CSE4001

Parallel and Distributed Computing

Project Report

Parallelization of The Smith-Waterman Algorithm

By

20BCE1229	Palesha Basumatary
20BCE1514	Nilashma Saha
20BCE1551	Bacham Sai Venkat Teja
20BCE1945	Jesmine Akhter

B.Tech. Computer Science and Engineering

Submitted to

Prof S K Ayesha

School of Computer Science and Engineering

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

November 2022

ABSTRACT

Parallel and Distributed computing is the future of technology. All products and their fundamental concepts are being shifted to a parallel computing model. Everybody would agree that serial computing is easy to implement and use but needs to be more efficient for industry-level purposes. Due to this reason, day by day higher number of industries is providing and using cloudsolutions that work based on parallel and distributed computing. For instance, Amazon’s AWS or Google’s Google Cloud platforms are becoming the center for development, be that in the web development field or data analytics.

But coming to the biotechnology field, the doors to parallel computing have yet to be opened much. There is much scope in this field for the use of parallel computing, but understanding where will employing the same bear fruit is also an important task. To cope with the fast-paced improvement in technology, domain familiarization is very important, and so is exploring the domain to discover areas where parallel computing can be employed. This project is an attempt to do the same.

The Gene sequencing problem is one of the significant issues for researchers regarding optimized system models that could help optimum processing and efficiency without introducing overheads in terms of memory and time. Bioinformatics and computational biology is the latest multidisciplinaryfield that explains many aspects of computer science, while computational biologyharnesses computational approaches and technologies to respond to biological questions conveniently.

INTRODUCTION

Genome is an emerging field, constantly presenting many new challenges to researchers in both biological and computational aspects for application. Sequence comparison is an essential operation. They detect similar or identical parts between two sequences called the query sequence and the reference sequence. Sequence comparison is significant for genetic analysis and finding the relationship between two organisms- whether they belong to the same species or genera or are unrelated- which opens new research fields. They indicate organisms' functional, structural, and evolutionary changes over time.

Sequence alignment is an integral part of the comparison. It is a way of arranging similar regions between two or more genomic sequences. Global and local alignments are the most prevalent kinds of sequence alignment. In global alignment, we find the superior counterpart between parts of the sequences. On the other hand,local alignment algorithms try to match parts of sequences, not the entirety. Local alignment is faster than global alignment due to the lack of need to align the entire sequence.

In our project, we would implement the Smith-Waterman Algorithm in a serial and parallel manner for comparison and analysis. As common sense suggests, the parallel implementation should execute and provide the same result as the serial implementationbut in less time. Furthermore, we will analyze the maximum common base pair count between the two sequences, followed by their agreement or deviation from the base-pair equality rule, and finally, use the similarity measure attained to decide in which field it is suitable to proceed with the analysis.

LITERATURE SURVEY

In molecular biology, a biological sequence comparison is essential for researchers. Since the growth rate of a biological sequence is exponentially high and the length of the sequence also becomes longer, requiring ample memory space and limiting the existing computational methods for data analysis. Multiple sequence alignment extension of pairwise alignment to alignment of more than two sequences at a time helps establish evolutionary relations. It is used in identifying conserved sequence regions across a group of sequences. Several methods are present for the alignment of two or more biological sequences.

Needleman -Wunsch algorithm is a dynamic programming algorithm that compares the sequences. It is used for global alignment by using optimal alignments of smaller subsequences. This algorithm consists of three steps: initialization of the score matrix, calculating the scores and filling the traceback matrix, and backtracking the alignment from the traceback matrix.

Another algorithm suggested by S. Aluru, N. Futamura, and K. Mehrotra is parallel algorithms using prefix computations. It includes affine gap penalty and subsequence matching and solves the comparison problems in O(mn) time and O(m+n) space where m, n is the length of two sequences. Significant time spent in all these dynamic programming algorithms was on calculating the dynamic programming matrix (an (n+1)x(m+1) matrix, where n and m are the lengths of two sequences), which is the core of these algorithms. These algorithms should store the dynamic programming in each parallel process to obtain an optimal result.

