Disease Diagnosis Using Pattern Matching in DNA Sequences

Sriram Anusha and Raju Bhukya

NIT Warangal Telangana, India

DOI: https://doi.org/10.51244/IJRSI.2024.1111009

Received: 21 October 2024; Accepted: 26 October 2024; Published: 26 November 2024

ABSTRACT

The introduction of parallel algorithms, specifically OPSI (Overlapping Pair-Set Indexing) and the Finite State Automaton (FSA), addresses the imperative need for accelerated DNA sequence pattern matching in the dynamic landscape of biological databases. As these databases undergo frequent updates, the demand for faster, error-reduced algorithms in real-world applications intensifies. Experimental results showcase the parallel approach's superior optimization and time efficiency, demonstrating its prowess over serial versions. Rigorous testing on the NCBI 'Homo Sapiens' dataset thoroughly evaluates all four algorithms—OPSI, FSA, and their serial counterparts—for disease diagnosis, meticulously recording and comparing their respective outputs and time efficiencies.

Keywords: Pattern matching, Disease Diagnosis, DNA sequencing, Nucleotide repeat diseases, OPSI Algorithm, Aho-Corasick Algorithm, Multithreading, Python, NCBI 'Homo Sapiens' Dataset.

INTRODUCTION

Pattern matching is a critical and dynamic field in computational molecular biology, aimed at discerning the frequency and distribution of candidate patterns within sequences or texts. These patterns typically define sets of strings composed of symbols. The primary goal is to determine a pattern's frequency and explore its occurrences in a given sequence or text. Numerous algorithms have been developed, each tailored to specific search types, employing different processing approaches and methods for optimal processing times.

Bioinformatics, a multidisciplinary science, employs mathematical and computer science principles to analyze biological data. DNA, life's fundamental building block, is represented as a sequence of four alphabets: A, C, G, and T. With the exponential growth of DNA data extraction, pattern matching techniques play a crucial role in computational biology applications, particularly in protein and gene data analysis. These techniques assist in locating specific patterns within DNA sequences, especially when biologists query new discoveries against sequence databases.

As data sizes expand, retrieving information from sequences becomes challenging, demanding more efficient and robust methods for rapid pattern matching. This area has been extensively studied in bioinformatics, with a focus on string matching. String matching entails searching for a specific pattern (P) within a large sequence or text (T) to determine its presence and, if found, indicating its position in T. If P is not present, the search confirms its absence in the given text.

Many existing pattern matching algorithms are reviewed and classified in two categories.

- Approximate Pattern Matching (APM)
- Exact Pattern Matching (EPM)

Exact pattern matching is a computational algorithmic technique used to find occurrences of an exact pattern within a given text or sequence. In this context, "exact" means an identical and complete match. The goal is to locate instances where the entire pattern appears exactly as it is specified in the text.

The problem can be formally defined as follows: Given a text (T) and a pattern (P), both composed of symbols from a finite alphabet, the task is to determine all positions in the text where the pattern occurs as a substring. If the pattern is not present in the text, the algorithm should report its absence.

Approximate word matching, also known as approximate string matching or fuzzy string matching, involves finding occurrences of a pattern within a text with some allowance for variations or errors. In contrast to exact pattern matching, where the goal is to find identical matches, approximate word matching allows for differences such as substitutions, insertions, deletions, or transpositions of characters within the pattern.

This type of matching is particularly useful in scenarios where the data might be noisy, or there might be errors or mutations in the text. In the context of computational biology, approximate word matching can be applied to identify similar sequences in DNA, RNA, or protein data.

