Comparative Analysis of Computationally Accelerated NGS Alignment

Ada Chaeli van der Zijp-Tan
University of South Alabama

THE UNIVERSITY OF SOUTH ALABAMA
COLLEGE OF ALLIED HEALTH PROFESSIONS

COMPARATIVE ANALYSIS OF COMPUTATIONALLY ACCELERATED NGS ALIGNMENT

BY

Ada Chaeli van der Zijp-Tan

A Thesis

Submitted to the Honors College of the
University of South Alabama
in partial fulfillment of the
requirements for the degree of

Bachelor of Science

in

Biomedical Sciences

May 2021

Approved: Date:

[illegible] 05/04/2021

Chair of Thesis Committee: Dr. Jingshan Huang

Robin Mockett

Digitally signed by Robin Mockett
DN: cn=Robin Mockett, o=Univewrsity of South Alabama, ou=College of Allied Health Professions, email=mockett@southalabama.edu, c=US
Date: 2021.05.12 19:50:00 -05'00'

Committee Member: Dr. Robin J. Mockett

[illegible] 05/04/2021

Committee Member: Dr. Terrence J. Ravine

Dean of the Honors College: Dr. Kathy Cook

COMPARATIVE ANALYSIS OF COMPUTATIONALLY ACCELERATED NGS ALIGNMENT

A Thesis

Submitted to the Honors College of the
University of South Alabama
in partial fulfillment of the
requirements for the degree of

Bachelor of Science

in

Biomedical Sciences

by
Ada Chaeli van der Zijp-Tan
May 2021

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

van der Zijp-Tan, Ada Chaeli, B. S., University of South Alabama, May 2021. Comparative Analysis of Computationally Accelerated NGS Alignment. Chair of Committee: Jingshan, Huang, Ph.D.

The Smith-Waterman algorithm is the basis of most current sequence alignment technology, which can be used to identify similarities between sequences for cancer detection and treatment because it provides researchers with potential targets for early diagnosis and personalized treatment. The growing number of DNA and RNA sequences available to analyze necessitates faster alignment processes than are possible with current iterations of the Smith-Waterman (S-W) algorithm. This project aimed to identify the most effective and efficient methods for accelerating the S-W algorithm by investigating recent advances in sequence alignment. Out of a total of 22 articles considered in this project, 17 articles had to be excluded from the study due to lack of standardization of data reporting. Only one study by Chen *et al.* obtained in this project contained enough information to compare accuracy and alignment speed. When accuracy was excluded from the criteria, five studies contained enough information to rank their efficiency. The study conducted by Rucci *et al.* was the fastest at 268.83 Giga Cell Updates Per Second (GCUPS), and the method by Pérez-Serrano *et al.* came close at 229.93 GCUPS while testing larger sequences. It was determined that reporting standards in this field are not sufficient, and the study by Chen *et al.* should set a benchmark for future reporting.

CHAPTER I

INTRODUCTION

Cancer is a disease categorized by uncontrollable division of abnormal cells (Huang, Alvarez, Hu, & Cheng, 2013). Cancer is, in part, the result of genetic mutations. Cancer often develops when cell growth control mechanisms in our deoxyribonucleic acid (DNA) are damaged. Current oncology research is invested in identifying genetic sequences associated with various types of cancer. Non-coding sequences, or non-coding ribonucleic acids (ncRNAs), contribute to the regulation of transcription and translation to protein from DNA. Although the exact functions of ncRNAs are unknown, microRNAs (miRNAs), a subset of ncRNAs, have recently been discovered to have some functions involving tumor suppression (Huang *et al.*, 2013). Small nucleolar RNAs (snoRNAs), which were previously thought to have no correlation to cancer, are another class of ncRNAs that have recently been discovered to be genetic markers in cancer (Chow & Chen, 2018). The sequences and precise roles of many snoRNAs and even more recently discovered sno-derived RNAs (sdRNAs) are still being identified.

Cancer research has been greatly improved by the Human Genome Project (HGP), which revolutionized how we investigate human biology. Next Generation Sequencing (NGS) techniques are part of the once hypothetical revolutionary advances in biomedical research and clinical practice (Moraes & Góes, 2016). Improvements in DNA sequencing technology have increased the efficiency of genetic investigation. Both technologic and analysis approaches have improved to better link genetic events to disease. Cancer is of particular interest due to the strong link between alterations in genetic code and various types of cancer. For example, massively parallel sequencing (MPS) allows thousands of genetic patterns to be identified from tens of different tumor types (Tucker, Marra, & Friedman, 2012). This resulted in new pattern recognition of genetic polymorphisms, some of which have been linked to increased risk of cancer (Han, Ding, & Kyung, 2015).

Most of the challenges derived from DNA sequencing are due to the growing size of biological databases. Supercomputers are required to align DNA and RNA sequences, which can be costly and not easily accessible. The sheer size of genetic databases creates additional complexity in alignment computations. Although NGS techniques have led to many developments in cancer genomics, and by extension precision cancer treatments, the diversity of changes in the cancer genome and recent growth in genetic database size has created pressure to develop more efficient versions of NGS alignment techniques (Behjati & Tarpey, 2013).

Bioinformatics is a rising interdisciplinary field that joins computer science and biology to process biological data quickly for a greater understanding of an organism's genetic makeup. Homologous genetic subsequences can be identified using sequence analysis methods, which utilize NGS technology. NGS alignment is a sequence analysis method necessary for analysis of DNA or RNA sequences (Nakagawa & Fujita, 2018). Cancer genome research is primarily performed using NGS, which is becoming increasingly more common and sophisticated. Computational analysis of biological data using NGS has led to major breakthroughs in discovering cancerous mutations within certain regions of the human genome. NGS alignment allows us to analyze cancer genomes significantly faster and more accurately than previously used methods (Meldrum, Doyle, & Tothill, 2011).

Bioinformatics research requires lengthy algorithms to process enormous data sets. Most modern genomic sequencing via NGS alignment is built around the Needleman-Wunsch (N-W) algorithm and Smith-Waterman (S-W) algorithm, which maintain high levels of accuracy at the cost of requiring time-consuming computations (Hendrix, 2019; Janes, 2005). Both are dynamic programming algorithms used to compare biological sequences. Dynamic programming is a computer programming technique used to solve complex problems by dividing them into several subproblems. The N-W and S-W algorithms are database search algorithms that find the best alignment between two sequences by comparing the nucleotides between two sequences and assigning positive scores for nucleotide matches and negative scores for mismatches or gaps. The N-W and S-W algorithms therefore identify optimal alignments between sequences by splitting the alignment procedure into several subproblems (Hendrix, 2019).

The S-W algorithm performs local biological alignment, as opposed to global alignment performed by the N-W algorithm (Janes, 2005). Global alignment is a type of alignment that attempts to align every element in a sequence with every element of the other sequence. Global alignment is useful for identifying how similar two sequences are to each other. Local alignment algorithms compare every element in a sequence with every element of the other sequence to identify regions of similarity between larger sequences. Local alignment algorithms are more advantageous when comparing different sequences that are suspected to have regions of similarities, which is often the case when comparing the known sequence of a tumor with a sequence from a patient. Since the S-W algorithm is the most used sequence alignment algorithm due to its comparatively high accuracy, many research efforts have gone toward accelerating S-W algorithm-based tools. Although the S-W algorithm is the most accurate local alignment algorithm, the number of computations required to identify local alignments increases with the size of the sequences. Some accelerations of the S-W algorithm have been tested, but research on NGS alignment is still being developed (Janes, 2005).

1.1 The Smith-Waterman Algorithm

The S-W algorithm determines the optimal alignment between two sequences of nucleic acids by systematically comparing the nucleotides of each sequence against each other (Janes, 2005; Smith & Waterman, 1981). The S-W algorithm follows three basic steps: initialization, matrix filling, and trace back. First, in the initialization step, two sequences, sequence A and sequence B, are arranged to label the first column and row of a matrix (Fig. 1). Then, all elements of the following row and column are set, or initialized, with zeros. The size of this matrix, not including the row and column containing the sequences, will be denoted by $(m + 1) \times (n + 1)$, where m is the number of nucleotides in sequence A and n is the number nucleotides in sequence B.

