of mismatch and gaps, but they have better memory consumption [7]. Some of the tools that use the BWA method are Bowtie [8], SOAP3 [9] and [7, 10], BWA, and Bowtie2 [11].

Most hash-based tools, such as Blast, use the Seed-and-extend strategy. They provide candidate regions (K-mer) in search, extension, and scoring reference databases and then provide the best match (K-mer is a substring of length k) [5, 12]. However, in this type of algorithms, 90% of the mapping time is spent on scoring and evaluation, which have the most calculations [13]. Hash-based methods are more accurate, comprehensive, and sensitive to BWA, reducing search time to O (1). Some of these tools are Blast [14], AGILE [15], SSAHA [16], NEXTGENMAP [17], and PatternHunter [18].

*Corresponding Author: Aghaee.Meybodi@maybodiau.ac.ir

These tools have a good speed despite the high memory they need [6, 19].

This study is conducted to perform the alignment process on long queries and to increase the process execution speed several times by eliminating the Seed-and-Extent technique by taking the idea from the SSHAH algorithm. One of the techniques used to reduce memory consumption is to compress data when stored in a hash table so that there is no need to decrypt the data during search and mapping. In this way, the saved memory is preserved to the end. Besides, in the indexing process, the hash table is created in one pass and only the K-mers in the reference sequence are stored in the table, while in most existing methods, the hash table is created in two passes. In the first pass, all possible states of K-mer are generated and placed in the table, and in the second pass, the position of each K-mer is extracted from the reference sequences and recorded in the table. Another feature of this algorithm is to perform both search and mapping concurrently using the OpenMP library so that eventually the nodes can present their results to the user without having to return to the original node and there will be no data dependencies to present the results of the nodes. Accordingly, the innovations and techniques used in the proposed method are as follows:

1. Reducing data volume for storage up to four times by using compression technique and performing search and mapping without decrypting data;
2. Reducing the size of the hash table by storing K-mers in sequences instead of all possible K-mers of length K;
3. Using the OpenMP library parallelization technique to split data and tasks between nodes, speeding up the process, ease of implementation, and high efficiency.

The rest of the study is organized as follows: In Section II, the literature in this field is reviewed. In Section III, the problem is defined. Section IV presents the proposed method. In Section V, the results of the proposed algorithm are presented and compared with the other two algorithms in terms of time and memory consumption in the form of graphs. Section VI presents conclusions.

II. Literature Review

Hash algorithms can use input-based, reference-based, and a combination of both techniques to store their data. Input-based hash algorithms place the index of their query sequence on the hash table in memory and then scan the reference sequence in front of it. In this method, a hash table must be created for each query sequence[20]. The Blast algorithm is known as the mother of hash-based methods and operates in the stages of input preprocessing, search, and evaluation. In the first step, a list of query sequence words is created. It is then measured with possible sequences from the base and scored based on the Blosum matrix. The scores obtained are measured by the threshold. Words with a score below the threshold are not considered. In the second step, a database search is performed to find the exact words to find the places for which there is a precise match. In the third stage, the pairwise matching continues by expanding the words in two directions so that the matching score does not fall below the threshold (the threshold is 22 for protein and 20 for DNA). Continuously matched segments without gaps are called high-scoring segment pairs (HSP). The final evaluation is then performed for the scores.

Ruan et al. [20] propose the MAQ algorithm, a concept of mapping quality, a program that quickly maps short queries to reference. The algorithm creates several hash tables to index the queries and then searches the genome in the hash tables to find the hits. It uses six hash tables corresponding to six seed space patterns for each paired-end by default, ensuring that there is a hit in a sequence with a maximum of two mismatches, returning a safe output to the user. According to Smit et al. [21], the mapping problem behaves like the approximate match of a pattern set in a text, in the sense that the match is first identified in the regions of the genome where K-mers are accurately matched because this type of match is much faster to identify than approximate match. Next, the approximate match between query and reference is evaluated in the areas around the matched K-mers.

Reference-based hash algorithms scan query sequences after indexing reference sequences and storing them in memory. The advantage of this method is that the hash table is created only once and can be used for all imported queries, but the maintenance of the hash table requires a lot of memory [15]. One of these algorithms is AGILE, which is presented by Misra et al. This algorithm suggests filtering many candidates during the search stage and preventing them from entering the mapping stage, thus reducing the number of candidates that enter the mapping stage according to the criteria. In this way, the work and time required to map the sequences will be reduced.

