Khwaja Yunus Ali University Journal

Publisher homepage: www.kyau.edu.bd
OPEN ACCESS
ISSN: 2791-3759 (Online), 2521-3121 (Print)
Journal homepage:www.journal.kyau.edu.bd

Research Article

An Effective Sequential Hybrid Single Pattern Searching Approach for Bioinformatics

Prince Mahmud[1*], and Md. Moinuddin[2]

Department of Computer Science and Engineering, School of Engineering Sciences, Khwaja Yunus Ali University, Sirajganj, Bangladesh

***Corresponding Author**: m.princecse@gmail.com (Prince Mahmud, Lecturer, Department of Computer Science and Engineering, Khwaja Yunus Ali University, Sirajganj, Bangladesh)

Abstract:

Pattern searching is a significant part of computer science. There are huge pattern searching approaches that exist to solve the string matching problems. There are huge fields of pattern matching such as pattern finding in a specific text, plagiarism detection, intelligent machine, bioinformatics, and video retrieval. The main intention of pattern searching is to reduce the shifting and comparisons. For reducing those factors, we have introduced an EPSA (Effective Pattern Searching Approach) algorithm which is a sequential hybrid approach. We have assembled the good properties of Berri Ravindran, Quick search, and raita algorithm to construct our EPSA algorithm. We have used DNA and protein sequences to test the efficiency of our EPSA algorithm. The proposed EPSA algorithm displays an effective result compared with the Back and Forth matching and Maximum Shift algorithm, which reduces the number of shifting and comparisons.

Keywords: Hybrid, Exact matching, Bioinformatics, Single pattern, on-line approach.

1. Introduction

Data and Files are an important part of every institution. Finding the specific data among the huge amount of data is the most important challenge in every organization [1]. Finding a convenient pattern in a text is a simple process, but as file sizes grow, they can become highly time-consuming [2].In computer science, string searching is a classic and conventional problem that involves some techniques. String Searching is a type of string algorithm that finds the occurrence of a pattern in a text string, where strings are alphabetic character sequences (finite set) Σ [3]. Let's say the text is *x* of length *a* and the pattern is *y* of length *b*, with the pattern's length being less than or equal to the length of text (i.e., $a<=b$).String matching in biological sequences is among the first problems that computer scientists tackled. Even though DNA and proteins encode all information as nucleotide sequences with a short and simple alphabet, analyzing biological sequences is regarded as a traditional string matching problem [4, 5].

There are mainly two forms of string matching based on their accuracy: Exact and approximate string matching. Exact matching provides an accurate search

for the exact happening of the pattern inside the text, whereas approximate string matching provides for erroneous searching [6, 8].String matching can be partitioned into two types based on the number of patterns [7]. Throughout the text, a single pattern is explored in single pattern string matching [9]. Multiple patterns are explored within the text simultaneously in multiple pattern string matching [10]. The exact string match and single pattern match have been the main focus of this research.