Sequence A = $a_1 a_2 \dots a_m$

Sequence B = $b_1 b_2 \dots b_n$

$$\text{Sequence A}$$

$$\text{Sequence B} \quad \begin{array}{c|c|c|c|c|c|}
 & a_1 & a_2 & \dots & a_m & \\
\hline
 & 0 & 0 & 0 & \dots & 0 \\
\hline
b_1 & 0 & & & & \\
\hline
b_2 & 0 & & & & \\
\hline
\vdots & \vdots & & & & \\
\hline
b_n & 0 & & & & \\
\hline
\end{array}$$

Fig. 1 Initialization Step

A scoring scheme (Fig. 2) is also established during this step to assign a score to each pair of nucleotides for matches, mismatches, and gaps. $sim(a_i,b_j)$ represents the similarity score between two nucleotides that are being compared, a_i and b_j. a_i represents the nucleotide in the ith position on sequence A, while b_j represents the nucleotide in the jth position in sequence B. The similarity score is positive. In the developed scoring scheme, the penalty scores for a mismatch and a gap must both be negative (Smith & Waterman, 1981). In the example scoring scheme (Fig. 2) the gap penalty g is -2, similarity score for a_i and b_j is +5 if a_i and b_j match, and the penalty is -3 if a_i and b_j are a mismatch.

$$sim(a_i, b_j) = \begin{cases} +5, & if\ a_i = b_j \\ -3, & if\ a_i \neq b_j \end{cases}$$
$$g \;=\; -2$$

Fig. 2 Scoring Scheme Example

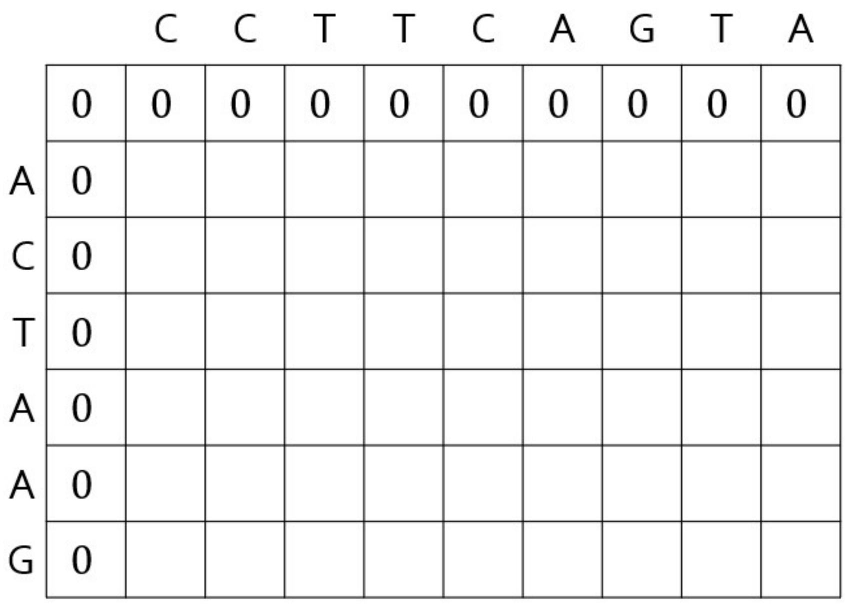
The second step, matrix filling, fills the cells of the matrix with scores determined by comparing each nucleotide in sequence A with the nucleotides in sequence B. The squares of the matrix are calculated by comparing the values of three surrounding squares that are located directly above, left adjacent, and the immediate upper left diagonal using the formula:

$$S_{i,j} = max \begin{cases} S_{i-1,j-1} + sim(a_i, b_j) \\ S_{i-1,j} + g \\ S_{i,j-1} + g \\ 0 \end{cases}$$

where $S_{i,j}$ is the maximum similarity score between two segments ending in a_i and b_j, $S_{i-1,j-1} + sim(a_i, b_j)$ is the score for aligning nucleotides a_i and b_j , $S_{i-1,j} + g$ is the score for adding a gap in the segment from sequence B, and $S_{i,j-1} + g$ the score for adding a gap in the segment from sequence A. This occurs such that match values and mismatch values are only added on the diagonal, while the gap penalty is only added along horizontal and vertical paths. The inclusion of 0 as the fourth element is one of the key differences between the S-W algorithm and the N-W algorithm. The inclusion of 0 allows the S-W algorithm to ignore negative alignment scores. This gives the S-W algorithm the ability to identify local alignments at any position, even when alignments are located near dissimilar segments (Janes, 2005; Smith & Waterman, 1981).

Filling the matrix is a crucial step for the S-W algorithm. The first row and column of the matrix for Sequence A = ‘CCTTCAGTA’ and Sequence B = ‘ACTAAG’ are filled or initialized with 0 (Fig. 3). The starting point for filling the matrix is the first empty square in the upper left corner of the matrix. The first nucleotides in sequences A and B in the example are ‘C’ and ‘A’ respectively (Fig. 3), and the similarity score is added to the neighboring diagonal value, 0. Using the scoring scheme (Fig. 2), the values compared by the algorithm are as follows:

$$S_{i,j} = max \begin{cases} S_{i-1,j-1} + s(a_i, b_j) \\ S_{i-1,j} + g \\ S_{i,j-1} + g \\ 0 \end{cases} \Rightarrow S_{1,1} = max \begin{cases} S_{0,0} + s(a_1, b_1) \\ S_{0,1} + g \\ S_{1,0} + g \\ 0 \end{cases} = max \begin{cases} 0 + (-3) \\ 0 + (-2) \\ 0 + (-2) \\ 0 \end{cases}$$



		C	C	T	T	C	A	G	T	A
	0	0	0	0	0	0	0	0	0	0
A	0									
C	0									
T	0									
A	0									
A	0									
G	0									

Fig. 3 Matrix Initialization Example

The gap penalty is subtracted from the scores in the square located above and the square located to the left of the square corresponding to the two nucleotides being compared. It is necessary to keep track which square each of the three calculated values originates from for later use in the traceback step. The origin of the calculated values for the box corresponding to the first ‘C’ and ‘A’ are shown below (Fig. 4).

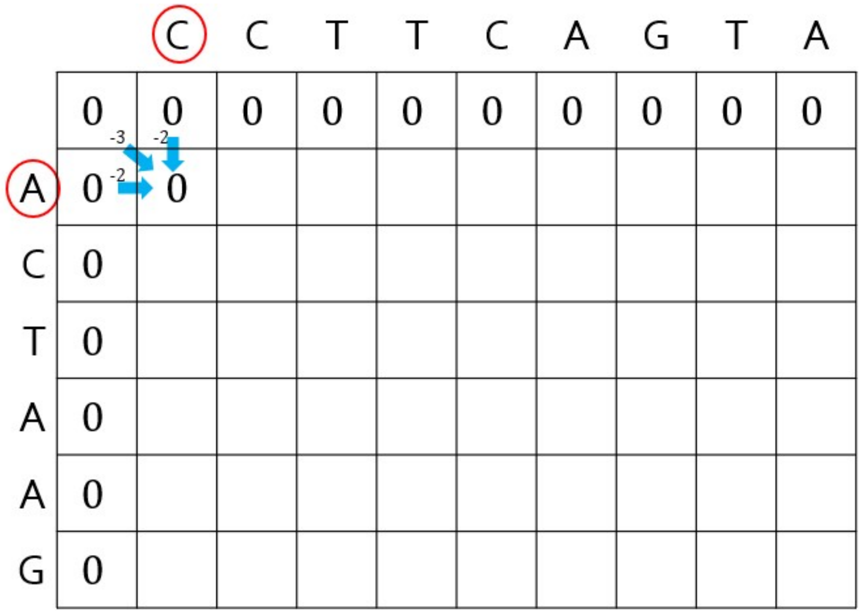


Fig. 4 Scoring by Comparing First Nucleotides of Sequence A and Sequence B

If none of the three calculated values are positive, then the similarity score for the segment is set to 0, otherwise the largest out of the three calculated values is chosen. It is also worth noting the matrix can be filled in any direction as long as there are three surrounding values, calculated from the adjacent left box, from the adjacent box above, and from the upper left diagonal, to compare (Janes, 2005; Smith & Waterman, 1981). Many calculations are necessary to completely fill the matrix for the two example sequences ‘CCTTCAGTA’ and ‘ACTAAG’ (Fig. 5). When the matrix is filled out, paths with positive values are considered (Fig. 6).