BACKGROUND AND RELATED WORK

String matching mainly deals with problem of finding all occurrences of a string in a given text. In most of the DNA applications it is necessary for the user and the developer to be able to locate the occurrences of specific pattern in a sequence.This section reviews some work related to DNA sequences. An alphabet set ∑ = {A, C, G, T} is the set of characters for DNA sequence which are used in this algorithm. The following notations are used in this paper.DNA sequence characters ∑= {A, C, G, T}. Denotes the empty string. S[n] Denotes that a text which is a string of length n. T[m] Denotes a pattern of length m.The Brute-Force algorithm is a simple pattern matching technique that starts by comparing the first character of the pattern (P) with the first character of the text (T). It then proceeds to match characters one by one until a mismatch is found or the end of the pattern is reached. In case of a mismatch, the pattern is shifted one character to the right, and the matching process continues.Complexity of the algorithm is O(mn), where m is the pattern length and n is the text length.Boyer-Moore is an efficient algorithm that matches the pattern from right to left, applying larger shift increments for mismatch detection. If a mismatch is found, the pattern is shifted right so that the mismatched character aligns with the rightmost occurrence of the same character in the pattern. This algorithm aims to reduce unnecessary character comparisons. Time complexity of the algorithm is O(n/m), whereas the worst-case time complexity is O(m+n). The Knuth-Morris-Pratt (KMP) Algorithm is based on a finite state machine automation. It preprocesses the pattern (P) to create a finite state machine that accepts transitions efficiently. This reduces unnecessary comparisons by avoiding recomparing characters that have already been matched. The complexity for both average and worst-case performance is O(m+n).Rabin-Karp is a probabilistic algorithm that uses hashing to quickly compare the pattern with substrings of the text. It has an average-case time complexity of O(n + m), where n is the text length and m is the pattern length. However, it can have a worst-case time complexity of O(nm).The Aho-Corasick Algorithm is designed for efficiently finding occurrences of multiple patterns in a single pass through the text. It constructs a finite state pattern matching machine from the keywords and processes the text in a single pass. The algorithm's preprocessing of patterns is done in linear time, leading to a time complexity of O(n), where n is the length of the text.

IMPLEMENTATION

Aho Corasick Algorithm is a multi-string exact pattern matching algorithm in which it should locate all occurrences of searching keywords in a single pass. It mainly covers two parts consisting of a finite state pattern matching machine from the keywords and using that it processes the string in a single pass. The main aim of the algorithm is to locate and identify all the substrings of set X which are keywords of K, where K is a finite set of strings to be searched. The main phases of this algorithm are preprocessing and searching. In the preprocessing phase it generates a finite state automata machine (FSM) for every pattern that is to be searched. The Aho- Corasick algorithm is based on finite state automata (FSA), it needs to preprocess the pattern P and after preprocessing it builds a FSA tree. One advantage of the algorithm is failure links, using that it guarantees that no character can be searched more than once. The last phase is the output function in which final results are obtained. In bioinformatics, the DNA strings as well as responsible disease affected genes are taken as an input and FSM is created according to that. The existed pattern is searched in large dataset of DNA and concludes with the result that whether that DNA has chances of some nucleotide repeat diseases as well as some cancer types.

In the initial phase (Fig. 1), memory allocation by the host ensures the efficient utilisation of resources throughout the DNA testing process. The creation of a Finite State Machine (FSM) serves as the backbone, orchestrating and coordinating the intricate steps involved in the genetic analysis. Robust error-checking mechanisms and failure handling protocols enhance the system's resilience, providing a proactive approach to potential issues. The subsequent stage witnesses the device meticulously testing the DNA sample, with a focused examination for specific genes of interest.

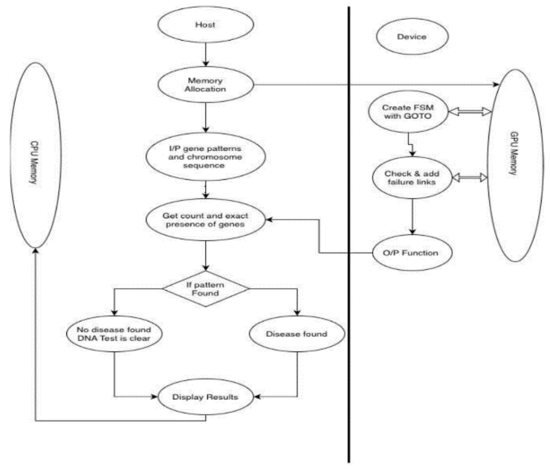

Host
Device
Memory Allocation
CPU Memory
GPU Memory
Create FSM with GOTO
I/P gene patterns and chromosome sequence
Check & add failure links
Get count and exact presence of genes
O/P Function
If pattern Found
No disease found DNA Test is clear
Disease found
Display Results

Figure 1

The FSM dynamically guides the process, responding to the intricate nuances of the genetic information. If the system successfully identifies a specific genetic pattern indicative of a disease, the process seamlessly advances. Finally, the result display stage not only communicates the outcome but also serves as a pivotal point of interaction, offering clarity on whether a disease-related genetic pattern has been detected or not. This user-centric approach ensures that the DNA testing flowchart is not just technically sound but also accessible and informative for end-users and healthcare professionals alike.