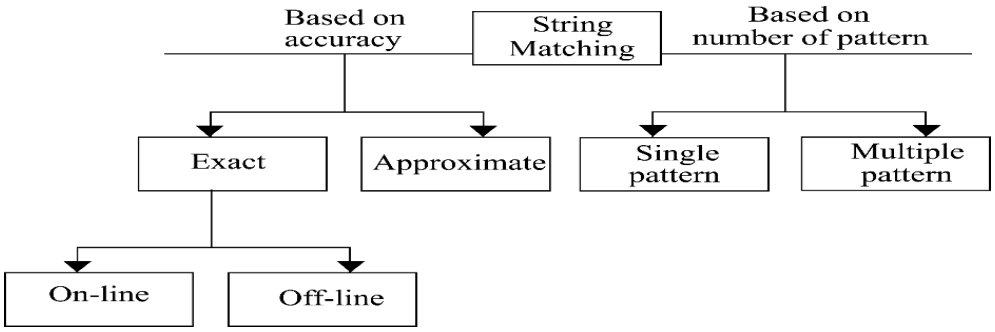


Fig 1.1: Forms of string matching

2. Related works

To address the string searching problem, a large number of algorithms employ their technique. Each algorithm uses its technique or a hybrid approach. Brute Force is most likely the first algorithm that comes to mind when considering how to solve the pattern-finding problem [11]. It has no preprocessing phase. This algorithm searches the pattern character by character and moves the pattern one location to the right when any match or mismatch occurs [12].The Boyer Moore (BM) method introduces three pattern-finding concepts: the bad character rule, the good suffix rule, and the right to left comparison [13]. This method is divided into 2 parts: preprocessing and searching [14]. Based on the pattern for the given alphabet, the pre-processing phase applies to the first two observations. The searching is related to the Brute Force scanning technique, but with the additional properties of aligning the pattern from left to right and attempting to compare characters from right to left order, which enhances the efficiency of that string matching algorithm [15]. Horspool algorithm is a modification of the Boyer Moore algorithm which uses only bad character rules of the BM algorithm [16].To identify a pattern in a text string, two phases are required: preprocessing and searching. The preprocessing phase is the same as the BM method using only the bad character table [17].The Quick Search (QS) algorithm is another variant of the BM algorithm [17, 18].The main difference between the Horspool algorithm and the quick search algorithm is shifting value calculation. The shifting value of the QS algorithm depends on the next character of the search window [19].The search window is compared to the pattern from left to right during the searching phase. The problem of the bad character rule (Small alphabet set) is improved by the good suffix rule. But the good suffix rule requires more comparisons and attempts for the longer alphabet. The Zhu-Takaoka (ZT) algorithm improves the BM algorithm's average case. The bad character shift function was improved by this algorithm. The core concept is concentrated on the preprocessing phase, which generates the preprocessed table using two characters rather than one in the original BM algorithm [20]. This algorithm's character comparison is performed from right to left order. Another adaptation of the BM method is Raita algorithm. The preprocessing phase same as the Horspool algorithm shifting technique and the searching phase is different [21]. This algorithm is experimentally efficient for searching the pattern in the English text. Berry-Ravindran's (BR) algorithm combines the best features of the QS and the modified of Zhu-Takaoka algorithm. Berry and Ravindran simply create a preprocessed moving function from the QS and Zhu-Takaoka functions. The pattern is started to shift during the searching phase based on the changing value of the two sequential characters of text beside the search window [17, 23].Maximum Shift

(MS) is an effective hybrid string matching algorithm [22]. They combine the features of the Quick Search, Zhu-Takaoka, and Horspool algorithms to reduce the number of shifting and comparisons. This algorithm outperforms the Quick Search, Smith, and Berry Rabindran algorithms in terms of efficiency. This algorithm is broken down into three stages: preprocessing, maximum, and searching. The preprocessing phase is integrated from the shifting function of QS and the Zhu-Takaoka algorithm. The maximum stage is concerned with the maximum value of two pre-calculated shifting functions. The searching phase is slightly different from the horspool algorithm. The Back and Forth Matching (BFM) algorithm is used the index-based technique [1]. This algorithm saves all first-position text where the pattern's initial and last character is the same as the initial and lattermost character of the search window. The pattern's upper-left and upper-right hand characters are checked with the initial and lattermost characters of the search window simultaneously during the searching phase.

3. Proposed algorithms

The fundamental purpose of this study is to develop an effective sequential hybrid algorithm by combining three existing techniques to solve the pattern matching issue. We blend the concept of Berry-Ravindran [17, 23], Quick Search [17, 18], and Raita [21] algorithm. The phases of our algorithm are as follows:

3.1Preprocessing Phase

For preprocessing, our algorithm uses the concept of Berry-Ravindran and the Quick Search algorithm. Our algorithm only preprocesses the pattern using those algorithms. First of all, our algorithm creates Berry-Ravindran bad character table using the following shifting function:

$$brBc[c,d] = min \begin{cases} 1 & if\ y[b-1] = c \\ b-i+1 & if\ y[i]y[i+1] = cd \\ b+1 & if\ y[0] = d \\ b+2 & otherwise \end{cases}$$

Here, y is the pattern, b is the length and i is the index of the pattern0 to b-1, c and d are two successive characters. Then, our algorithm creates the Quick Search bad character table using the following function:

$$qsBc(x) = \begin{cases} (i:\ 0 \le i < b\ and\ y[\,b-i\,] = x) & if\ x\ occurs\ in\ y \\ b+1 & otherwise \end{cases}$$

Here, b is the length and idenote the index of the pattern y from 0 to b-1. The character x defines each element in the text.