Some bioinformatics software tools, like FAST, BLAST, etc., are used for comparing biological sequences such as amino acid and nucleotide sequences. A few existing tools have a parallel implementation of the Smith-Watermanalgorithm, but the most prominent one is Clustal W [EMBOSS WATER]. **EMBOSS Water** uses the Smith-Waterman algorithm (modified for speed enhancements) to calculate the local alignment of two sequences. We can perform the alignment for protein, DNA, or RNA sequences.

There have been some approaches to parallelizing the Smith-Waterman Algorithm, notably Python, CUDA, OpenSHMEM, and MPI. Researchers are trying to parallelize the algorithm using a hybrid of MPI and OpenMP for efficient computation.

PROPOSED SOLUTION

- BASE ALGORITHM:

The smith-Waterman algorithm calculates the local alignment of two sequences. It guarantees tofind the best possible local alignment considering the specified scoring system. This includes a substitution matrix and a gap-scoring method. Scores consider a match, mismatch, and substitution. To measure the comparison between two sequences, a score is calculated as follows:

Given an alignment between sequences S0 and S1, the following values must be assigned for each column:

ma = (+5)	[Match]
mi = (-3)	[Mismatch]
G = (-4)	[Gap, can be any negative value decided by the stakeholders]

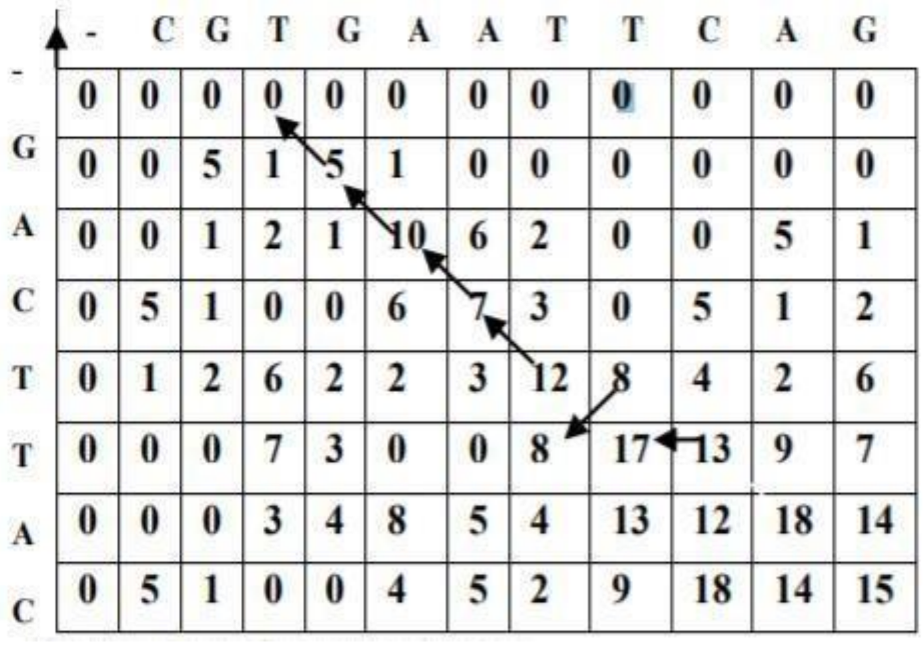
Procedure:

- Initialization of the matrix and consider two sequences, A and B.
- Matrix filling with suitable scores. The two sequences are set in a matrix form using A+1 columns and B+1 rows. The value in the first row and first column areset to zero.

$$= Max \begin{cases} M_{i-1,j-1} + S_{i,j}, \\ M_{i,j-1}+W, \\ M_{i-1,j} + W, \\ 0 \end{cases}$$

- The second and essential step of the algorithm is filling the entire matrix. To fill each cell, it is necessary to know the diagonal values.
- Trace back the sequence for an appropriate alignment is trace backing; before that, the maximum score obtained in the entire matrix must be detected for the local alignment of the sequences.
  Those maximum scores can likely be present in one or more than one cells. In such cases, there may be an option of two or more alignments, and the best alignment can be obtained by scoring it.