Optimised Pattern Similarity Identification algorithm ensures that each character in the sequence is compared only once, hence further improving time efficiency. It makes use of a Shift Value table for doing so.

OPSI algorithm identifies pattern similarity by allowing the mutation caused by substitution. During the search of a pattern in the DNA sequence, if the pattern is exactly matched with the current window of the sequence, the portion of the sequence matched with the pattern is not considered for approximate matching. Since the approximation is permitted, the time efficiency of the process depends on the number of mismatches considered. Hence, OPSI algorithm has a time complexity in the order of the product of the length of the text and the threshold for mismatches allowed (O (ls. è)). If the number of mismatches permitted is least considerable, then the OPSI algorithm is having the time complexity in linear order.

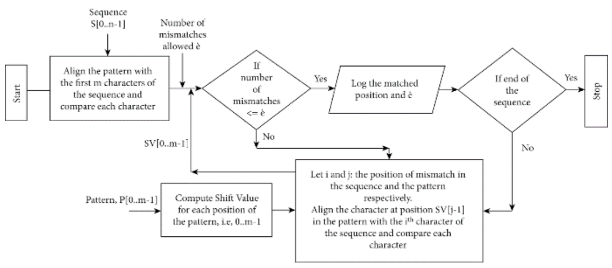

Sequence S[0..n-1]
Number of mismatches allowed è
Start
Align the pattern with the first m characters of the sequence and compare each character
If number of mismatches <= è
Yes
Log the matched position and è
If end of the sequence
Yes
Stop
No
SV[0..m-1]
No
Let i and j: the position of mismatch in the sequence and the pattern respectively. Align the character at position SV[j-1] in the pattern with the ith character of the sequence and compare each character
Pattern, P[0..m-1]
Compute Shift Value for each position of the pattern, i.e, 0..m-1

Figure 2 [2]

The OPSI pattern matching algorithm, encapsulated in the flowchart, presents a dynamic and efficient approach to identifying patterns within a given sequence. It commences with a sequence S[0..n-1] and a pattern P[0..m-1], initiating the comparison by aligning the pattern with the sequence. Upon detecting a mismatch, a pivotal step in the algorithm unfolds as it calculates shift values SV[0..m-1] for each position of the pattern.

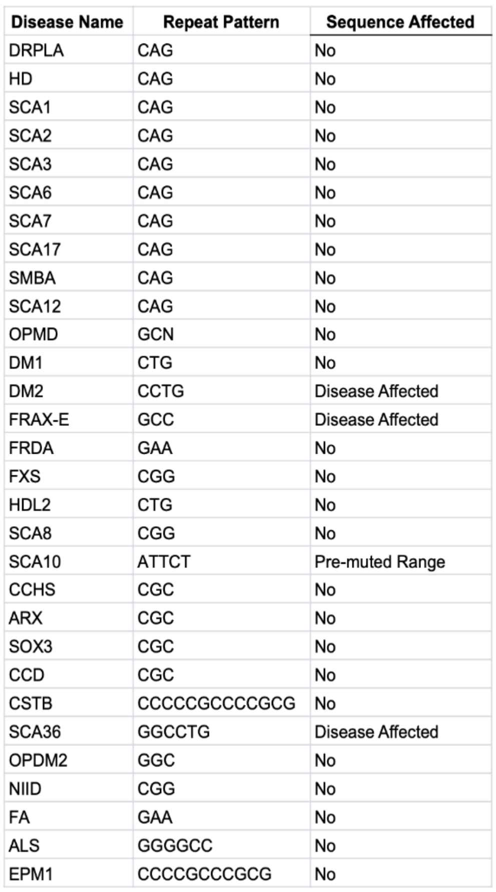
These shift values become instrumental in the subsequent realignment of the pattern with the sequence, strategically optimising the comparison process. The algorithm leverages these calculated shifts to intelligently adjust the position of the pattern, mitigating the impact of mismatches and enhancing the overall efficiency of the search. The iterative nature of this algorithm involves a cyclic process of comparison, detection of mismatches, calculation of shift values, and subsequent realignment.

This cycle repeats until the pattern is successfully located within the sequence or the entire sequence is exhaustively traversed. Notably, OPSI's strength lies in its ability to dynamically adapt to variations in the sequence, efficiently handling mismatches without the need for exhaustive backtracking. This robust and iterative strategy makes OPSI well-suited for scenarios where patterns may occur with variations or interruptions within the sequence. The flowchart provides a visual representation of this sophisticated pattern matching algorithm, elucidating the steps involved in its dynamic and adaptive approach to sequence analysis.