3.2 Searching Phase

The searching phase consists of two main stages: the maximum calculation and the searching technique. For shifting the pattern, our technique uses the largest value from Berry-Ravindran and Quick Search's bad character database. It is noted that the shifting position of the Berry-Ravindran bad character table depends on the preprocessed value of the two successive characters of the rightmost next to the search window and the shifting position of the Quick Search table depends on the preprocessed value of the rightmost next character of the search window.

Our proposed EPSA employs Raita algorithm techniques to find a pattern inside the text. This algorithm fits the pattern beneath the text first, and then compares the pattern's rightmost character to the search window's last character. If a match is found, then the first character of the pattern is compared to the search window's leftmost character. If a match is found, it compares the pattern's middle character and searches before actually comparing the others. The remaining characters are likened in order from left to right. If there is any mismatch, the pattern will be shifted by the maximum value from two tables calculated in the preprocessing phase.

Figure 3.1 shows the search technique of our proposed algorithm:

Let, a text x = ACGCGTAGTCAGTACGTAGCG and pattern y = CGTAGTC from the x.

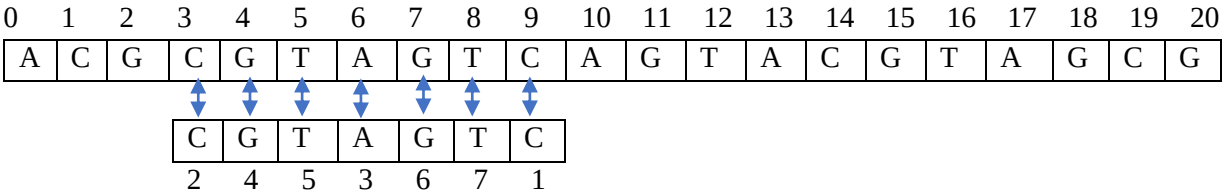


Fig 3.1: Search technique of our proposed EPSA algorithm

4. Working example

Helianthus annuus is a significant member of the Composite family. It is a key source of oil and a complementary source of protein for humans and livestock all over the world. We employed the Helianthus annuus cultivar XRQ/B chromosome 1 whole genome shotgun sequence to test the suggested EPSA method. According to the FASTA format, we selected a small portion of the gene's nucleotide sequence from index 85273 to 85297 [24].

The text of the DNA sequence to be considered, with the alphabet set Σ = {A, C, G, T} is

x= TGATCTGGCATGTACAGAATGAAAA and pattern y = TGTAC from the x.

Berry Ravindran bad character table for two consecutive characters a ,and b are shown in Table 4.1 for pattern p.

	A	C	G	T
A	7	2	7	6
C	1	1	1	1
G	7	7	7	4
T	3	7	5	6

Table 4.1: Berry Ravindran Bad Character Table

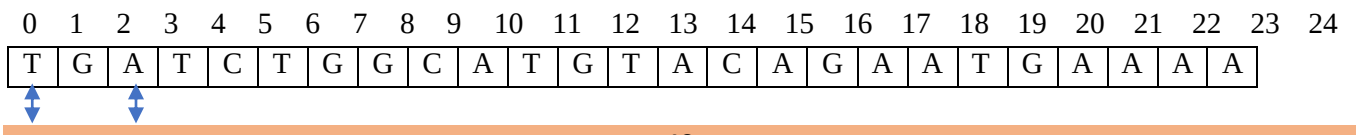
The Quick Search table is shown in table 4.2 for pattern p.

x	A	C	G	T
qsBc(x)	2	1	4	5

Table 4.2: Quick Search Table

For searching, firstly align the pattern in the first position within the text. The method uses the maximum shift value from two tables since the middle character is mismatched. Here, Berry Ravindran's and Quick search bad character table values are equal so this algorithm takes any one table value and shifts the pattern to text position 6. Figure 4.1 shows the searching procedure of our EPSA algorithm:

1st attempt:



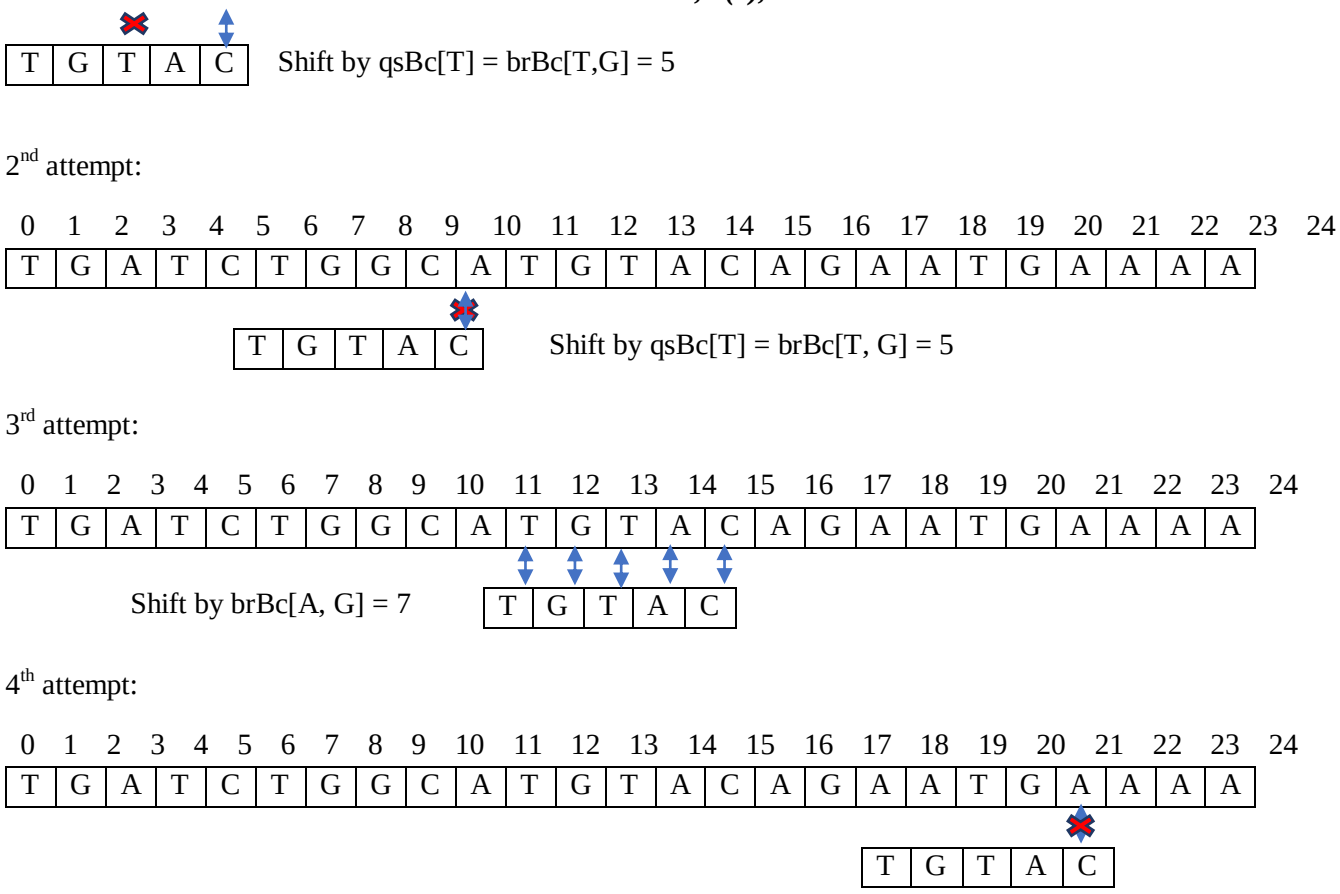


Fig 4.1: Searching procedure of our EPSA algorithm

The last character is mismatched in the second attempt, so this algorithm uses the highest value from two tables. Here, two tables' shifting value is also equal so takes anyone and shifts the pattern to text position 11. In the 3rd attempt, the pattern is matched within the search window of the text. After matching the pattern, this algorithm takes the maximum value from two tables for shifting the pattern. Here, Quick search bad character table value is 2 and Berry Ravindran's bad character table value is 7 so our algorithm takes Berry Ravindran's bad character table value and shifts the pattern to text position 18. In the 4th attempt, the lastposition of the pattern and search window is mismatched but there is no need to be shifted the pattern because the pattern exits the text. However, our algorithm needs 4 shifting and 10 comparisons for probing the pattern within the text.

5. Result analysis

Data analysis is the most important fact to discover and implement a new methodology or algorithm. We have used two types of data to appraise the performances of the EPSA algorithm. These are the DNA sequence and the Protein sequence. We have used one-hundred MB of all datasets to appraise our EPSA algorithm. All of the datasets are obtained from the "Pizza & Chili Corpus" website [25].

The number of shifting and comparisons are two important characteristics for evaluating the efficiency of the pattern matching approach. We chose pattern lengths of 3, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, 150, and 200 at random from the text for analyzing those two factors.

Figures 5.1 and 5.2 displays the number of shifting and comparisons respectively for the DNA sequence. This result shows that the MS algorithm is better for small pattern lengths such as pattern lengths 3, 5, etc., but the BFM algorithm is better for large pattern lengths. Our proposed EPSA algorithm shows the worst result for small patterns compared with the MS algorithm but shows better results for large pattern lengths compared with the BFM and MS algorithm. The worst outcome

is caused by character repetitions in a tiny alphabet set of DNA sequences.

Figures 5.3 and 5.4 display the number of shifting and comparisons respectively for the Protein sequence. This result shows that the performance of our proposed EPSA algorithm outperforms the BFM method for the full pattern set, whereas the BFM algorithm outperforms the MS algorithm.

Now if we merge the result of shifting from Figure 5.1 and the result of character comparisons from Figure 5.2, we get the following result shown in Figure 5.5 and merge the result of shifting from Figure 5.3 and the result of character comparisons from Figure 5.4, we get the following result shown in Figure 5.6. It is clearly seen that if the number of shifting is increased, the number of comparisons will increase for any length of the pattern. Moreover, the graph shows random points for both the number of attempts and comparisons. There is no continuous increment or decrement with the length of the pattern.