- Tracing back begins from the position with the highest value, pointing back with the pointers, consequently finding out the possible predecessor, then going to the next predecessor and continuing until it reaches the score 0.



	-	C	G	T	G	A	A	T	T	C	A	G
-	0	0	0	0	0	0	0	0	0	0	0	0
G	0	0	5	1	5	1	0	0	0	0	0	0
A	0	0	1	2	1	10	6	2	0	0	5	1
C	0	5	1	0	0	6	7	3	0	5	1	2
T	0	1	2	6	2	2	3	12	8	4	2	6
T	0	0	0	7	3	0	0	8	17	13	9	7
A	0	0	0	3	4	8	5	4	13	12	18	14
C	0	5	1	0	0	4	5	2	9	18	14	15

Predicted possible alignment using backtracking

- <u>MODIFICATIONS MADE:</u>

➢ Our model is designed to parallelize the Smith-Waterman algorithm, and we have used the OpenMP module in c to use the same.
➢ Based on the similarity matrix, we find the count of the number of bases (Adenine, Guanine, Thymine, and Cytosine) in the similar sequence obtained
➢ We find the highest base count and proceed to find if the bases in a similar sequence follow the base-pair quality rule, where

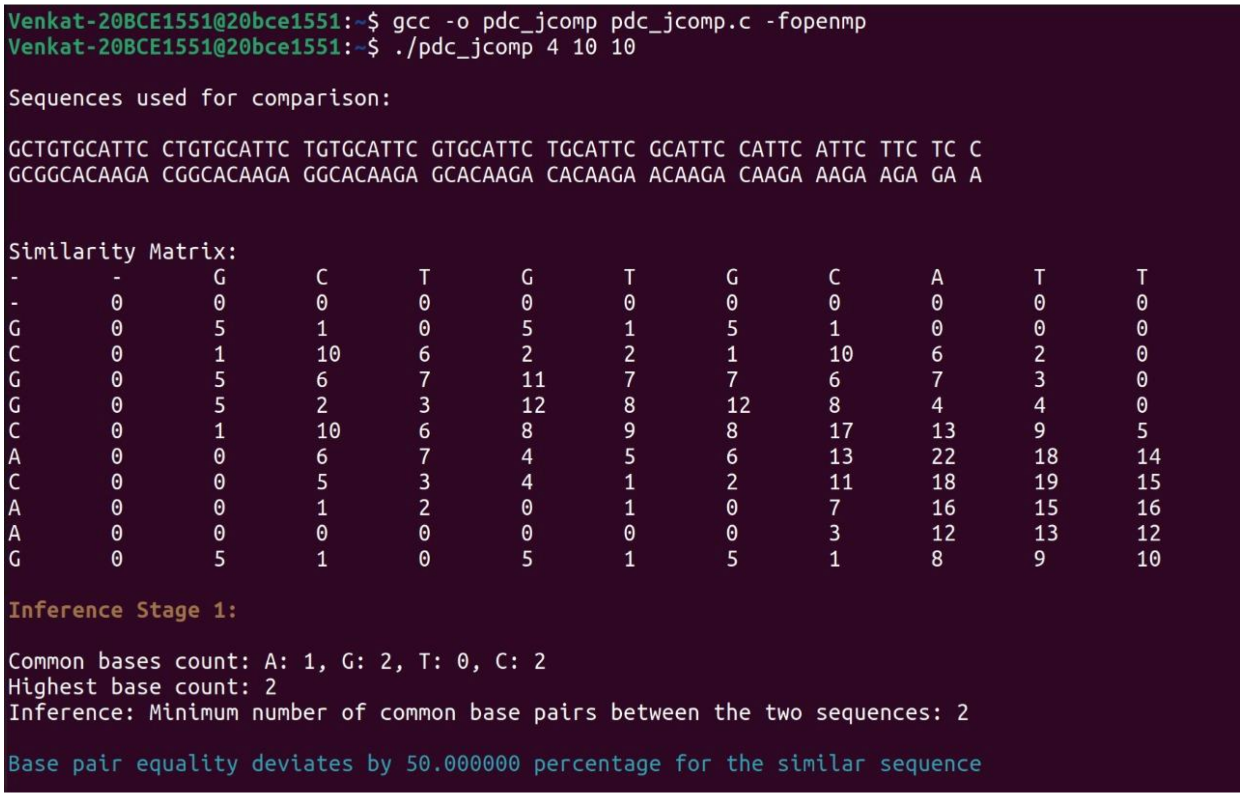
n(Adenine) + n(Guanine) = n(Thymine) + n(Cytosine)