We will run all of them for the NCBI ‘Homo Sapiens’ Dataset. This dataset acts as a DNA (Genome) sequence of a patient on which we will perform Disease Diagnosis. We will also be making use of a disease dataset, which contains the information about the range of number of occurrences of a particular sequence, so that we could classify the DNA Sequence to be in ‘Normal Range’, ‘Pre-muted Range’ or ‘Disease-Affected Range’ for that particular disease.

EXPERIMENTAL RESULTS

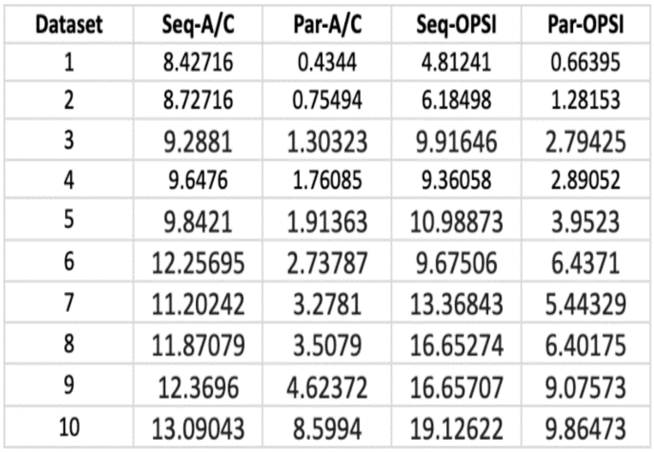
Upon analysing specific datasets, the results (Fig. 3) reveal that the patient faces potential risks of developing FRAX-E, DM2, and SCA36. Additionally, the assessment identifies a pre-muted sequence for SCA10, suggesting a predisposition to this genetic anomaly in its early stages. These findings provide valuable insights into the patient's genetic landscape, enabling timely interventions and informed healthcare decisions. The early identification of potential risks and pre-muted sequences contributes to a proactive approach in managing and addressing genetic susceptibilities.



Disease Name	Repeat Pattern	Sequence Affected
DRPLA	CAG	No
HD	CAG	No
SCA1	CAG	No
SCA2	CAG	No
SCA3	CAG	No
SCA6	CAG	No
SCA7	CAG	No
SCA17	CAG	No
SMBA	CAG	No
SCA12	CAG	No
OPMD	GCN	No
DM1	CTG	No
DM2	CCTG	Disease Affected
FRAX-E	GCC	Disease Affected
FRDA	GAA	No
FXS	CGG	No
HDL2	CTG	No
SCA8	CGG	No
SCA10	ATTCT	Pre-muted Range
CCHS	CGC	No
ARX	CGC	No
SOX3	CGC	No
CCD	CGC	No
CSTB	CCCCCGCCCCGCG	No
SCA36	GGCCTG	Disease Affected
OPDM2	GGC	No
NIID	CGG	No
FA	GAA	No
ALS	GGGGCC	No
EPM1	CCCCGCCCCGCG	No

Figure 3

The results (Fig. 4) depict a time analysis conducted on various DNA sequences to diagnose nucleotide repeat diseases. The analysis involves running the diagnostic process on multiple datasets, gradually increasing in size. Notably, the execution performance notably improves with parallel approaches, showcasing their effectiveness in handling larger datasets efficiently. As the patterns within the sequences increase, the parallel algorithms demonstrate superior speed-up compared to their sequential counterparts. These findings underscore the scalability and enhanced performance of parallel approaches in the context of nucleotide repeat disease diagnosis, affirming their suitability for large-scale genomic analyses.



Dataset	Seq-A/C	Par-A/C	Seq-OPSI	Par-OPSI
1	8.42716	0.4344	4.81241	0.66395
2	8.72716	0.75494	6.18498	1.28153
3	9.2881	1.30323	9.91646	2.79425
4	9.6476	1.76085	9.36058	2.89052
5	9.8421	1.91363	10.98873	3.9523
6	12.25695	2.73787	9.67506	6.4371
7	11.20242	3.2781	13.36843	5.44329
8	11.87079	3.5079	16.65274	6.40175
9	12.3696	4.62372	16.65707	9.07573
10	13.09043	8.5994	19.12622	9.86473

Figure 4

The results (Fig. 5) offer a time analysis of various algorithms employed in diagnosing nucleotide repeat diseases on a specific dataset. Notably, the parallel Aho-Corasick algorithm stands out by delivering the most optimal execution performance. Additionally, a noteworthy observation is that the parallel versions consistently outpace their sequential counterparts, highlighting the efficiency gains achieved through parallelization in the context of nucleotide repeat disease diagnosis.