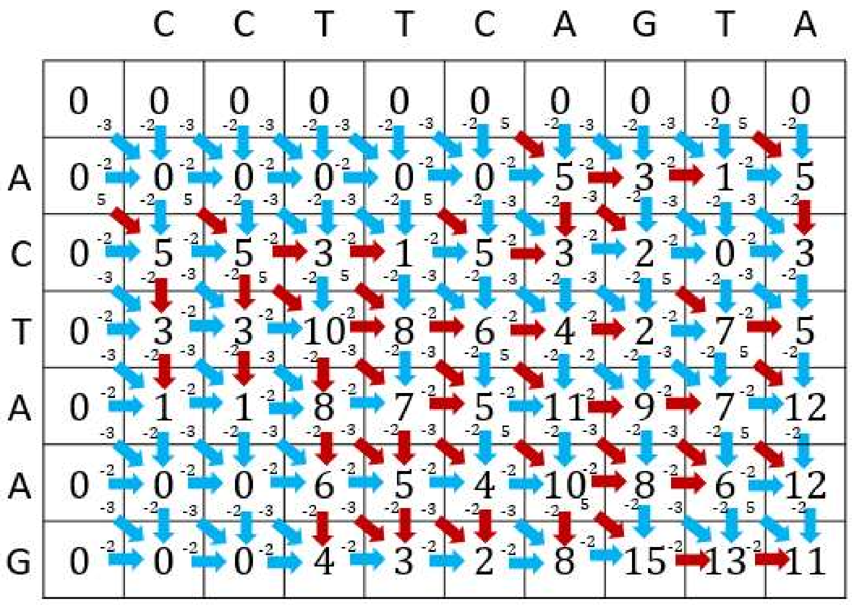


Fig. 5 Filled Matrix Example Tracking All Calculated Scores (Blue Arrows) and Maximum Positive Scores (Red Arrows)

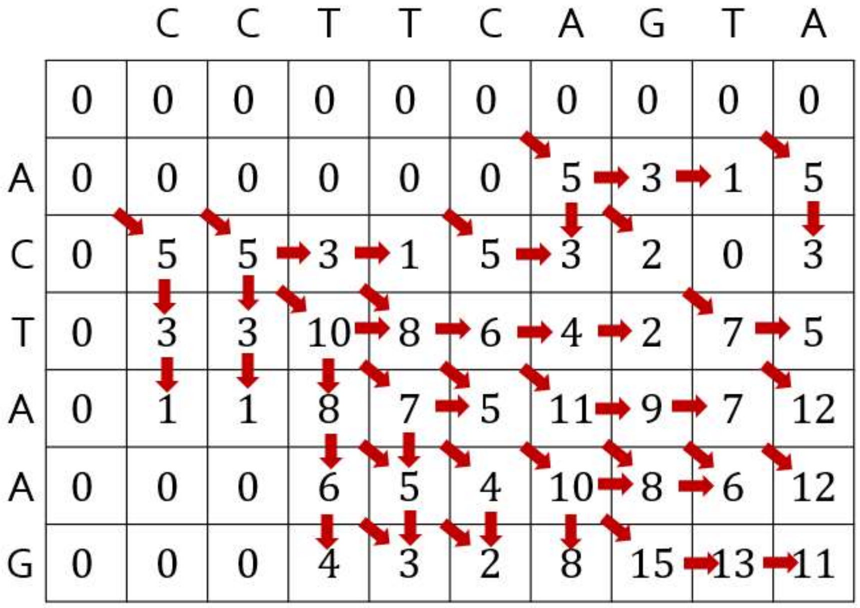
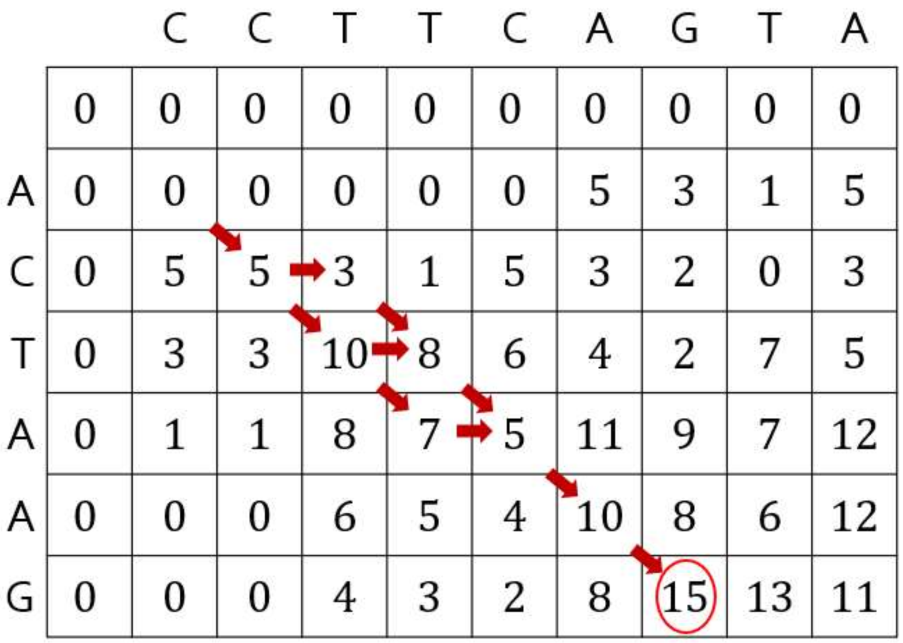


Fig. 6 Filled Matrix Example Showing Origin of Positive Max Scores

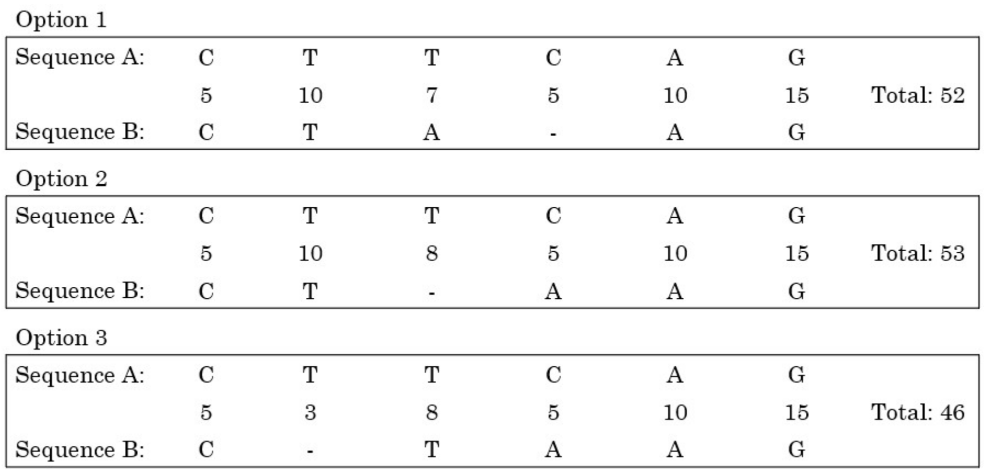
In the traceback step, the position with the highest score is first located. It is possible for there to be more than one alignment with the same highest score (Fig. 7). In such cases, there may be more than one possible optimal alignment. Then, the preceding box that was used to calculate the highest score is identified, which will either be located adjacent to the left, above, or the upper left diagonal (Fig. 7). The trace back direction for each box depends on which preceding box was used to calculate its score. This process of tracing back through the matrix will continue to the next preceding box until a score of 0 is found (Janes, 2005; Smith & Waterman, 1981).