Adenine pairs with Thymine via a double covalent bond and Guanine with Cytosine via a triple one.

➢ Based on the predictor matrix and the similarity matrix, we find the % of similarity between the two sequences and use the value obtained to infer what type of analysis should be approached to go forward with the experiment (for which similarity calculation has been employed)

RESULTS

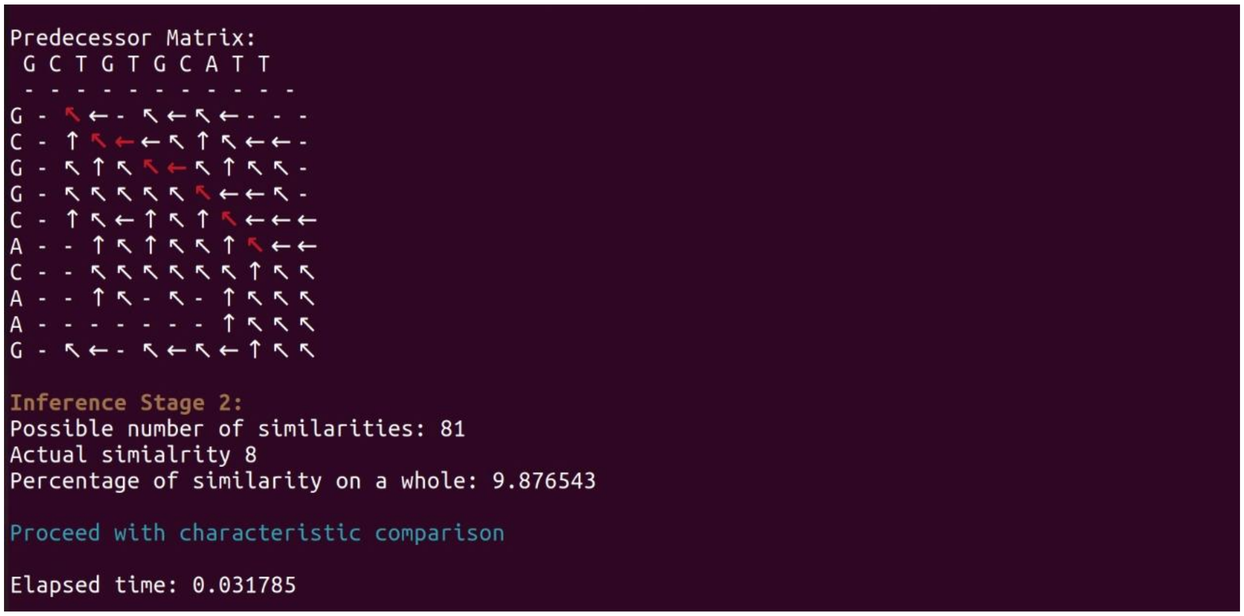
Our code, on simulation, gave the following output:



```
Venkat-20BCE1551@20bce1551:~$ gcc -o pdc_jcomp pdc_jcomp.c -fopenmp
Venkat-20BCE1551@20bce1551:~$ ./pdc_jcomp 4 10 10

Sequences used for comparison:

GCTGTGCATTC CTGTGCATTC TGTGCATTC GTGCATTC TGCATTC GCATTC CATTC ATTC TTC TC C
GCGGCACAAGA CGGCACAAGA GGCACAAGA GCACAAGA CACAAGA ACAAGA CAAGA AAGA AGA GA A


Similarity Matrix:
-       -       G       C       T       G       T       G       C       A       T       T
-       0       0       0       0       0       0       0       0       0       0       0
G       0       5       1       0       5       1       5       1       0       0       0
C       0       1       10      6       2       2       1       10      6       2       0
G       0       5       6       7       11      7       7       6       7       3       0
G       0       5       2       3       12      8       12      8       4       4       0
C       0       1       10      6       8       9       8       17      13      9       5
A       0       0       6       7       4       5       6       13      22      18      14
C       0       0       5       3       4       1       2       11      18      19      15
A       0       0       1       2       0       1       0       7       16      15      16
A       0       0       0       0       0       0       0       3       12      13      12
G       0       5       1       0       5       1       5       1       8       9       10

Inference Stage 1:

Common bases count: A: 1, G: 2, T: 0, C: 2
Highest base count: 2
Inference: Minimum number of common base pairs between the two sequences: 2

Base pair equality deviates by 50.000000 percentage for the similar sequence
```