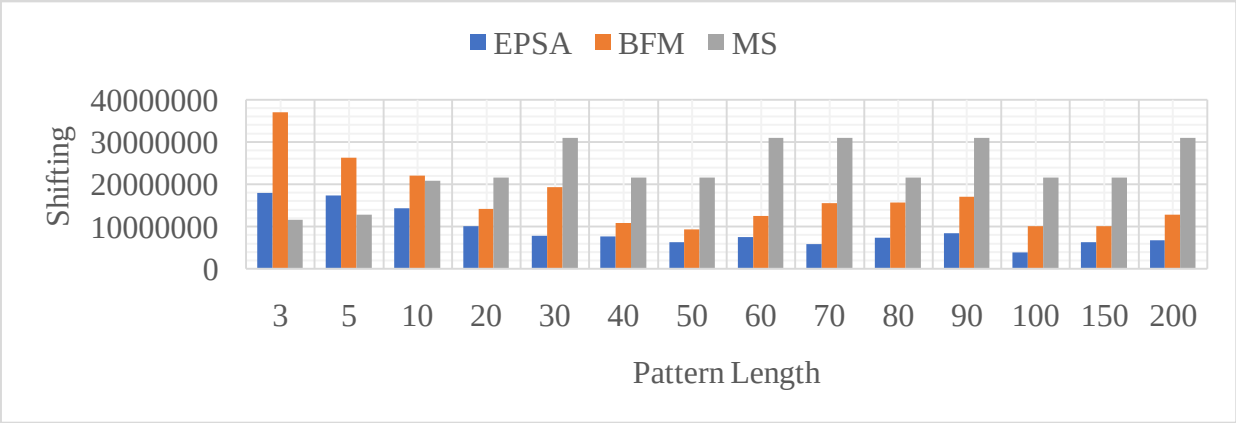


Fig. 5.1: Theshifting using DNA sequence

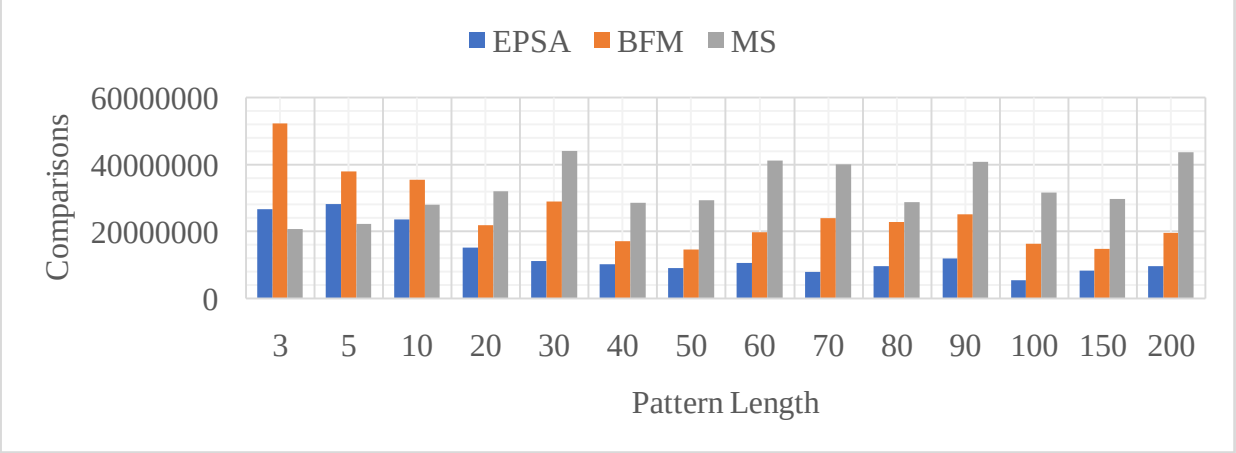


Fig. 5.2: The comparisons using DNA sequence

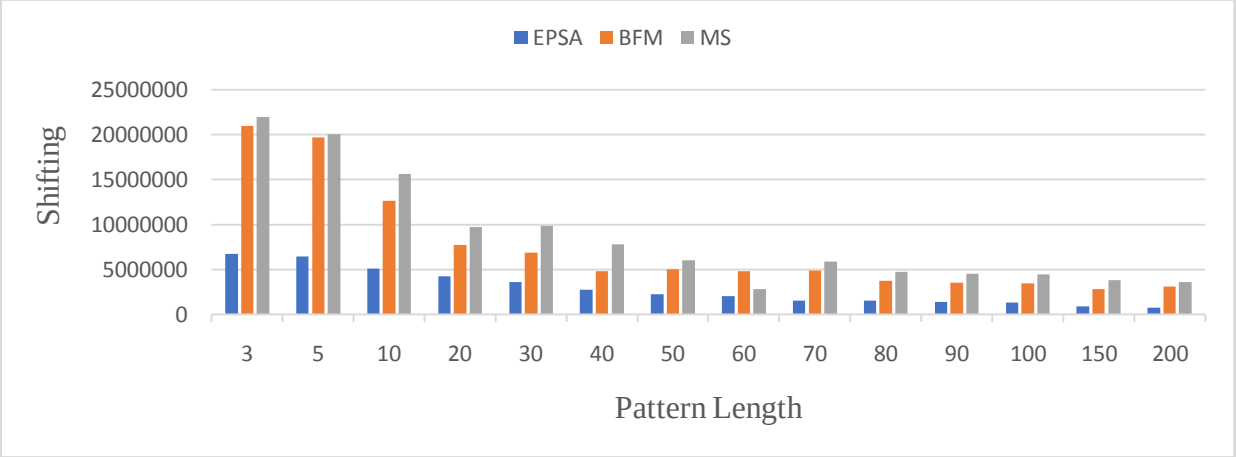