		C	C	T	T	C	A	G	T	A
	0	0	0	0	0	0	0	0	0	0
A	0	0	0	0	0	0	5	3	1	5
C	0	5	5	3	1	5	3	2	0	3
T	0	3	3	10	8	6	4	2	7	5
A	0	1	1	8	7	5	11	9	7	12
A	0	0	0	6	5	4	10	8	6	12
G	0	0	0	4	3	2	8	15	13	11

Fig. 7 Possible Optimal Alignment Paths on Filled Matrix Example

In the case that there is a tie when tracing back from a box, all tied paths may be considered (Fig. 8). The path with the largest *sum* of alignment scores is considered to be the path that corresponds to the optimal alignment. The optimal local alignment is the sequence that corresponds to these boxes, where tracing back vertically or horizontally will result in gaps in sequence A and B, respectively. The sequence that is associated with these alignment scores is printed in the reverse order of the traceback. Additionally, the value of the highest score in the matrix provides some indication for how optimal the alignment is: higher scores indicate more optimal alignment (Janes, 2005; Smith & Waterman, 1981).



Option 1

Sequence A:	C	T	T	C	A	G	
	5	10	7	5	10	15	Total: 52
Sequence B:	C	T	A	-	A	G	

Option 2

Sequence A:	C	T	T	C	A	G	
	5	10	8	5	10	15	Total: 53
Sequence B:	C	T	-	A	A	G	

Option 3

Sequence A:	C	T	T	C	A	G	
	5	3	8	5	10	15	Total: 46
Sequence B:	C	-	T	A	A	G	

Fig. 8 Possible Optimal Local Alignments

1.2 Methods for Accelerated S-W Alignment

There are a variety of approaches used to increase the efficiency of the S-W algorithm because it requires many time-consuming computations and decisions. Many methods for accelerating the S-W algorithm have been proposed, but the most effective direction has not yet been determined by considering alignment speed, accuracy, development costs, ease of use, and hardware costs relative to other acceleration methods. Many factors affect the performance of methods for accelerating the S-W algorithm. The number of lines of code used in the development of alignment tools can affect both the development time and alignment efficiency. More lines of code often correspond to longer alignment run times and increased development efforts. Many software implementations of an accelerated S-W algorithm measure their efficiency by how many lines of code are required to develop their related alignment tools (Nakagawa & Fujita, 2018). However, one caveat to software-based sequence alignment tools is that many have limited capacity for handling massive numbers of long sequences (Vineetha, Biji, & Nair, 2019).

Some of the software-based methods include unique methods for following the logic of the S-W algorithm. Parallelization is a computing method where a program is used to solve multiple problems in parallel rather than one at a time. Some methods utilize enhanced parallelization through the programming language Spark. These methods are often referred to as Spark parallelization, which offers enhanced parallelization of tasks. The design of Spark allows tools to be more efficient and increase the speed of certain applications. The Spark-OSW algorithm introduces a new method for improving the efficiency of the S-W algorithm by utilizing Spark parallelization (Liu, Li, & Gao, 2018). Another method utilizes the Residue Number System (RNS). RNS enhances the acceleration of tools that utilize addition, subtraction, and multiplication through the use of modular arithmetic, which is an integer-based system of arithmetic (Kehinde Bello & Alagbe Gbolagade, 2018).

There are also three software accelerated versions of the Burrows-Wheeler Aligner-Maximal Exact Match (BWA-MEM) genomic mapping tool that implement an optimized S-W algorithm. BWA-MEM-CUDA, BWA-MEM-OpenCL, and BWA-MEM-VHDL are designed using Compute Unified Device Architecture (CUDA), Open

Computing Language (OpenCL), and Very High-Speed Integrated Circuit Hardware Description Language (VHSIC-HDL, VHDL) computing platforms, respectively. CUDA and OpenCL are different programming models that are useful for parallel computing. CUDA is specific for programming GPUs developed by NVIDIA, while OpenCL is an open platform can be used to program devices from other vendors. VHDL is a computing language that is used in circuit design, which is utilized in programming Field-programmable gate arrays (FPGAs) (Houtgast, Sima, Bertels, & Al-Ars, 2018).

Many hardware-based methods for accelerating the S-W algorithm are focused on running the algorithm on hardware that specializes in parallelization. FPGAs consist of an array of programmable logic blocks that can be wired together in different configurations. This circuit-based architecture of FPGAs allows for massive parallelism and accelerated performance when compared to general-purpose processors (Salamat & Rosing, 2020). Other hardware-based accelerations of the S-W algorithm utilize the computing power of the graphics processing unit (GPU) and central processing unit (CPU), which each specialize in different types of calculations (Barnes, 2020). FPGAs are an alternative to the combined work of GPU and CPU. The combined computing ability of GPUs and CPUs allows for more complex problem solving when compared to FPGAs. The GPU and CPU generally work together to complete various computing tasks. GPUs in particular were originally designed for the purpose of handling images but have since been utilized for accelerating calculations with large amounts of data. GPUs excel at performing parallel tasks and are therefore often used for scientific computation. This ability of GPUs for massive parallelization provides much utility for S-W acceleration. However, the range of tasks the GPU can perform is limited compared to

CPUs. CPUs have limited ability to perform parallel tasks but can perform a wide variety of tasks such as solving complex problems and coordinating computing tasks. Therefore, GPUs often must work in tandem with CPUs to complete tasks (Palacios & Triska, 2011).

Many hardware-based methods for accelerating the S-W algorithm are focused on running the algorithm on improved and specialized hardware. Hardware-based methods, therefore, often involve using state-of-the-art devices to improve the speed. For example, integration of the S-W algorithm utilizing NVIDIA GPUs in a study by Ligowski and Rudnicki (2009) has been shown to increase efficiency of alignment programs. Some alignment tools must be run on specialized hardware, some of which may be difficult to obtain. Thus, two key elements to consider with hardware-based accelerations are the availability and the cost of the hardware itself. Hardware with higher specifications is often more costly and more difficult to obtain. There are other creative methods for accelerated hardware that are more easily available (Gálvez *et al.*, 2016). For example, Accelerating Smith-Waterman (ASW) increases the efficiency of the S-W algorithm by integrating the CPU and GPU to share the same memory. The chip integrating the CPU and GPU is called the Accelerated Processing Unit (APU). This simultaneous computation on CPU and GPU makes alignment efficient by subdividing the workload and executing commands concurrently. This model is effective for frequent data exchange because it eliminates the need for Peripheral Component Interconnect express (PCI-e) bus, which is responsible for connecting these components. The APU can therefore be more efficient at certain tasks by bypassing the need for communication between the memories of the GPU and CPU, which would normally be necessary in a system where the two processors are separated (Zou *et al.*, 2019). Other methods for operating on hardware with lower specifications utilize both software and hardware to accelerate the algorithm, as is the case with the VHDL design by Hakim, Kashtwari, Tiwari, and Sharma (2019) that utilizes improved FPGAs. This unique approach that utilizes both software and hardware acceleration has yet to gain popularity in bioinformatics. However, combined methods are still limited by their hardware's ability to handle calculations.

Currently the most outstanding issue in sequence alignment is the demand for acceleration due to the sheer volume of biological databases. The application of these accelerated alignment methods to cancer research presents the possibility of potential breakthroughs in the discovery of new ncRNAs and diagnostic technology. So far over 50,000 cancer genomes have been analyzed by NGS methods (Huang *et al.*, 2013). This number will continue to grow as sequence alignment is further accelerated. The S-W algorithm, which is the basis for all progressive local sequence alignment methods, requires many complex and tedious calculations. Several potential solutions have been developed that implement various forms of hardware and software accelerations. These studies present boundless potential in accelerated sequence alignment; however, the most effective direction has not yet been determined by considering alignment speed, accuracy, development costs, ease of use, and hardware costs relative to other acceleration methods. Although there are numerous methods and tools for accelerating S-W-based sequence alignment, there is currently no agreement on an optimal approach. This project aims to analyze current accelerations of S-W-based NGS alignment to determine which methods are most efficient.

