# CSE4001

# Parallel and Distributed Computing

# Project Report

Parallelisation 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



*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 simply not efficient enough for industry- level purposes. Due to this reason, day by day higher number of industries are providing and using cloud solutions which work on the basis on parallel and distributed computing. For instance, Amazon’s AWS or Google’s Google Cloud platform are becoming the center for development, may it be in the field of web development, or in the field of data analytics.

But coming to the field of biotechnology, the doors to parallel computing have not been opened much yet. 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 up with the fast-paced improvement in technology, domain familiarization is very important and so is exploring the domain to find out areas where parallel computing can be employed. This project is an attempt to do the same.

Gene sequencing problem is one of the major issues for researchers regarding optimized system models that could help optimum processing and efficiency without introduction overheads in terms of memory and time. Bioinformatics and computational biology is a latest multidisciplinary field which explains many aspects of the fields of computer science, while computational biology harnesses computational approach and technologies to respond biological questions conveniently.

# INTRODUCTION

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

Sequence alignment is an integral part of comparison. It is a way of arranging the similar regions between two or more genomic sequences. The 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 and not the entirety of them. Local alignment is faster than global alignment, due to the lack of need to align the entire sequences.

In our project, we would be implementing the Smith-Waterman Algorithmin a serial and parallel manner to for comparison and analysis. As common sense suggests, the parallel implementation should execute and provide the same result as the serial implementation but in a lesser amount of 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 is it suitable to proceed the analysis with.

# LITERATURE SURVEY

In the world of molecular biology, a biological sequence comparison is an important tool for researchers. Since the growth rate of a biological sequence is exponentially high and the length of the sequence becomes long, it requires large memory space and hence limits the existing computational methods for data analysis. Multiple sequence alignment extension of pairwise alignment to alignment of more than two sequences at a time which helps in establishing 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 basically used for global alignment by using optimal alignments of smaller subsequences. This algorithm consists of three steps that include initialization of the score matrix, calculating the scores and filling the traceback matrix, and lastly 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. Major 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. There are few existing tools which have a parallel implementation of the Smith-Waterman algorithm, 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 using Python, CUDA, OpenSHMEM and MPI. Researches are trying to parallelize the algorithm using a hybrid of MPI and OpenMP for efficient computation

**PROPOSED SOLUTION**

* BASE ALGORITHM:

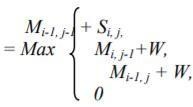
Smith-Waterman algorithm calculates the local alignment of two sequences. It guarantees to find out the best possible local alignment taking into account the specified scoring system. This includes a substitution matrix and a gap-scoring method. Scores consider match, mismatch and substitution. To measure 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]

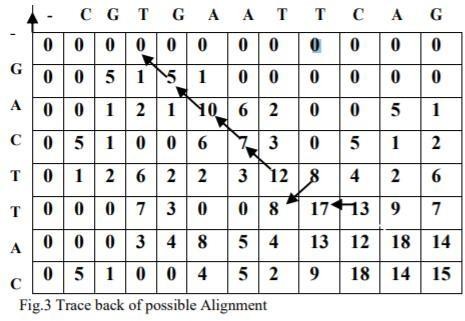
Procedure:

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



* The second and essential step of the algorithm is filling to entire matrix. To fill each and every cell it is important 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 has to be detected for the local alignment of the sequences.

It is likely to those maximum scores can be present in one or more than one cell, in such case there may be option of two or more alignments, and the best alignment can be obtained by scoring it.

* Tracing back begins from the position which has the highest value, pointing back with the pointers, consequently find out the possible predecessor, then go to next predecessor and continue until it reaches the score 0.