The similarity matrix is calculated using backtracking and utilizing the gap penalty and negativity condition for each cell value in the matrix. The count for each base is calculated inside the missmatchscore function while finding the similarity matrix. The count of the highest occurring base is determined, and the same is shown on the screen.

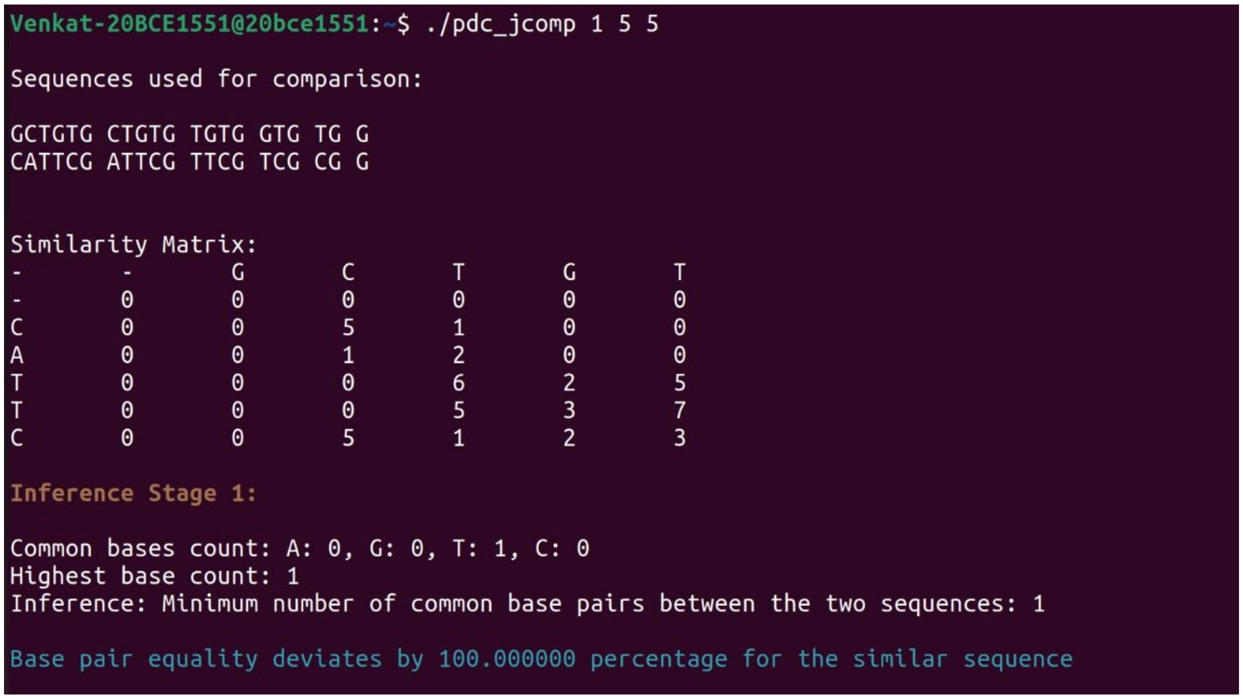
Next, we added the adenine count with guanine and thymine with cytosine and checked if they were equal, followed by the subsequent inferences.