CHAPTER II

HYPOTHESIS AND AIMS

The purpose of this project was to determine which methods for accelerating the Smith-Waterman algorithm are the fastest and which are the most accurate. The motivation behind finding methods to accelerate the Smith-Waterman algorithm is that identifying the optimal alignment between sequences requires many calculations and can become very time consuming. The number of calculations is proportional to the sizes of the sequences aligned, which can be incredibly large in the case of biological sequences. Several methods have been proposed to accelerate the speed of Smith-Waterman-based NGS alignment, but there is currently no universally agreed upon best method for accelerating the Smith-Waterman algorithm.

To determine viable solutions for satisfying the need for faster alignment, it was hypothesized that the most effective and most efficient methods for accelerating the S-W algorithm could be identified by investigating recent advances in S-W-based alignment methods. By analyzing recent advances in the efficiency of S-W algorithm, this project hoped to determine which alignment methods are most viable for quickly identifying homology between sequences.

CHAPTER III

MATERIALS AND METHODS

3.1 Initial Search

In this analysis, published experiments for accelerating sequence alignment tools that utilize the S-W algorithm were examined. The publications were obtained from PubMed, Google Scholar, and Researcher using a predefined search method, as follows. The initial search considered all experiments published after the initial proposal of the S-W algorithm in 1981 until 2021. These academic search engines were searched using the following key phrases: “accelerated NGS alignment”; “Smith-Waterman acceleration”; “accelerated pairwise sequence alignment”; “accelerated local alignment” and various combinations of these phrases. The search was also limited to studies that are written in English.

3.2 Inclusion Criteria

Both software and hardware-based accelerations of the S-W algorithm were considered. Inclusion of studies was restricted by several criteria: (i) the experiments used the Smith-Waterman algorithm as opposed to other alignment algorithms, (ii) the tools used in this study performed alignment for DNA sequences and RNA sequences rather than protein sequences, (iii) the accelerated tools must report the sizes of the sequences tested, accuracy of the tool, and alignment speed, (iv) alignment speed must be reported using cell updates per second (CUPS), which is a common measure for alignment performance, to be considered for the final comparison. In addition, (v) each acceleration method compared must also report the specific type of hardware used in the study.

3.3 Data Synthesis

The utility of each acceleration method was analyzed by collecting the data from the article corresponding to each accelerated tool and examining its effectiveness in relation to other acceleration methods. A table was built detailing the type of acceleration, name of the accelerated tool, sizes of the two sequences being aligned, size of the sequence files, alignment speed, total run-time, accuracy, development time, the number of lines of code (for software-based methods), hardware specifications, and which database search engine the study was obtained from, for each acceleration method.

After building the table detailing the data from each study on accelerated alignment, the tools were ranked by their efficiency and effectiveness. Alignment speed served to measure the efficiency for each alignment tool. Accuracy of alignment by each tool was used to measure their effectiveness. Alignment speed and accuracy were weighted highest and given equal weight as they are the primary factors for sequence alignment.

CHAPTER IV

RESULTS

A total of 39 articles pertaining to NGS sequence alignment were obtained from the initial search (Appendix A). Of these 39 articles, four were excluded for using alignment methods besides the S-W algorithm, six were excluded for using the S-W algorithm on protein sequences as opposed to DNA or RNA sequences, and seven were excluded for using the S-W algorithm in multiple sequence alignment (Fig. 9). DNA and RNA alignment are effectively equivalent because they are both made up of four base pairs, while protein sequences are made up of several amino acids. Out of the remaining 22 studies, ten studies reported total run time without explicitly reporting speed or sequence lengths (Table 1).

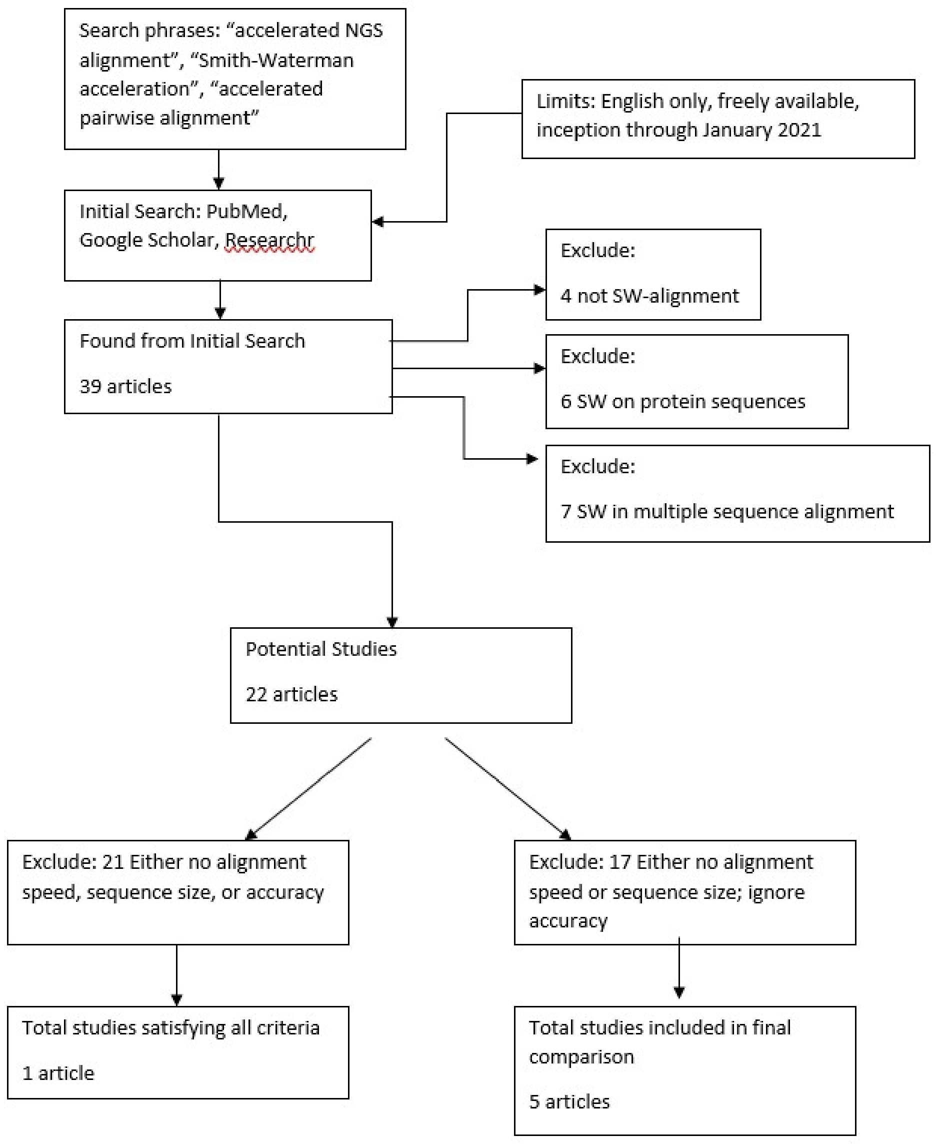

Search phrases: “accelerated NGS alignment”, “Smith-Waterman acceleration”, “accelerated pairwise alignment”
Limits: English only, freely available, inception through January 2021
Initial Search: PubMed, Google Scholar, Researchr
Exclude: 4 not SW-alignment
Found from Initial Search 39 articles
Exclude: 6 SW on protein sequences
Exclude: 7 SW in multiple sequence alignment
Potential Studies 22 articles
Exclude: 21 Either no alignment speed, sequence size, or accuracy
Exclude: 17 Either no alignment speed or sequence size; ignore accuracy
Total studies satisfying all criteria 1 article
Total studies included in final comparison 5 articles

Fig. 9 Search Diagram

Only three out of the 22 studies reported the accuracy of their experimental methods. Two (Liu *et al.*, 2018; Vineetha *et al.*, 2019) of these studies had incomplete reporting for alignment speed and the sizes of the sequences tested in the studies (Table 1). Among the three studies that reported their accuracy, Liu *et al.*, 2018 reported the highest accuracy at 100%, but their study failed to explicitly report alignment speed (Table 1). The study by Chen *et al.* (2021) reported 99% accuracy using their acceleration of the Smith-Waterman algorithm. Only one study (Chen *et al.*, 2021) fully met the final set of criteria; however, due to the lack of information provided from the other two studies, attempting to determine the most effective tool from these three studies would not be fruitful.