Choi et al. [5] use a combination of two hash indexes and suffix arrays to create a Hybrid Index to perform the alignment process. Query positions are then retrieved from this table and expanded and evaluated by finding candidate areas between the two imported sequences.

Ning et al. [16] use a technique other than Seed-and-extend, the SSAHA algorithm, in search and mapping stages. In this algorithm, non-overlapping subsections are used to store data in the hash table. This drastically reduces the accuracy of the output. In this method, the hash table is created in two passes. In the first pass, all possible subsequences of length K are created and stored in the table, and in the second pass, the positions of each of the K-mers in the table are extracted from the reference sequences and stored in the table. The positions of the query K-mers are then extracted from the reference table, and a hit list is generated for it. Finally, mapping is done from the hit list. Hessan et al. [22] parallelize the Blast algorithm using the MPI technique to split the reference database between more nodes using a greedy algorithm. Each Worker node performs a search and mapping operation on the data they have copied to their local disk and returns the result to the Master for integration.

Marek et al.[23] perform the Blast algorithm parallelization process in HPC supercomputers and clusters using thousands of processors. Work distribution and search management are done using a Java library called parallel computing in Java (PCJ). Mozafari et al. [24] proposed an Optic parallel processing architecture. In this method, the

query points that are present in the reference are extracted in parallel. A simple

algorithm is then used to find the edit distance between the extracted points and their location. The algorithm analyzes the correlation between them. Although Optic offers high-speed processing of alignment results, not every algorithm can be executed effectively with Optic. Stephen et al. [25] proposed an algorithm that can accept the gap in addition to accepting the mismatch. This algorithm performs process parallelization based on OpenMP and uses a mask to generate possible K-mers, according to which some bases are not included in the mapping. In other words, the match and mismatch between these bases will not affect the result of the mapping. They tried to increase the sensitivity with this technique so that the mapping based on matching was not completely accurate. In a study by Xueting et al. [26], a machine learning-based bit mapping method was proposed for mapping query sequences to reference. In this method, the hash functions are trained from the transcriptome law and generate sequential binary hash codes. In the next step, the query sequences are mapped to the transcriptomes based on their hash code.

In a study by Tony et al. [27], a high-performance parallel K-mer indexing and counting library used in distributed memory environments was developed. In this library, a set of simple and stable APIs with serial semantics and parallel implementation with the MPI technique has been designed.

In most of the algorithms discussed, special hardware systems or the addition of special software and algorithms are required to create parallelization capabilities, which themselves will be overheads. Furthermore, tools often use non-overlapping techniques to split reference sequences to reduce memory consumption. This causes only 1/k of the reference sequences to be placed in the hash table. In other words, not all K-mers in the reference sequence go to the search and mapping stage. So, it is very effective on the accuracy of output. A comparison of the above methods is given in Table 1.

TABLE 1. Comparison of alignment algorithms

Reference	Algorithm	Indel / SNP	Input type	Hash Type	Alignment technique
Altschul et al. [14]	Blast	Both	Short	Query	Hash-Base
Ruan et al. [20]	MAQ	NA	Short	Query	Hash-Base
Smite et al. [21]	RMAP	NA	Short	Query	Hash-Base
Misra et al. [15]	AGILE	Both	Long	Reference	Hash-Base
Choi et al. [5]	HIA	Both	Short	Reference	Suffix Array & Hash-Base
Ning et al. [16]	SSAHA	Both	Long	Reference	Hash-Base
Hesan et al. [22]	MPIBLAST	Both	Short	Query	Hash-Base
Nowicki et al. [23]	MPIBLAST	Both	Short	Query	Hash-Base
Mozafari et al. [24]	WOC	Indel	Short	Reference	Hash-Base
Stephen et al. [25]	SHRiMp	Both	Short	Query	Hash-Base
Xueting et al. [26]	Learning hash-table	Both	NA	Reference	Hash-Base
Tony et al. [27]	Kmerind	Indel	NA	Reference	Hash-Base
Seyed Ehsan et al	Proposed	Indel	Short/ Long	Reference	Hash-Base

This table compares different algorithms in terms of some features such as indel, SNP, input type, and hash type. Insertion–deletion mutations (indels) refer to insertion and/or deletion of nucleotides into genomic DNA and include events less than 1 kb in length. A single nucleotide polymorphism(SNP) is the replacement of one nucleotide with another nucleotide that occurs in the genome between individuals of a biological species or between a pair of chromosomes in an individual. NGS instruments generate data that contains a large number of very short DNA sequences called "reads" or queries that contain little information. Several tools for mapping short queries to a reference genome are provided. Which are very efficient for short queries but are inefficient and impractical against queries with more than 200 bp. In this way, many resequencing produces long queries. Therefore, tools are needed that can map long queries well and at high speed.