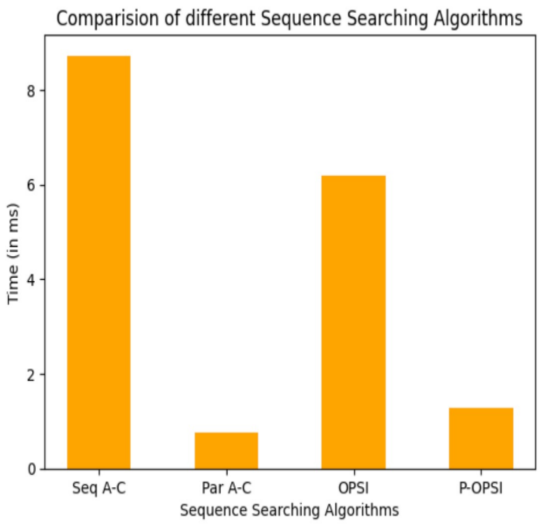


Figure 5

The results (Fig. 6) provide a direct comparison of the execution performance between the sequential Aho-Corasick algorithm and the sequential OPSI algorithm in diagnosing nucleotide repeat diseases. Notably, the analysis indicates a superior execution performance for the sequential OPSI algorithm when compared to its Aho-Corasick counterpart. This observation underscores the efficiency gains achieved by the OPSI algorithm in the context of sequential execution, highlighting its potential as a promising approach for nucleotide repeat disease diagnosis. The findings contribute valuable insights into the comparative strengths of these sequential algorithms, guiding the selection of optimal strategies for genomic analyses.

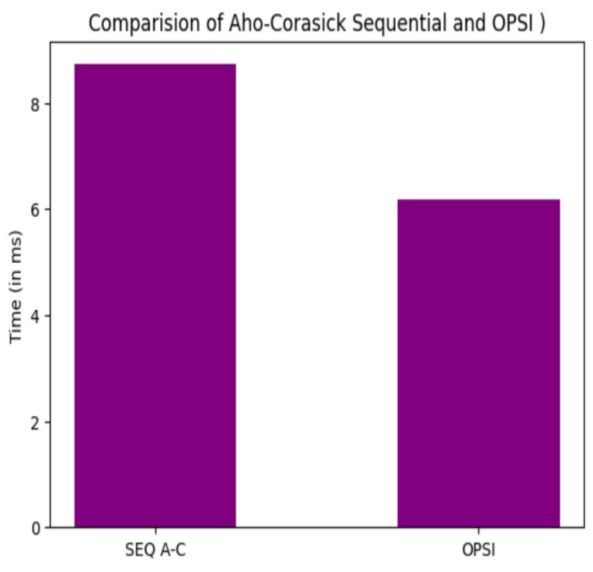


Figure 6

In the results (Fig. 7), a direct comparison is made between the execution performance of the parallel Aho-Corasick algorithm and the sequential OPSI algorithm in diagnosing nucleotide repeat diseases. Notably, the findings reveal superior execution performance for the parallel Aho-Corasick algorithm in contrast to its sequential OPSI counterpart. This observation emphasises the efficacy of parallelization, particularly in the context of the Aho-Corasick algorithm, as it outperforms the sequential OPSI approach in handling genomic analyses efficiently. The results contribute valuable insights into optimising algorithmic choices for nucleotide repeat disease diagnosis in parallel computing environments.