Table 1 Studies Considered After Initial Exclusion ** = not reported

Study*	Seq. 1 Length (bp)	Seq. 2 Length (bp)	Alignment-Speed (GCUPS)	Run Time (s)	Accuracy	Time to Develop	Lines of Code
Liu *et al.*, 2018	64	64	**	20.14	100%	**	**
Vineetha *et al.*, 2019	**	**	**	115	99.20%	**	**
Chen *et al.*, 2021	256	256	51.20	165	99%	**	**
Rucci *et al.*, 2018	5.43×10^3	5.36×10^3	268.83	**	**	**	**
Pérez-serrano *et al.*, 2018	59.4×10^6	26.3×10^6	229.93	6802.18	**	**	**
Houtgast *et al.*, 2018	**	**	215	**	**	1 month	700 lines
Okada *et al.*, 2015	46.0×10^6	47.0×10^6	202	3240	**	**	**
Zou *et al.*, 2019	3.00×10^6	3.00×10^6	7.2	**	**	**	**
Rognes & Seeberg, 2000	**	**	1.5	**	**	**	**
Koliogeorgi *et al.*, 2019	**	**	**	5827	**	**	**
Bermúdez, 2019	900	900	**	3600	**	**	**
Ng *et al.*, 2020	**	**	**	3000	**	**	**

Ahmed *et al.*, 2015	**	**	**	530.5	**	**	**
Lee *et al.*, 2013	700	700	**	4.48	**	**	**
Snytsar & H, 2019	**	**	**	1.63	**	**	**
Li *et al.*, 2007	**	**	**	0.0428	**	**	**
Kehinde Bello & Alagbe Gbolagade, 2018	**	**	**	0.00723	**	**	**
Hasan & Al-Ars, 2007	**	**	**	0.0072	**	**	**
Park *et al.*, 2017	400	400	**	**	**	**	**
Khajeh-Saeed *et al.*, 2010	**	**	**	**	**	**	**
Iván *et al.*, 2016	**	**	**	**	**	**	**
Hakim *et al.*, 2019	**	**	**	**	**	**	**

*Full reference information located in Appendix A

Excluding accuracy, a total of five articles (23% of S-W articles) reported test sequence lengths, alignment speed in cell updates per second (CUPS), and hardware used in the study (Table 2). These five articles were used to identify the most efficient accelerated Smith-Waterman alignment method. None of the articles in the final consideration reported development time, lines of code, or file sizes of the tested sequences.

Two (Chen *et al.*, 2021; Rucci *et al.*, 2018) of the five methods utilized an FPGA design while the other three studies each utilize different methods (Table 2). The hardware design by Zou *et al.* (2019) utilizes an APU to accelerate the S-W algorithm, while Pérez-Serrano *et al.*, 2018 utilize multiple GPUs to increase alignment efficiency.

Okada *et al.*, 2015 have a software-based design using CUDA programming on two GPUs. The most efficient acceleration of the Smith-Waterman algorithm was determined by identifying the study with the fastest reported speed. Out of the five articles considered, the study by Rucci *et al.* (2018) had the fastest speed at 268.83 giga cell updates per second (GCUPS). So, it was determined that the FPGA design by Rucci (2018) was the most efficient for accelerating the Smith-Waterman algorithm.

Table 2 Studies Included in Final Comparison ****** = not reported

Study	Acceleration Model	Seq. 1 (bp)	Seq. 2 (bp)	**Alignment Speed (GCUPS)**	**Accuracy**	**Hardware**
Zou *et al.*, 2019	APU	3.00×10^6	3.00×10^6	7.2	**	AMD A12 APU
Chen *et al.*, 2021	FPGA	256	256	51.20	99%	Stratix-V GX FPGA
Okada *et al.*, 2015	CUDA programming	46.0×10^6	47.0×10^6	202	**	Dual Tesla K40 GPUs
Pérez-Serrano, *et al.*, 2018	Multi-GPU	59.0×10^6	26.0×10^6	229.93	**	Quad GeForce GTX980 GPU
Rucci *et al.*, 2018	FPGA	543×10^3	536×10^3	268.83	**	Intel Arria 10 GX FPGA

CHAPTER V

DISCUSSION

The overall goal of this project was to identify the current most efficient and effective methods for accelerating S-W-based sequence alignment by exploring several different S-W-based alignment tools. Many NGS alignment methods were obtained, but the results do not provide enough information to conclusively determine an ideal method for accelerating the Smith-Waterman algorithm.

Out of 22 articles pertaining to the S-W algorithm, only one article contained enough information for the final criteria to rank its efficiency and effectiveness. However, a single article is not sufficient to support the original hypothesis as there is not enough information to make a valid comparison to identify the most efficient and effective acceleration method. Therefore, the original hypothesis was not supported because a most effective method for accelerating the S-W algorithm could not be accurately determined due to the lack of available data.

Although one article fully satisfied the final criteria, there were four other articles that contained enough information to rank efficiency. The speeds of the five studies ranged from 7.2 giga cell updates per second (GCUPS) to 268.83 GCUPS. Among the five articles considered, the study by Rucci *et al.* (2018) had the fastest speed at 268.83 GCUPS when testing sequences of order 10^3 (i.e., ~500 $\times 10^3$ base pairs). The study by

Pérez-Serrano (Pérez-Serrano *et al.*, 2018) was the next fastest at 229.93 GCUPS when testing much larger sequences that had an order of 10^6 (i.e., ~25-60 $\times 10^6$ base pairs).

There was less available data from the sample than expected. In particular, each of the five studies used a different method for accelerating the original S-W algorithm, with the exception of the designs by Rucci *et al.* (2018) and Chen *et al.* (2021). Ideally there should be multiple studies that utilize similar acceleration methods and meet the criteria for analysis. The two FPGA studies achieved their alignment speeds by testing sequences of different sizes and resulted in the study by Chen *et al.* (2021) reporting an alignment speed that was 217.63 GCUPS lower than that of Rucci's design (Rucci *et al.*, 2018). The sizes of the tested sequences should not greatly affect alignment speed. Differences in speed between these FPGA designs is likely due to the specific type of FPGA used to run the S-W algorithm in each study. However, with few studies available, it can be said that there are not enough data points to conclusively determine which method is indeed the most efficient one. Therefore, the original hypothesis was not supported because a most efficient method for accelerating the S-W algorithm could not be conclusively determined from this sample.

Inconsistency in reporting among articles was another issue. Out of a total of 22 articles using accelerated S-W tools for pairwise alignment of genetic sequences, 17 articles had to be excluded from the study due to a lack of standardization in reporting data, representing 77% of the potential samples. These findings illustrate a new potential problem in determining an ideal method to both effectively and efficiently accelerate the S-W algorithm. The lack of consistency in data reporting may continue to create difficulties in improving tools that utilize the S-W algorithm. At a minimum, it is recommended that future studies in this field should report the sizes of the sequences tested, alignment speed in cell updates per second (CUPS), accuracy of the accelerated alignment method, and specifications of hardware used in the study. Additionally, it would be helpful to report information concerning the time required to develop the accelerated alignment method, execution time of the tool used in the study, and the number of lines of code used in the tool where applicable. This raises the possibility that no standard exists for reporting this type of data. The results of this project suggest that no such standard exists, yet.