In the alignment process, different tools generally use two types of indexing algorithms, based on queries and reference genomes. A set of tools indexes the query sequence and stores it in memory; The genome sequence is then scanned against it. Although this method requires less memory But for each query sequence, re-indexing must be performed. Other tools index the genome sequence and store it in memory. This

method requires a lot of memory, but once the genome is indexed, all input query sequences can be aligned.

III. Problem Definition

In the following, definitions of query, reference sequences, hash table, K-mer, and how to display them in the proposed algorithm are displayed.

Definition 1

Reference and query sequences, which are shown as a set of (Ref, Q), are DNA strings. Queries are unknown sequences and the reference is the set of identified sequences in the database and the purpose of alignment is to identify the similarities between them and thus predict the structure of the query. So that:

$Ref= (Seq_1, Seq_2, ..., Seq_m)$, $m>=1$
$Q= (Seq_1, Seq_2, ..., Seq_n)$, $n>=1$

For the alignment process, proceed base-by-base along with Q from base 0 to base L-k, in which L is the length of Q and its position is extracted from the reference to check its similarity.

Definition 2

Each string of DNA is defined as a set (∑, S, n), where ∑ is a set of characters in the DNA string that includes the A, C, G, T. S is a combination of these characters that make up the sequence, and n is the length of the sequence. Then binary numbers are used instead of characters to reduce storage space. And each character is displayed with only two bits instead of eight. Therefore, each string can be encoded as an unsigned integer with two bits in each nucleotide[13, 16]:

$f(A)=(00)_2$ $f(C)=(01)_2$
$f(G)=(10)_2$ $f(T)=(11)_2$

Definition 3

K-mer is a substring of length k, which is a continuous sequence of DNA bases. In other words, K-mer is a set (w, k) and is represented by the following relation [16]:

K-mer: $w=b_1,b_2,....,b_k$

Where b represents each of the bases in the substrings. A sequence S of DNA that is n bases long will contain (n-k+1) overlapping K-mer. Each K-mer can be displayed with a 2k bit which is named the K-mer index. it is unique and created using Equation (1).

$$E(w) = \sum_{i=1}^{k} 4^{i-1} f(b_i) \qquad i=1, 2,...., k \quad (1)$$

Definition 4

The hash table in the proposed method is defined as a ternary (w, E (w), position), Where w is the desired K-mer, E (w) is its index and Position is an array of the K-mer positions in the reference sequence defined as (i, j) so that i is the sequence number and j represents the position of K-mer in the reference sequence.

Definition 5

After creating the hash table, all query sequence hits can be searched and extracted using the table[13,16]. For this purpose, it proceeds base-by-base from base zero to (L-k) along the input sequence, L is the length of the query. At base t, the list of r positions of the hits of the K-mer wt(Q) must be obtained. Therefore, the list of positions for each K-mer is extracted and stored in another table for the mapping process which is done using the relationship below[16].

$$List = (i_1, j_1)(i_2, j_2).....(i_r, j_r)$$

After this step, the list of hits is calculated using Equation 2 [16].

$$H = (i_1, j_{1-t}, j_1)(i_2, j_{2-t}, j_2).....(i_r, j_{r-t}, j_r) \quad (2)$$

Where t is the K-mer distance from the beginning of the query sequence. The hit list elements are index (i_r), shift (j_{r-t}), and offset (j_r), respectively.

IV. The Proposed Method

In this section, a brief definition of the proposed algorithm is provided. An overview of this algorithm can be seen in Figure 1. As mentioned earlier, the proposed algorithm aims to improve the storage process and increase speed compared to similar algorithms in addition to increasing accuracy so that no special hardware or software platform is required. So, the algorithm performs the alignment steps concurrently using the OpenMP library and benefits from other advantages of this technique such as simplicity of parallelization, scalability, high performance, portability of the written program, and no need for complex programming.