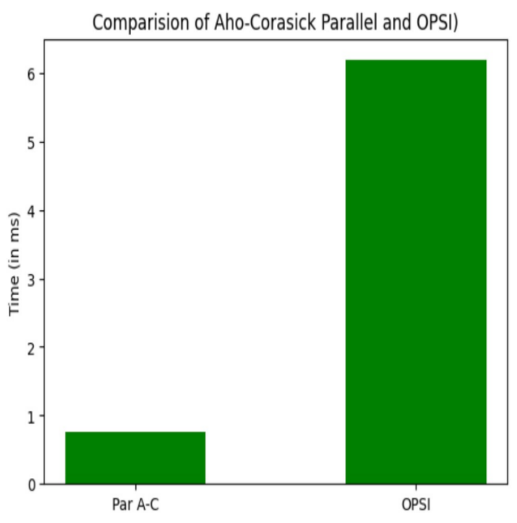


Figure 7

The results (Fig. 8) offer a detailed comparison between the execution performance of the parallel OPSI algorithm and its sequential counterpart in the diagnosis of nucleotide repeat diseases. Notably, the findings highlight a notable improvement in execution performance for the parallel OPSI algorithm when contrasted with the sequential OPSI approach. This observation underscores the effectiveness of parallelization specifically within the OPSI algorithm, showcasing its capacity to enhance the efficiency of nucleotide repeat disease diagnosis. The results contribute valuable insights into the advantages of parallel OPSI implementations, emphasising their potential impact on optimising genomic analyses and advancing computational approaches in the realm of bioinformatics. The findings encourage further exploration of parallel strategies for enhanced performance in nucleotide repeat disease diagnostics.

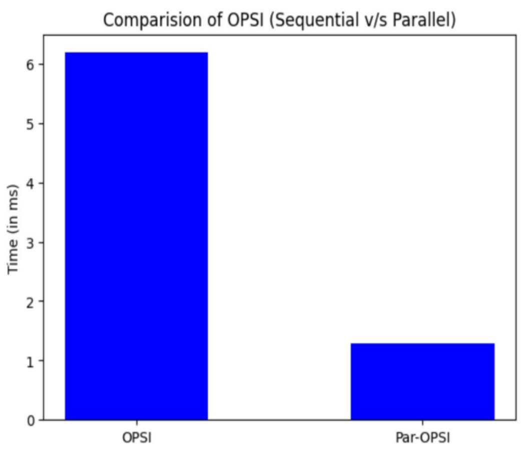


Figure 8

Our observation (Fig. 9) underscores a consistent trend wherein the parallel implementations of the optimised Aho-Corasick algorithm surpass both the sequential and parallel versions of the OPSI algorithm in terms of speed and efficiency. While acknowledging that the running time of both sequential and parallel algorithms naturally increases with the growth in file or dataset size, the parallel approach consistently exhibits superior performance when compared to its sequential counterpart.

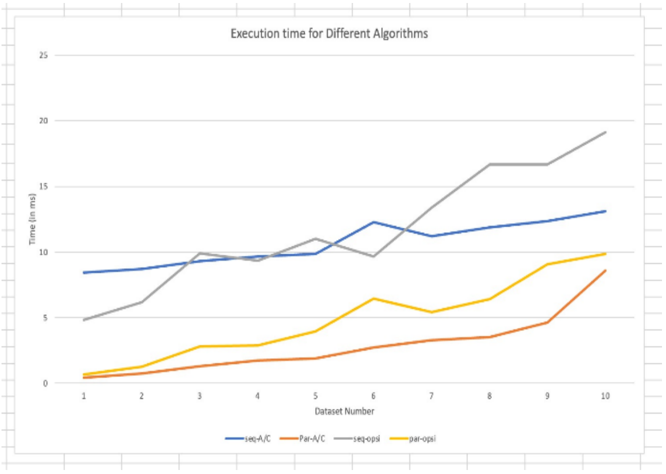


Figure 9

CONCLUSIONS

A pivotal phase of our study involved conducting Disease Diagnosis utilising both the serial and parallel implementations of the algorithms. Rigorous verification procedures were implemented to ensure the accuracy of the Disease Diagnosis outcomes. Remarkably, the adoption of parallel versions of the optimised Aho-Corasick algorithm yielded results within a significantly abbreviated timeframe, highlighting its prowess in expediting complex genomic analyses.

The implications of these findings extend to the realm of Bioinformatics applications, particularly in Disease Diagnosis through pattern-matching in DNA Sequences. The insights gleaned from this study hold promise for advancing computational approaches in Bioinformatics, offering heightened efficiency and precision in disease diagnostics and pattern-matching analyses. This study contributes valuable knowledge that could significantly impact the field, fostering advancements in Bioinformatics applications and genomic research.

This study, after vigorous testing and experiments, confirm that parallel version of Aho Corasick Algorithm is indeed the fastest method to find pattern matching.