There are several limitations in this project that may influence the findings. First, there was a lack of consistency among articles (Table 1). This lack of consist data reporting prevented some articles from being included in the comparison to identify the most efficient and effective S-W alignment method. Four studies did not report their alignment speed using a standard metric such as CUPS or execution time. Eight studies reported execution time but did not report the lengths of the sequences tested in their respective studies. Execution time alone does not provide enough information for a fair comparison because the time required to align sequences increases with the lengths of the sequences being aligned. Similarly, two studies reported alignment speed using GCUPS, but lacked information about the lengths of sequences being tested. Three studies reported speed using total execution time of the tool and included the lengths of the sequences tested, but they failed to report alignment speed using CUPS. This is problematic because the total execution time of an alignment tool includes not only the time for alignment of the sequences, but also the time for other events such as determining the scoring scheme. The types of events affecting run time can also differ among tools utilizing different software methods.

Second, the articles included in the study were obtained using limited key phrases to search through three academic search engines: Google Scholar, PubMed, and Researcher. Expanding the search to additional search engines would likely significantly increase the number of studies, not including duplicate search results, obtained from the initial search depending on how many additional search engines were included. Finally, the studies included in this project were limited only to those that were accessible without paid subscriptions to various databases and written in English. Additionally, around 20 articles were encountered that could not be included because they required a subscription to access.

It is also worth noting that the initial goal of finding a best alignment method may be too broad. The study conducted by Rucci *et al.* (2018) was the fastest, but Pérez-Serrano *et al.* (2018) came close with larger sequences, suggesting that different methods may be more efficient for different purposes. There are many scenarios that call for different types of alignment, which were excluded from this study. For example, studies that utilize the S-W algorithm in multiple sequence alignment were excluded from the analysis to allow for fair comparison of speed. Pairwise alignment techniques can be used progressively in multiple sequence alignment, so certain accelerated alignment methods may be more suited to different needs.

Future work on this topic should explore standardization of reporting for S-W-based sequence alignment. The study by Chen *et al.* (2021) should set a benchmark for reporting data in future work in this field. Future studies for accelerating the S-W algorithm should report a complete set of information, including the sizes of the sequences tested, alignment speed in CUPS, accuracy of the accelerated alignment method, and specifications of hardware used in the study.

CHAPTER VI

CONCLUSION

This project initially aimed to identify the current most efficient method for accelerating Smith-Waterman-based sequence alignment. This type of sequence analysis can be used to determine homology between two sequences of interest. However, the findings revealed potential new problems in determining the ideal way to accelerate the Smith-Waterman algorithm. Out of the 22 articles considered for the study, 17 were excluded due to their inconsistent reporting of data. An ideal method for accelerating the Smith-Waterman algorithm could not be accurately determined due to the lack of data. Such a lack of a standardized reporting protocol suggests that the current reporting methods are not sufficient and that future work in this topic needs to explore standardization of reporting for Smith-Waterman-based sequence alignment. Future studies should include sizes of the test sequences, alignment speed, and accuracy of the accelerated method, and specifications of hardware used in the study.

REFERENCES

REFERENCES

Barnes, R. (2020). A Review of the Smith-Waterman GPU Landscape. *Electrical Engineering and Computer Sciences University of California at Berkeley*. Retrieved from https://www2.eecs.berkeley.edu/Pubs/TechRpts/2020/EECS-2020-152.html

Behjati, S., & Tarpey, P. S. (2013). What is Next Generation Sequencing? *Archives of Disease in Childhood: Education and Practice Edition*, *98*(6), 236–238. https://doi.org/10.1136/archdischild-2013-304340

Chen, Y. L., Chang, B. Y., Yang, C. H., & Chiueh, T. D. (2021). A High-Throughput FPGA Accelerator for Short-Read Mapping of the Whole Human Genome. *IEEE Transactions on Parallel and Distributed Systems*, *32*(6), 1465–1478. https://doi.org/10.1109/TPDS.2021.3051011

Chow, R. D., & Chen, S. (2018). Sno-derived RNAs are Prevalent Molecular Markers of Cancer Immunity. *Oncogene*. https://doi.org/10.1038/s41388-018-0420-z

Gálvez, S., Ferusic, A., Esteban, F. J., Hernández, P., Caballero, J. A., & Dorado, G. (2016). Speeding-Up Bioinformatics Algorithms with Heterogeneous Architectures: Highly Heterogeneous Smith-Waterman (HHeterSW). *Journal of Computational Biology*, *23*(10), 801–809. https://doi.org/10.1089/cmb.2015.0237

Hakim, A., Kashtwari, A., Tiwari, R., & Sharma, J. (2019). *Performance Analysis of DNA Sequencing Using Smith- Waterman Algorithm on FPGA*. *9*(July), 1–5.

Han, M., Ding, C., & Kyung, H. M. (2015). Genetic Polymorphisms in Pattern Recognition Receptors and Risk of Periodontitis: Evidence based on 12,793 subjects. *Human Immunology*, *76*(7), 496–504. https://doi.org/10.1016/j.humimm.2015.06.006

Hendrix, D. A. (2019, October 3). *Chapter 3: Sequence Alignments*. Oregon State University.

Houtgast, E. J., Sima, V. M., Bertels, K., & Al-Ars, Z. (2018). Comparative Analysis of System-Level Acceleration Techniques in Bioinformatics: A Case Study of Accelerating the Smith-Waterman Algorithm for BWA-MEM. *Proceedings - 2018 IEEE 18th International Conference on Bioinformatics and Bioengineering, BIBE 2018*, 243–246. https://doi.org/10.1109/BIBE.2018.00053

Huang, T., Alvarez, A., Hu, B., & Cheng, S. Y. (2013). Noncoding RNAs in Cancer and Cancer Stem Cells. *Chinese Journal of Cancer*, *32*(11), 582–593. https://doi.org/10.5732/cjc.013.10170

Janes, R. W. (2005). Bioinformatics Analyses of Circular Dichroism Protein Reference Databases. *Bioinformatics*, *21*(23), 4230–4238. https://doi.org/10.1093/bioinformatics/bti690

Kehinde Bello, H., & Alagbe Gbolagade, K. (2018). Acceleration of Biological Sequence Alignment Using Residue Number System. *Asian Journal of Research in Computer Science*, *1*(2), 1–10. https://doi.org/10.9734/ajrcos/2018/v1i224735

Ligowski, L., & Rudnicki, W. (2009). An Efficient Implementation of Smith Waterman Algorithm on GPU using CUDA, for Massively Parallel Scanning of Sequence Databases. *IPDPS 2009 - Proceedings of the 2009 IEEE International Parallel and Distributed Processing Symposium*. https://doi.org/10.1109/IPDPS.2009.5160931

Liu, Y., Li, L., & Gao, J. (2018). An Improved Smith-Waterman Algorithm Based on Spark Parallelization. *International Journal Bioautomation*, *23*(1), 117–129. https://doi.org/10.7546/ijba.2019.23.1.117-129

Meldrum, C., Doyle, M. A., & Tothill, R. W. (2011). Next-Generation Sequencing for Cancer Diagnostics: A Practical Perspective. *Clinical Biochemist Reviews*, *32*(4), 177–195.

Moraes, F., & Góes, A. (2016). A Decade of Human Genome Project Conclusion: Scientific Diffusion About Our Genome Knowledge. *Biochemistry and Molecular Biology Education*, *44*(3), 215–223. https://doi.org/10.1002/bmb.20952

Nakagawa, H., & Fujita, M. (2018). Whole Genome Sequencing Analysis for Cancer Genomics and Precision Medicine. *Cancer Science*, *109*(3), 513–522. https://doi.org/10.1111/cas.13505

Palacios, J., & Triska, J. (2011). A Comparison of Modern GPU and CPU Architectures: And the Common Convergence of Both. *History of the GPU*, 1–20.