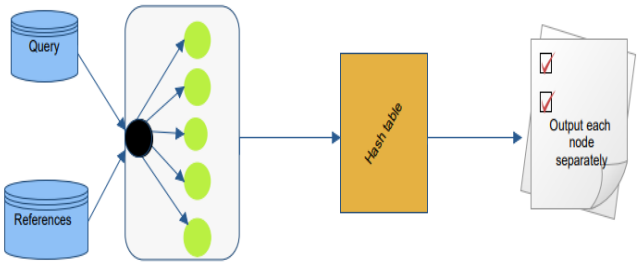

Fig. 1. The proposed framework

In this algorithm, the alignment process is done in several parts. In the first part, the reference sequences are received from the input and then compressed and distributed among the processors. Each processor divides its sequence into overlapping K-mers, extracts their positions from the sequence, and indexes it in the hash table. All processors then combine their output to create each of the rows in the reference hash table. These two steps are performed concurrently. Query sequences are then received from the input file and, after compression, distributed among the processors so that their overlapping K-mers are extracted by each processor. In the next step, each processor extracts the position of its query sequence K-mers from the shared hash table and performs the mapping operation of its sequence. They are then sorted, and the output is generated. Finally, the

list with the most matching positions between the two sequences can be presented as the best matching sequence between the query and the reference by each processor and displayed in the output. So, there is no need for nodes to return their results to the central node or combine the results. In the following, the details of each part will be provided separately.

A. Receiving input

Since DNA sequences are double-stranded, query and reference files can be placed in two separate files that must be combined in the preprocessing step. In this algorithm, a query file in the .fastq format is received and used. The reference file can also contain any number of sequences that are received in the. fasta format to be saved in the hash table in the next step. In the first step, the query and reference files are preprocessed. Input files sometimes contain characters other than the four main DNA characters that are not properly known and are indicated by N in the sequences. Some algorithms, such as SSAHA, randomly replace these unknown characters with one of the four main characters. However, in the proposed algorithm, unknown characters are removed to increase accuracy and prevent false events since close events will not be lost with limited mismatch and indel. Besides, since the use of the non-overlapping technique greatly reduces sensitivity and accuracy, this algorithm uses the overlapping technique in reference sequences in addition to query sequences to display more complete results. Using the overlapping technique requires more memory and increases the time required to create the table, but since the table is created only once and can be used for all query sequences, this does not increase the time.

B. Creating a hash table

As presented in Definition 4, the hash table has three parts w, E (w), and Position. In the SSAHA algorithm, two data structures, a list of positions and an array of pointers to the list, are used. 4k pointers are required since, in this method, all possible states of K-mers with length k are placed in the table. However, the proposed algorithm can create and complete a hash table with a single pass. In other words, it extracts the position of the K-mers in order of passing them and puts them in the position list by passing through the beginning of the reference sequences. In this step, the parallelization technique is used to create a hash table to improve alignment and use multi-processor capabilities. Thus, the sequences must be divided between the processors. If the number of processors and sequences are taken as n and m, respectively, approximately m/n sequences are assigned to each processor.

Algorithm 1. Create a Hash table in parallel

Input: A set of sequences in the Reference file
Output: A hash table with K-mers of reference sequences and their position

```
1.    #pragma omp for
2.   parallel for  i IN 0 TO mers_vector.size  do  //vectoer of K-mers
3.     for all mer in mers_vector[i] do
4.        temp.push(K-mer,seqno,position)  // K-mer properties
5.        index_mer(temp,K-mer)          //calcute index of K-mer
6.        If K-mer is found
7.        table.mers_vector.push(temp)
8.     endfor
9.    endfor
10.  for w  IN 0 TO table[0].size  do   //information of on Rows
11.    for  i IN 1 TO table.size  do
12.      for each K-mer in table[i][w].mers_vector
13.         table[0][w].mersvector.push(K-mer)
14.      endfor
15.    endfor
16.     Hash-table.push(table[0][w])  //Each time a row is completed
17.  Endfor
```

As shown in Algorithm 1, by moving over their sequences, each processor extracts the K-mer positions and their associated sequence numbers and places them in a list with the desired K-mer name. The mer index is also calculated to be used as the unique identifier for the K-mer. Finally, the output of the processors that are in lists LP_1, …, LP_n (the number of lists is equal to the number of processors used) are combined to create each row in the final hash table.