Predicted possible alignment using backtracking

* MODIFICATIONS MADE:
* Our model is designed to parallelize the Smith-Waterman algorithm, and we have used 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 the similar sequence follow the base-pair quality rule, where

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

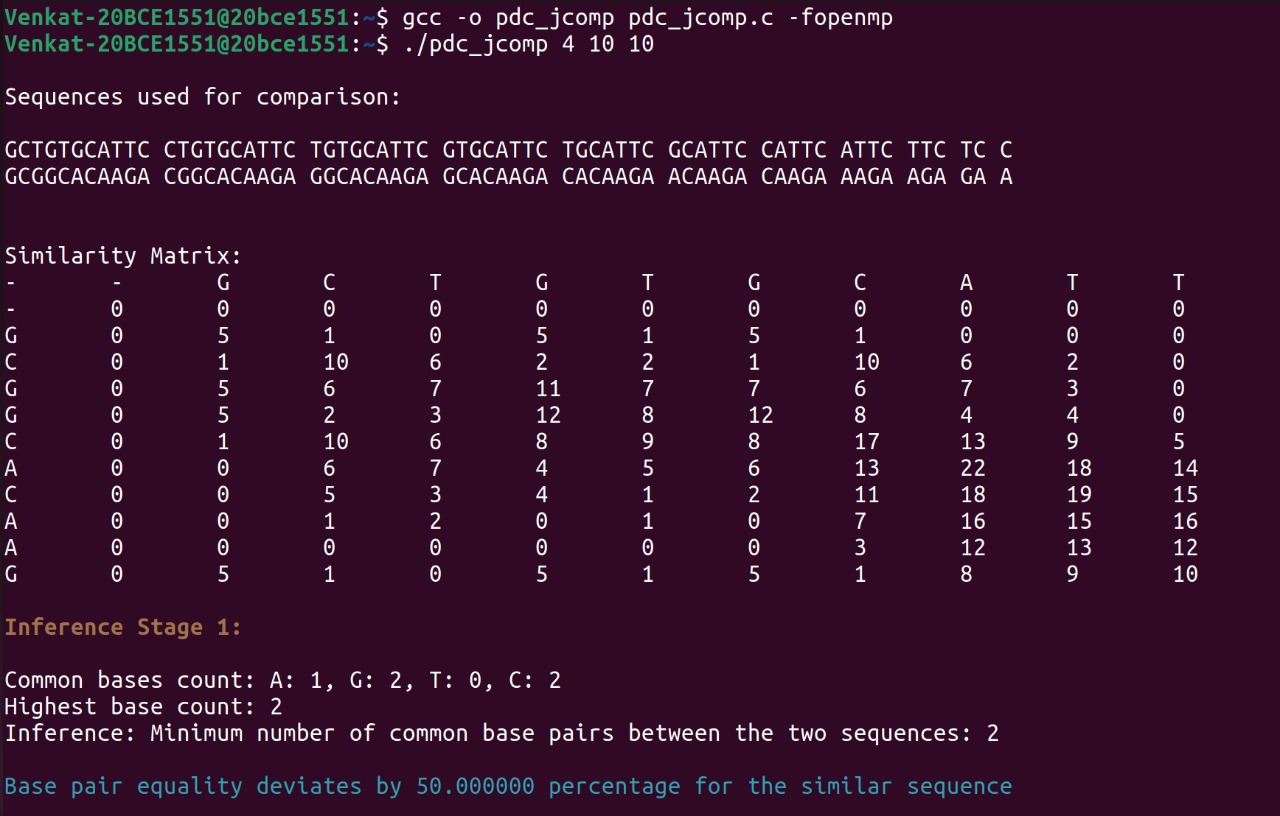
wher e, Adenine pairs with Thymine via a double covalent bond

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 to obtained to infer what type of analysis should be approached to go forward with the experiment (for which similarity calculation has been employed)