Pérez-Serrano, J., Sandes, E., Cristina, A., Alves, M., & Ujaldón, M. (2018). *Open Access DNA sequences alignment in multi-GPUs : acceleration and energy payoff*. *19*(Suppl 14). https://doi.org/10.1186/s12859-018-2389-6

Rucci, E., Garcia, C., Botella, G., De Giusti, A., Naiouf, M., & Prieto-Matias, M. (2018). SWIFOLD: Smith-Waterman Implementation on FPGA with OpenCL for Long DNA sequences. *BMC Systems Biology*, *12*(Suppl 5). https://doi.org/10.1186/s12918-018-0614-6

Salamat, S., & Rosing, T. (2020). FPGA Acceleration of Sequence Alignment: A Survey.

Department of Computer Science and Engineering, UC San Diego.

Smith, T. F., & Waterman, M. S. (1981). Identification of Common Molecular Subsequences. *Journal of Molecular Biology*, *147*(1), 195–197. https://doi.org/10.1016/0022-2836(81)90087-5

Tucker, T., Marra, M., & Friedman, J. M. (2012). Massively Parallel Sequencing. *Molecular Analysis and Genome Discovery: Second Edition*, *3*(3), 114–134. https://doi.org/10.1002/9781119977438.ch6

Vineetha, V., Biji, C. L., & Nair, A. S. (2019). SPARK-MSNA: Efficient Algorithm on Apache Spark for Aligning Multiple Similar DNA/RNA Sequences with Supervised Learning. *Scientific Reports*, *9*(1), 1–11. https://doi.org/10.1038/s41598-019-42966-5

Zou, H., Tang, S., Yu, C., Fu, H., Li, Y., & Tang, W. (2019). ASW: Accelerating Smith–Waterman Algorithm on Coupled CPU–GPU Architecture. *International Journal of Parallel Programming*, *47*(3), 388–402. https://doi.org/10.1007/s10766-018-0617-3

Appendix A: Studies Retrieved from Initial Search

Ahmed, N., Sima, V., Houtgast, E., Bertels, K., & Al-ars, Z. (2015). *Heterogeneous Hardware / Software Acceleration of the BWA-MEM DNA Alignment Algorithm*. 240–246.

Bermúdez, J. (2019). SLAST : *Simple Local Alignment Search Tool*. https://doi.org/10.1101/840546

Chen, Y. L., Chang, B. Y., Yang, C. H., & Chiueh, T. D. (2021). A High-throughput FPGA Accelerator for Short-read Mapping of the Whole Human Genome. *IEEE Transactions on Parallel and Distributed Systems*, *32*(6), 1465–1478. https://doi.org/10.1109/TPDS.2021.3051011

Hakim, A., Kashtwari, A., Tiwari, R., & Sharma, J. (2019). Performance Analysis of DNA Sequencing Using Smith- Waterman Algorithm on FPGA. *Journal of VLSI Design Tools & Technology, 9*(July), 1–5.

Hasan, L., & Al-Ars, Z. (2007). Performance improvement of the smith-waterman algorithm. 18th Annual Workshop on Circuits, Systems and Signal Processing(ProRISC), November, 211–214. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.80.3616&rep=rep1&type=pdf%5Cnhttp://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Performance+improvement+of+the+smith-waterman+algorithm#0%5Cnweb.nwfpuet.edu.pk/departments/Dcse_F_Publicat

Houtgast, E. J., Sima, V. M., Bertels, K., & Al-Ars, Z. (2018). Comparative analysis of system-level acceleration techniques in bioinformatics: A case study of accelerating the Smith-Waterman Algorithm for BWA-MEM. *Proceedings - 2018 IEEE 18th International Conference on Bioinformatics and Bioengineering, BIBE* 2018, 243–246. https://doi.org/10.1109/BIBE.2018.00053

Iván, G., Bánky, D., & Grolmusz, V. (2016). Fast and exact sequence alignment with the Smith-Waterman algorithm: The SwissAlign webserver. *Gene Reports*, 4(September 2013), 26–28. https://doi.org/10.1016/j.genrep.2016.02.004

Kehinde Bello, H., & Alagbe Gbolagade, K. (2018). Acceleration of Biological Sequence Alignment Using Residue Number System. *Asian Journal of Research in Computer Science*, 1(2), 1–10. https://doi.org/10.9734/ajrcos/2018/v1i224735

Khajeh-Saeed, A., Poole, S., & Blair Perot, J. (2010). Acceleration of the Smith-Waterman algorithm using single and multiple graphics processors. *Journal of Computational Physics*, 229(11), 4247–4258. https://doi.org/10.1016/j.jcp.2010.02.009

Koliogeorgi, K., Voss, N., Fytraki, S., Xydis, S., Gaydadjiev, G., & Soudris, D. (2019). Dataflow acceleration of smith-waterman with traceback for high throughput next generation sequencing. *Proceedings - 29th International Conference on Field-Programmable Logic and Applications, FPL 2019*, 74–80. https://doi.org/10.1109/FPL.2019.00021

Lee, S. T., Lin, C. Y., & Hung, C. L. (2013). GPU-based cloud service for smith-waterman algorithm using frequency distance filtration scheme. *BioMed Research International, 2013*, 1–8. https://doi.org/10.1155/2013/721738

Li, I. T. S., Shum, W., & Truong, K. (2007). 160-fold acceleration of the Smith-Waterman algorithm using a field programmable gate array (FPGA). *BMC Bioinformatics*, *8*, 1–7. https://doi.org/10.1186/1471-2105-8-185

Liu, Y., Li, L., & Gao, J. (2018). An improved Smith-Waterman algorithm based on Spark parallelization. *International Journal Bioautomation*, 23(1), 117–129. https://doi.org/10.7546/ijba.2019.23.1.117-129

Ng, H., Liu, S., Coleman, I., Chu, R., Yue, M., & Luk, W. (2020). Acceleration of Short Read Alignment with Runtime Reconfiguration. *Department of Applied Mathematics, Hong Kong Polytechnic University*

Okada, D., Ino, F., & Hagihara, K. (2015). Accelerating the Smith-Waterman algorithm with interpair pruning and band optimization for the all-pairs comparison of base sequences. *BMC Bioinformatics*, 16(1), 1–15. https://doi.org/10.1186/s12859-015-0744-4

Park, D., Beaumont, J., Mudge, T., & Arbor, A. (2017). Accelerating Smith-Waterman Alignment workload with Scalable Vector Computing. *Computer Science and Engineering* 1–8.

Pérez-Serrano, J., Sandes, E., Cristina, A., Alves, M., & Ujaldón, M. (2018). *Open Access DNA sequences alignment in multi-GPUs : acceleration and energy payoff. 19*(Suppl 14). https://doi.org/10.1186/s12859-018-2389-6

Rognes, T., & Seeberg, E. (2000). Six-fold speed-up of Smith-Waterman sequence database searches using parallel processing on common microprocessors. *Bioinformatics*, 16(8), 699–706.

Rucci, E., Garcia, C., Botella, G., De Giusti, A., Naiouf, M., & Prieto-Matias, M. (2018). SWIFOLD: Smith-Waterman implementation on FPGA with OpenCL for long DNA sequences. *BMC Systems Biology, 12*(Suppl 5). https://doi.org/10.1186/s12918-018-0614-6

Snytsar, R., & H, H. (2019). De (con) struction of the lazy-F loop : improving performance of Smith Waterman alignment. *2019 IEEE 19th International Conference on Bioinformatics and Bioengineering (BIBE), 2019, pp. 7-10, doi: 10.1109/BIBE.2019.00011.*

Vineetha, V., Biji, C. L., & Nair, A. S. (2019). SPARK-MSNA: Efficient algorithm on Apache Spark for aligning multiple similar DNA/RNA sequences with supervised learning. *Scientific Reports*, 9(1), 1–11. https://doi.org/10.1038/s41598-019-42966-5

Zou, H., Tang, S., Yu, C., Fu, H., Li, Y., & Tang, W. (2019). ASW: Accelerating Smith–Waterman Algorithm on Coupled CPU–GPU Architecture. International Journal of *Parallel Programming*, 47(3), 388–402. https://doi.org/10.1007/s10766-018-0617-3