C. Searching for the query sequences

Query K-mer positions must be extracted from the hash table to search for a query sequence using the hash table. In other words, the processors extract the overlapping K-mers from the received query sequences, search for their position in the hash table, and record them in their table after extraction to map them. This step is done concurrently and without the data dependencies so that the speed is increased. Algorithm 2 shows the steps of the search process.

Algorithm 2: Query search using threads in parallel

Input: A set of query sequences and Hash table

Output: A table of query for map

```
1.    #omp num_threads set(maximum available)
2.   #progma omp parallel for shared(Hash-table ,map)
3.      if (K-mer in Hash-table ) then
4.          map.find(K-mer)
5.          Extract positions of K-mer from Hash-table
6.        for i IN 0 TO Hash-table
     [it.second].mers_vector.size()  do
7.            Extract seqno of K-mer from Hash-table
8.            Query-table.push(seqno,positions)
9.        endfor
10.     endif
11.   endfor
```

D. Mapping

In this step, the hit list is prepared based on definition 5 to achieve similarities between the two sequences. It is then sorted by the Quicksort algorithm so that it is sorted first by index, then by shift, and finally by position. Since the sequences are entered sequentially, their position is extracted by moving on the sequence, and they are relatively orderly, the process is done in O(nlogn) time. Because this algorithm uses the overlapping technique to create K-mers, the difference in the positions in which the event occurred is one. In other words, according to the hit list, the items that have the same index and shift and consecutive positions represent the matching part in two sequences. In this algorithm, we can make some allowance for insertions/deletions or indel by

permitting the shift between consecutive hits in a run to differ by a small number of bases. The maximum indel size is considered to be 2 to achieve the same matches by considering the accuracy and is not limited to the exact match.

According to the indel, the size of the shift can be up to 2 units different from its previous position. The gap can also be supported by linking closely matched parts. Finally, the sequence with the longest continuous match length can have the highest match with the query sequence.

V. Evaluation

The algorithm's implementation is evaluated on a machine with 24 core 3 GHz and 32 GB RAM specifications. In these tests, to evaluate and compare the proposed algorithm with the Blast and SSAHA algorithms, one data set has been used. data set consists of a reference file along with 1 million reference sequences with 120 MB size and a query file of 15 million sequences with 1.2 GB size based on table2.

A. Comparison of the proposed method with other methods using one processor.

In this section, different sizes of K-mer are considered in all three algorithms, i.e. SSAHA, Blast, and the proposed algorithm using one processor to compare memory and time.

TABLE 2. Dataset used in experiments

Input	Algorithm	Time (Sec)			
		K=10	**K=15**	**K=17**	**K=20**
DB	Blast	372000	125600	46520	14066
	SSAHA	27566	5157	8947	8982
	Propos	22052	41700	7260	7299

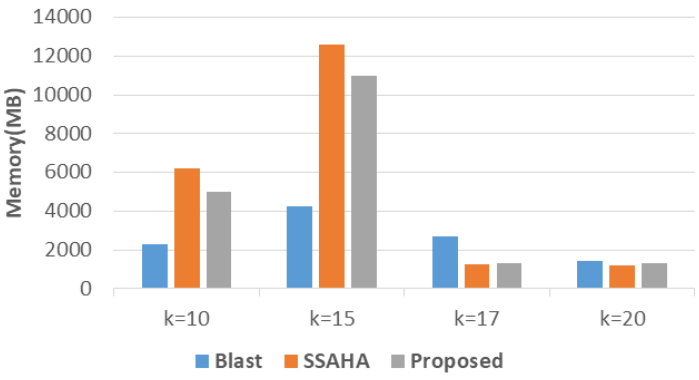

Fig. 2. Comparison of memory consumption for three algorithms

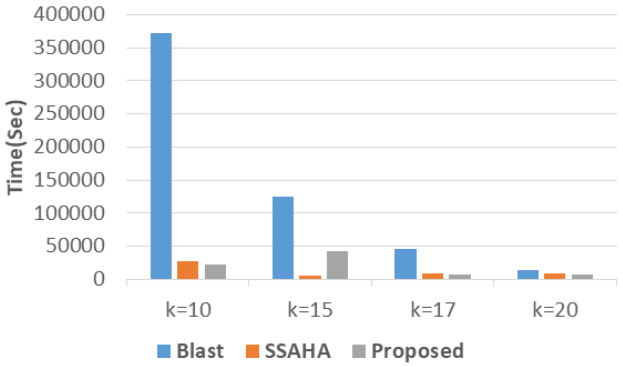

Figure 3: Comparison of execution time for three algorithms

As evident in Figure 2, the proposed algorithm uses the overlap technique to increase the accuracy of the output results in both query and reference sequences, which increases the amount of memory required; However, this algorithm has been able to manage the amount of memory with the techniques described in the proposed method so as not to significantly increase memory consumption. But as shown in Figure 3, it performs better in terms of runtime than the other two algorithms. Table 3,4 presents a summary of the test results.