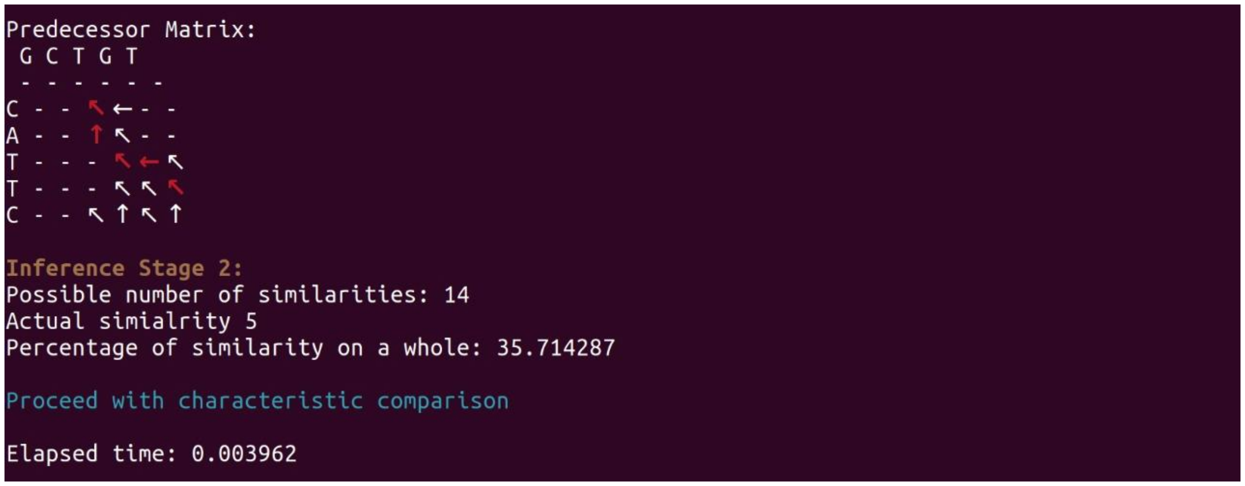
```
Predecessor Matrix:
 G C T G T G C A T T
 - - - - - - - - - - -
G - ↖ ← - ↖ ← ↖ ← - - -
C - ↑ ↖ ← ← ↖ ↑ ↖ ← ← -
G - ↖ ↑ ↖ ↖ ← ↖ ↑ ↖ ↖ -
G - ↖ ↖ ↖ ↖ ↖ ↖ ← ← ↖ -
C - ↑ ↖ ← ↑ ↖ ↑ ↖ ← ← ←
A - - ↑ ↖ ↑ ↖ ↖ ↑ ↖ ← ←
C - - ↖ ↖ ↖ ↖ ↖ ↖ ↑ ↖ ↖
A - - ↑ ↖ - ↖ - ↑ ↖ ↖ ↖
A - - - - - - - ↑ ↖ ↖ ↖
G - ↖ ← - ↖ ← ↖ ← ↑ ↖ ↖

Inference Stage 2:
Possible number of similarities: 81
Actual simialrity 8
Percentage of similarity on a whole: 9.876543

Proceed with characteristic comparison

Elapsed time: 0.031785
```

We created the predecessor matrix using a backtracking algorithm on the similarity matrix. Depending on the results, we have found similar matches and calculated the percentage of similarity between the two sequences based on (similar matches/total possible matches) *100. The subsequent inference has been drawn based on the rate of similarity obtained.