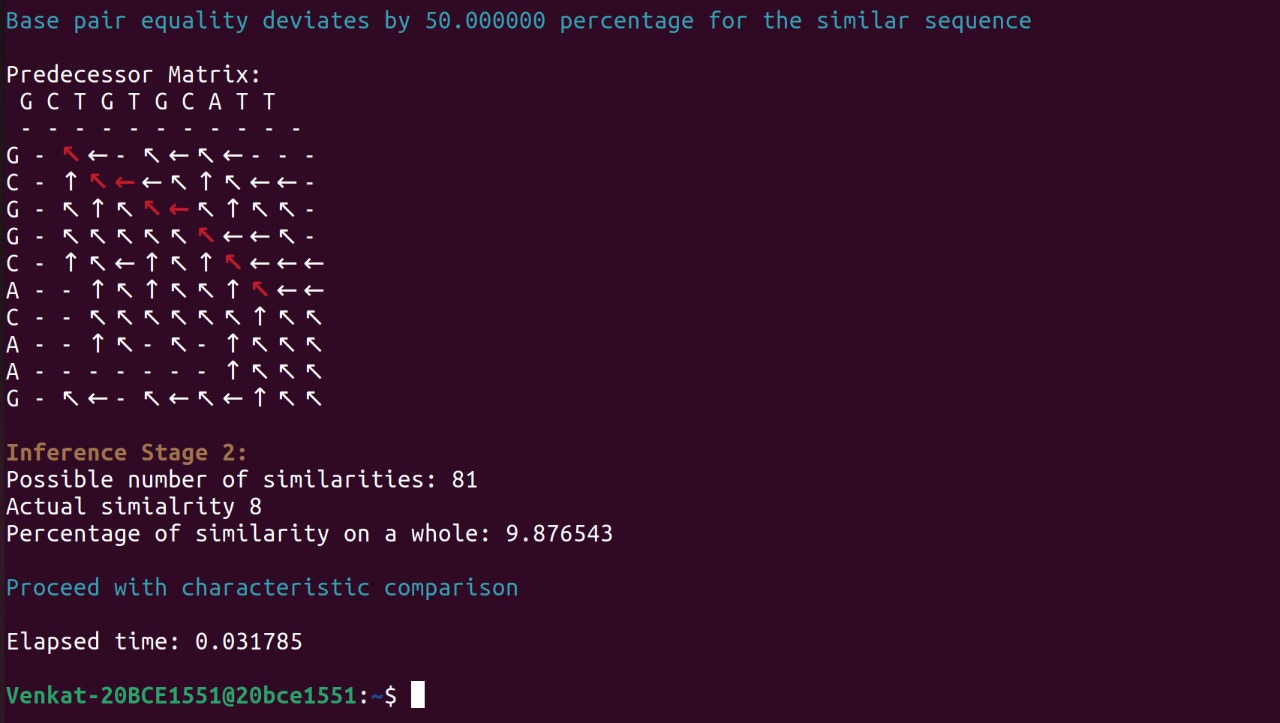
# RESULTS

Our code, on simulation, gave the following output:



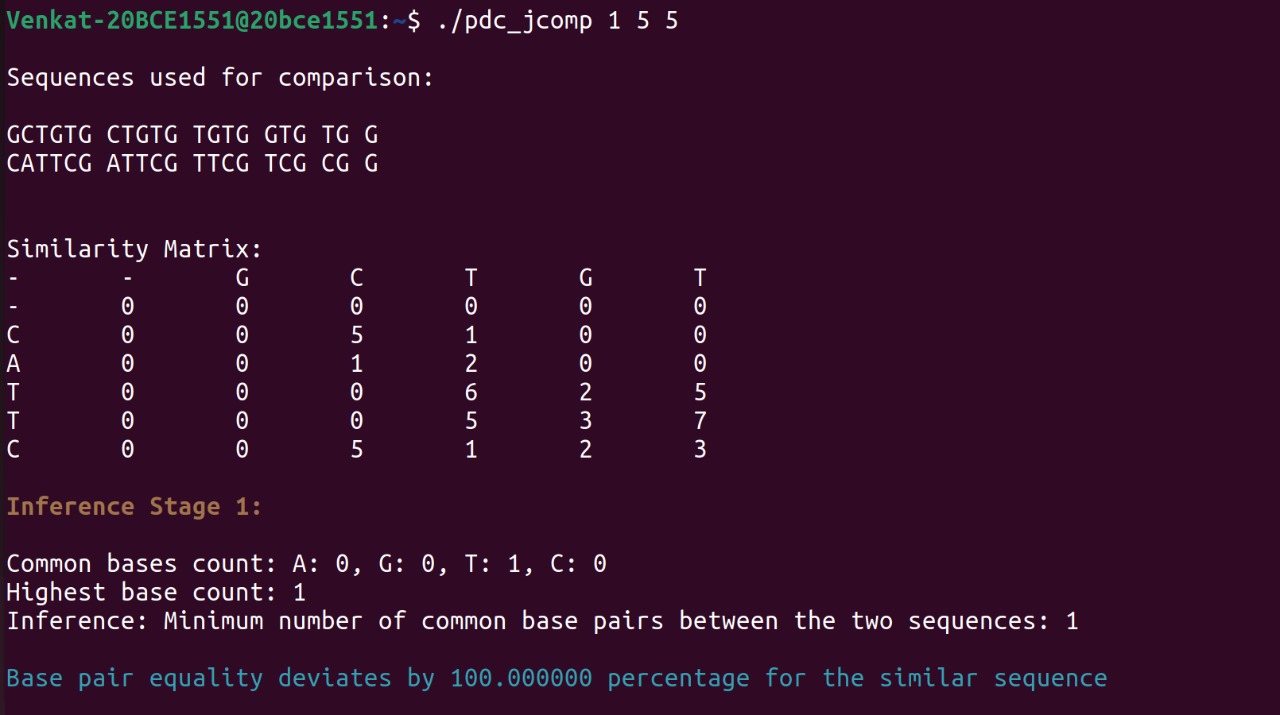
The similarity matrix is calculated using backtracking, and utilizing the gap penalty and negativity condition for each cell value in the matrix. The count for each base is calculated inside the missmatchscore function while finding the similarity matrix. The count of the highest occurring base is figured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

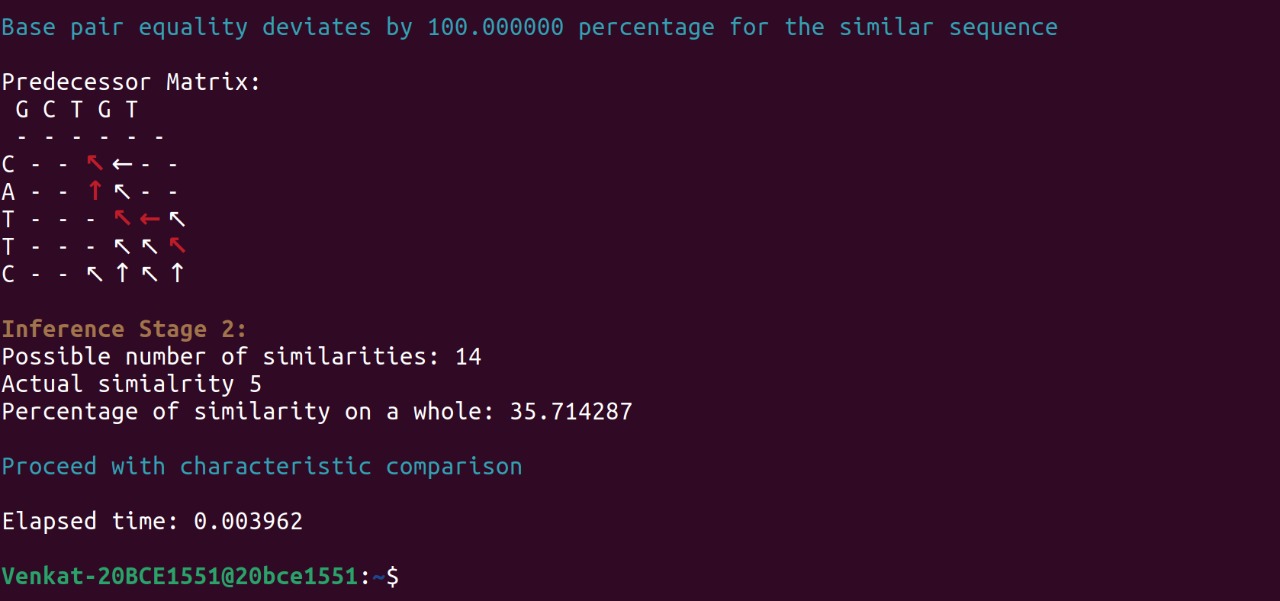
Next up, we have added the adenine count with guanine, while thymine and cytosine on the other hand and checked if they are equal, followed by the subsequent inferences.