TABLE 3. Comparison of algorithms (Based on Memory)

Input	Algorithm	Memory (MB)			
		K=10	**K=15**	**K=17**	**K=20**
DB	Blast	2283	4254	2677	1428
	SSAHA	6223	12594	1285	1200
	Propos	5025	10984	1322	1329

TABLE 4. Comparison of algorithms (Based on Time)

Dataset	**Filename**		**Sequence-number**	**Size**
DB	Query	SRR494099.fastq	15 million	1.2GB
	Ref	est_human.fasta	1 million	120MB

B. Comparison of the proposed algorithm with multiple processors

This section focuses on the extent of speed improvement using multiple processors instead of one processor which does not have a significant effect on memory. In conducting the tests, one, six, and twelve processors were used to run the proposed algorithm. Figure 4 shows the increase in speed for multiple processors.

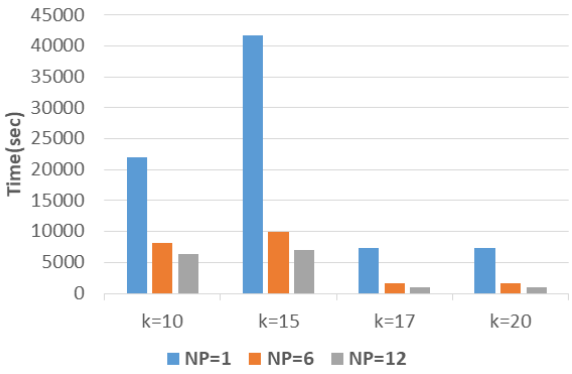

Fig. 4. Comparison of the proposed algorithm's execution time with multiple processors

According to Figure 4, using multiple processors will have a significant effect on the speed of the alignment. The difference between the execution time of the algorithm using one processor and several processors is clearly shown and proves that the speed of the alignment process is multiplied when several processors are used. Table 5 shows the time evaluation results in the proposed algorithm with a different number of processors.

TABLE 5. Time comparison in executing multiple processors

Input	Num Thread	Time (Sec)			
		K=10	**K=15**	**K=17**	**K=20**
DB	N=1	22052	41700	7260	7299
	N=6	8091	7896	1629	1641
	N=12	6287	4984	1004	967

VI. Conclusion

The number of DNA sequences is constantly increasing, and storing and searching for them within databases is time-consuming and requires a lot of memory. So, it seems necessary to use an efficient algorithm to search in a shorter time and optimal memory consumption. The goal of the SSAHA algorithm is to multiply the algorithm execution speed and search large databases faster than the Blast method and other methods based on Seed-and-Extend by removing the extension and evaluation used in most hash-based methods. The purpose of this study is to provide solutions to improve this algorithm to manage memory consumption in addition to increasing the accuracy and speed and to perform the indexing and mapping process using multiple processors with OpenMP library parallelization technique to take advantage of features such as simple implementation, portability, and ease of programming. One of the techniques used to increase the accuracy of the algorithm is the overlap of K-mers with a window of one length so that all K-mers of the sequences are extracted and entered in the alignment.

VII. References

[1] *Jones, N.C., P.A. Pevzner, and P. Pevzner, An introduction to bioinformatics algorithms. 2004: MIT press.*

[2] *Herve, D., et al., A perceptual hash function to store and retrieve large scale DNA sequences. arXiv preprint arXiv:1412.5517, 2014*

[3] *Needleman, S.B. and C.D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins. Journal of molecular biology, 1970. 48(3): p. 443-453.*

[4] *Wilbur, W.J. and D.J. Lipman, Rapid similarity searches of nucleic acid and protein data banks. Proceedings of the National Academy of Sciences, 1983. 80(3): p. 726-730.*