With a different sequence size:



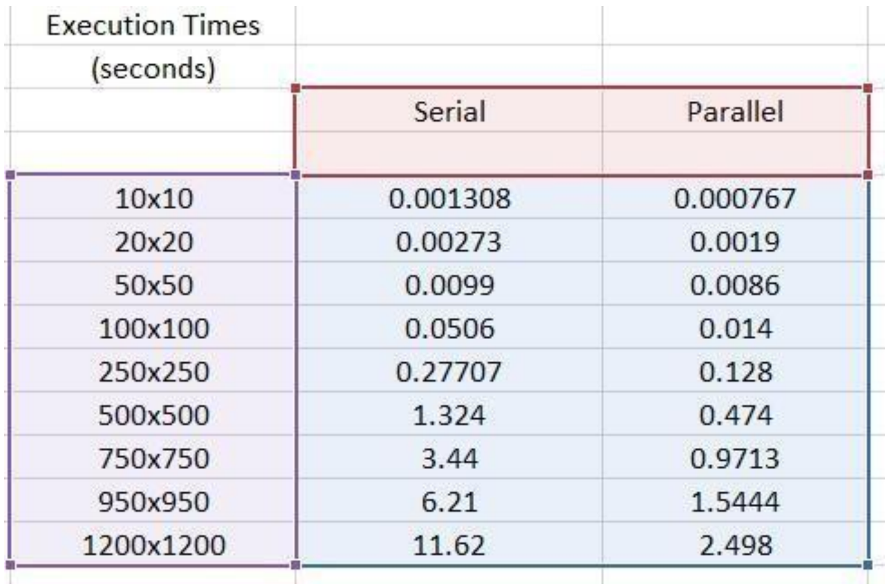
```
Venkat-20BCE1551@20bce1551:~$ ./pdc_jcomp 1 5 5

Sequences used for comparison:

GCTGTG CTGTG TGTG GTG TG G
CATTCG ATTCG TTCG TCG CG G


Similarity Matrix:
-       -       G       C       T       G       T
-       0       0       0       0       0       0
C       0       0       5       1       0       0
A       0       0       1       2       0       0
T       0       0       0       6       2       5
T       0       0       0       5       3       7
C       0       0       5       1       2       3

Inference Stage 1:

Common bases count: A: 0, G: 0, T: 1, C: 0
Highest base count: 1
Inference: Minimum number of common base pairs between the two sequences: 1

Base pair equality deviates by 100.000000 percentage for the similar sequence
```



```
Predecessor Matrix:
 G C T G T
 - - - - - -
C - - ↖ ← - -
A - - ↑ ↖ - -
T - - - ↖ ← ↖
T - - - ↖ ↖ ↖
C - - ↖ ↑ ↖ ↑

Inference Stage 2:
Possible number of similarities: 14
Actual simialrity 5
Percentage of similarity on a whole: 35.714287

Proceed with characteristic comparison

Elapsed time: 0.003962
```

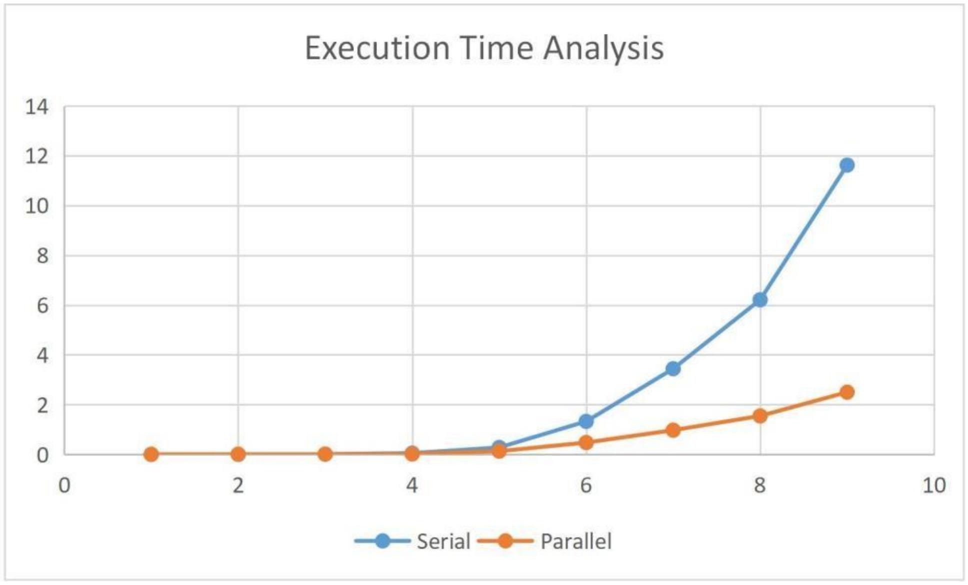
The code was run for various lengths of sequences, and the elapsed time was recorded in eachcase (as shown in the outputs).