We created the predecessor matrix using backtracking algorithm on the similarity matrix. Depending on the results, we have found out the 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 percentage of similarity obtained.

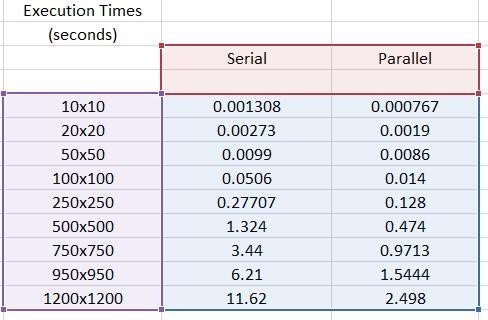
With a different sequence size:



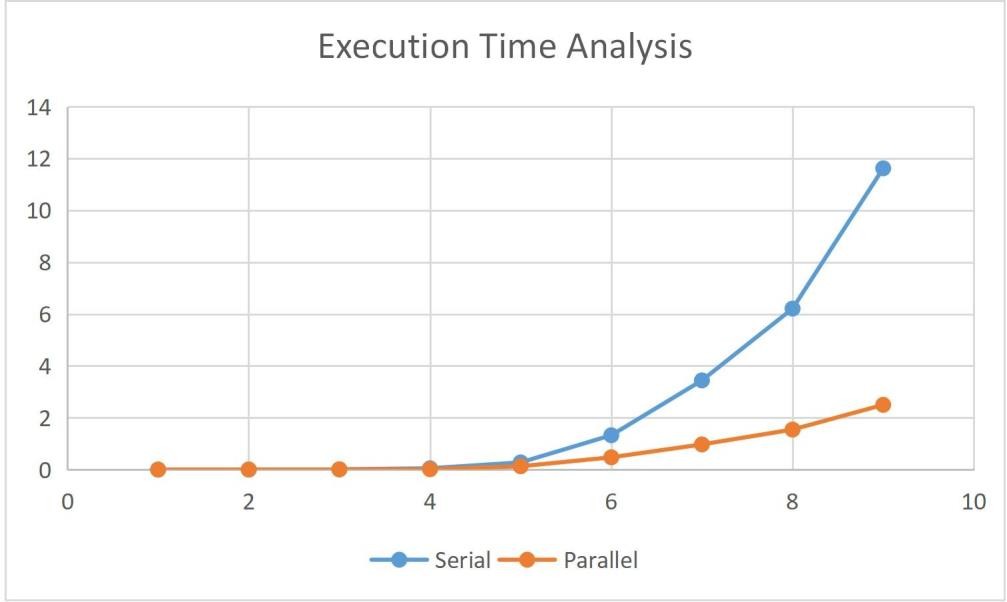


The code was run for various lengths of sequences, and the elapsed time was recorded in each case (as shown in the aforementioned outputs).

After tabulating all the execution times, we got:



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



As we can see 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, hence 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 what field of analysis can we 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 not a 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 in order to figure out the sequences occur the most frequently and whether this contributes to an evaluation metric in gene sequencing. The model can also be compared with other similar models made using CUDA, Python and MPI- to figure out which one will be the most effective practically 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 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.