[5] *Choi, J., et al., HIA: a genome mapper using hybrid index-based sequence alignment. Algorithms for Molecular Biology, 2015. 10(1): p. 30.*

[6] *Li, H. and N. Homer, A survey of sequence alignment algorithms for next-generation sequencing. Briefings in bioinformatics, 2010. 11(5): p. 473-483.*

[7] *Li, H. and R. Durbin, Fast and accurate long-read alignment with Burrows-Wheeler transform. Bioinformatics, 2010. 26(5): p. 589-595.*

[8] *Chen, Y., T. Souaiaia, and T. Chen, PerM: efficient mapping of short sequencing reads with periodic full sensitive spaced seeds. Bioinformatics, 2009. 25(19): p. 2514-2521.*

[9] *Liu, C.-M., et al., SOAP3: ultra-fast GPU-based parallel alignment tool for short reads. Bioinformatics, 2012. 28(6): p. 878-879.*

[10] *Li, H. and R. Durbin, Fast and accurate short read alignment with Burrows-Wheeler transform. bioinformatics, 2009. 25(14): p. 1754-1760.*

[11] *Langmead, B. and S.L. Salzberg, Fast gapped-read alignment with Bowtie 2. Nature methods, 2012. 9(4): p. 357.*

[12] *Wu, T.D., Bitpacking techniques for indexing genomes: I. Hash tables. Algorithms Mol Biol, 2016. 11: p. 5.*

[13] *Toh, S.-H., H.-J. Lee, and K.-H. Do. Basic sequence search by hashing algorithm in DNA sequence databases. in Advanced Communication Technology, 2009. ICACT 2009. 11th International Conference on. 2009. IEEE.*

[14] *Altschul, S.F., et al., Basic local alignment search tool. Journal of molecular biology, 1990. 215(3): p. 403-410.*

[15] *Misra, S., et al., Anatomy of a hash-based long read sequence mapping algorithm for next-generation DNA sequencing. Bioinformatics, 2010. 27(2): p. 189-195.*

[16] Ning, Z., A.J. Cox, and J.C. Mullikin, SSAHA: a fast search method for large DNA databases. Genome research, 2001. 11(10): p. 1725-1729.

[17] Sedlazeck, F.J., P. Rescheneder, and A. Von Haeseler, NextGenMap: fast and accurate read mapping in highly polymorphic genomes. Bioinformatics, 2013. 29(21): p. 2790-2791.

[18] Ma, B., J. Tromp, and M. Li, PatternHunter: faster and more sensitive homology search. Bioinformatics, 2002. 18(3): p. 440-445.

[19] Canzar, S. and S.L. Salzberg, Short read mapping: An algorithmic tour. Proceedings of the IEEE, 2017. 105(3): p. 436-458.

[20] Li, H., J. Ruan, and R. Durbin, Mapping short DNA sequencing reads and calling variants using mapping quality scores. Genome research, 2008. 18(11): p. 1851-1858.

[21] Smith, A.D., et al., Updates to the RMAP short-read mapping software. Bioinformatics, 2009. 25(21): p. 2841-2842.

[22] Lin, H., et al. Efficient data access for parallel BLAST. in 19th IEEE International Parallel and Distributed Processing Symposium. 2005. IEEE.

[23] Nowicki, M., D. Bzhalava, and P. BałA, Massively parallel implementation of sequence alignment with basic local alignment search tool using parallel computing in java library. Journal of Computational Biology, 2018. 25(8): p. 871-881.

[24] Mozafari, F., et al., Speeding up DNA sequence alignment by the optical correlator. Optics & Laser Technology, 2018. 108: p. 124-135.

[25] Rumble, S.M., et al., SHRiMP: accurate mapping of short color-space reads. PLoS Comput Biol, 2009. 5(5): p. e1000386.

[26] Yu, X. and X. Liu, Mapping RNA-seq reads to transcriptomes efficiently based on learning to a hash method. Computers

[27] Pan, T., et al., Kmerind: A Flexible Parallel Library for K-mer Indexing of Biological Sequences on Distributed Memory Systems. IEEE/ACM Trans Comput Biol Bioinform, 2019. 16(4): p. 1117-1131.

How to cite: S.E.Parsaeian, N.Aghaee-Maybodi. Local Sequence Alignment with a Parallel Hash based Model, Journal of Distributed Computing and Systems(JDCS), Vol 4, Issue 2, Pages 65-72, 2021.