REFERENCES

1. Osman Ali Sadek Ibrahim, Belal A. Hamed, and Tarek Abd El-Hafeez. A new fast technique for pattern matching in biological sequences. In Proceedings of The Journal of Supercomputing, pages 1573–1594, 2022.
2. J. A. M. Rexie, Kumudha Raimond, Mythily Murugaaboopathy, D. Brindha, and Henock Mulugeta. Lightweight pattern matching method for DNA sequencing in the internet of medical things. In Proceedings of the Computational Intelligence and Neuroscience, pages 1687–1702, 2022.
3. Sandeep U. Mane and Ketaki H. Pangu. Disease diagnosis using pattern matching algorithm from dna sequencing: a sequential and gpgpu based approach. In Proceedings of International Conference on Informatics and Analytics, pages 1–5, 2016.
4. Chandra Mohan Dasari Raju Bhukya. Disease diagnosis based on various pattern frequencies from extracted exons. In Proceedings of 2022 IEEE 3rd Global Conference for Advancement in Technology (GCAT), pages 1–7, 2022.
5. Abdullah Ammar Karcioglu and Hassan Bulut. Improving hash-q exact string matching algorithm with perfect hashing for dna sequences. In Proceedings of Computer in Biology and Medicine, pages 131–142, 2021.
6. Rajmani Arya Shekhar Prasad Raju Bhukya, Priyanka Agrawal. Exact and approximate repeats in dna sequences using suffix array and divide conquer. In Proceedings of HELIX, pages 1783–1789, 2017.
7. Raju Bhukya and DVLN Somayajulu. Exact multiple pattern matching algorithm using dna sequence and pattern pair. In Proceedings of International Journal of Computer Applications, pages 32–38, 2011.
8. Raju Bhukya and DVLN Somayajulu. An index based forward backward multiple pattern matching algorithm. In Proceedings of International Journal of Biomedical and Biological Engineering, pages 422–430, 2010.
9. Raju Bhukya and DVLN Somayajulu. 2-jump dna search multiple pattern matching algorithm. In Proceedings of Int. J. Comput. Sci. Issues, pages 320–329, 2011.
10. Paolo Ciaccia Marco Patella Stefano Ceri Piero Montanari, Ilaria Bartolini and Marco Masseroli. Pattern similarity search in genomic sequences. In Proceedings of IEEE Transactions on Knowledge and Data Engineering, pages 3053–3067, 2016.
11. Simone Faro and Thierry Lecroq. The exact online string matching problem: a review of the most recent results. In Proceedings of ACM Comput Surveys (CSUR), pages 1–42, 2013.
12. Steffen Heyne Gad M Landau Mathias Möhl Christina Otto Mika Amit, Rolf Backofen and Sebastian Will. Local exact pattern matching for non-fxedrna structures. In Proceedings of IEEE/ACM Trans ComputBiolBioinf, pages 219–230, 2013.
13. Simone Faro Domenico Cantone and Arianna Pavone. Linear and efcient string matching algorithms based on weak factor recognition. In Proceedings of ACM Journal of Experimental Algorithmics, pages 1–20, 2019.
14. Kalyan Kumar Kaipa Parthasarathy Venkataraman Kyusang Lee TaeJin Ahn Abhilash Srikantha, Ajit S. Bopardikar and Rangavittal Narayanan. A fast algorithm for exact sequence search in biological sequences using polyphase decomposition. In Proceedings of Bioinformatics, pages 414–419, 2010.
15. Steffen Heyne Mika Amit Gad M Landau Rolf Backofen Christina Otto, Mathias Möhl and Sebastian Will. Exparna-p: simultaneous exact pattern matching and folding of rnas. In Proceedings of BMC Bioinform, pages 1–14, 2014.
16. K. K. V. V. S. Reddy nad Chinta Someswara Rao S. Viswanadha Raju. Parallel string matching with linear array, butterfly and divide and conquer models. In Proceedings of Analysis of Data Science, pages 181–207, 2018.
17. MONTASSIR HADI PEYMAN NEAMATOLLAHI and MAHMOUD NAGHIBZADEH. Simple and efficient pattern matching algorithms for biological sequences. In Proceedings of IEEE Access, pages 23838—23846, 2020.
18. T. Gururaj and G. M. Siddesh. Hybrid approach for enhancing performance of genomic data for stream matching. In Proceedings of International Journal of Cognitive Informatics and Natural Intelligence, pages 1–18, 2021.