After tabulating all the execution times, we got the following:



Execution Times (seconds)	Serial	Parallel
10x10	0.001308	0.000767
20x20	0.00273	0.0019
50x50	0.0099	0.0086
100x100	0.0506	0.014
250x250	0.27707	0.128
500x500	1.324	0.474
750x750	3.44	0.9713
950x950	6.21	1.5444
1200x1200	11.62	2.498

The graph depicting the difference between the execution time analysis for the serial and parallel algorithm is given below:



As shown in the above graph, for small lengths of the sequence, serial and parallel programs tend to give the output in almost the same amount of time. However, as the length increases, the execution time for serial implementation also increases exponentially, forming a steep graph.

CONCLUSION

The main goal of the proposed model is not just to decrease the computational time but also to aid in decision-making by indicating which direction of analysis we can proceed from here. From the comparative analysis,we can infer that our system outperforms the serially executing model implemented using the same algorithm. However, the parallel implementation remains stable with no high rise in the execution time because of the parallel execution of the task with two threads, making the process faster than its serial counterparts.

As a future scope, the model can be modified to determine the sequences that occur the most frequently and whether this contributes to an evaluation metric in gene sequencing. The model can also be compared with similar models made using CUDA, Python, and MPI- to determine which one will be the most effective practice or whether a hybrid model made out of these will serve the purpose better.

REFERENCES

[1] K. Mehrotra, S. Aluru and N. Futamura, "Parallel Biological Sequence Comparison Using Prefix Computations," in Parallel Processing Symposium, International, San Juan, Puerto Rico, 1999 pp. 653.

[2] Zhang, F., Qiao, X. Z., & Liu, Z. Y. (2002, October). A parallel smith-waterman algorithm based on divide and conquer. In Fifth International Conference on Algorithms and Architectures for Parallel Processing, 2002. Proceedings. (pp. 162-169). IEEE.

[3] Baker, M.B., Welch, A., & Venkata, M.G. (2015). Parallelizing the Smith-Waterman Algorithm Using OpenSHMEM and MPI-3 One-Sided Interfaces. *OpenSHMEM*.

[4] Prasad, D.V., & Jaganathan, S. (2018). Improving the performance of the Smith–Waterman sequence algorithm on GPU using shared memory for biological protein sequences. *Cluster Computing*, 1-10.

[5] Karamjeet Kaur et al 2022 J. Phys.: Conf. Ser. 2161 012028

[6] Oliveira, Fabio & Dias, Leonardo & Fernandes, Marcelo. (2021). Parallel Implementation of Smith-Waterman Algorithm on FPGA. 10.1101/2021.07.27.454006.

[7] El-Saghir, Z., Kelash, H.M., Elnazly, S., & Faheem, H.M. (2014). Parallel Implementation of Smith-Waterman Algorithm using MPI, OpenMP, and Hybrid Model.

[8] Laiq Hasan, Zaid Al-Ars, "Performance improvement of the Smith-Waterman Algorithm" Delft University of technology computer Engineering Laboratory.

[9] F. Guinand, "Parallelism for computational molecular biology," in ISThmus 2000 Conference on Research and Development for the Information Society, Poznan, Poland, 2000.

[10] Meng-Lai Yin, Hsien-yu Liao, Cheng. Y, "A parallel Implementation of the Smith-Waterman Algorithm for massive sequences searching," Engineering in Medicine and Biology Society, 26th Annual International Conference of the IEEE, vol.2, pp.2817, 2820, 1-5 Sept. 2004.