Fig. 5.3: The shifting using Protein sequence

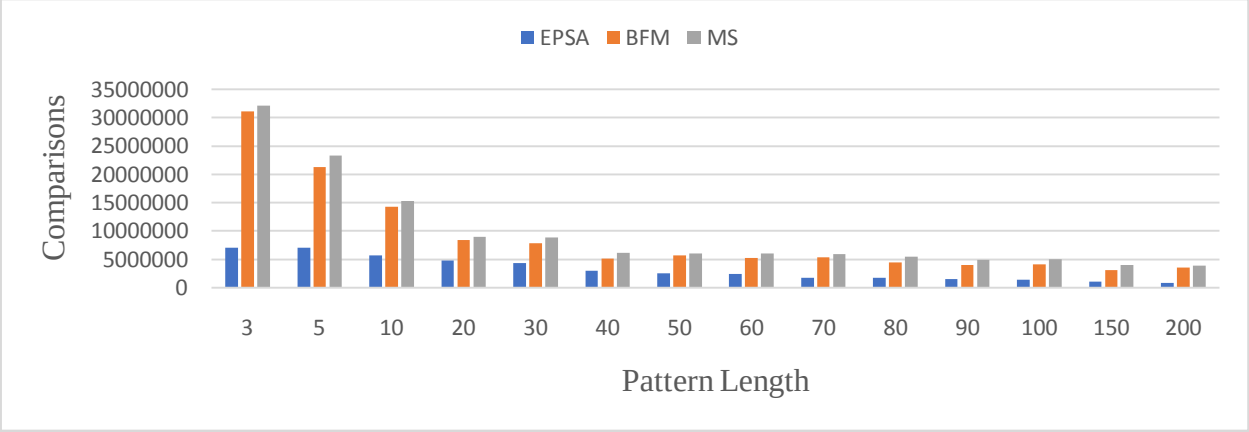


Fig. 5.4: The comparisons using Protein sequence

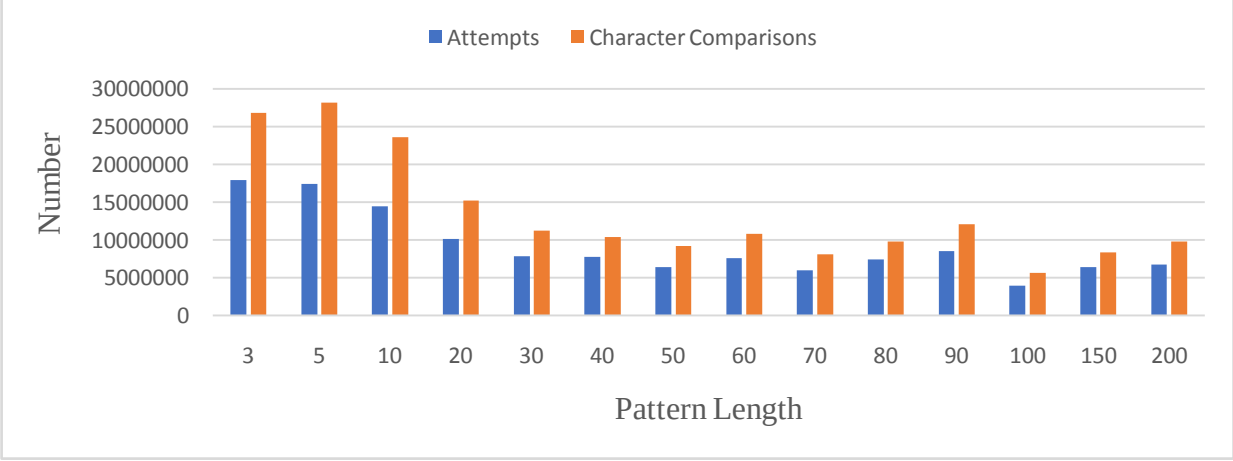


Fig. 5.5: Number of shifting and comparisons using DNA sequence

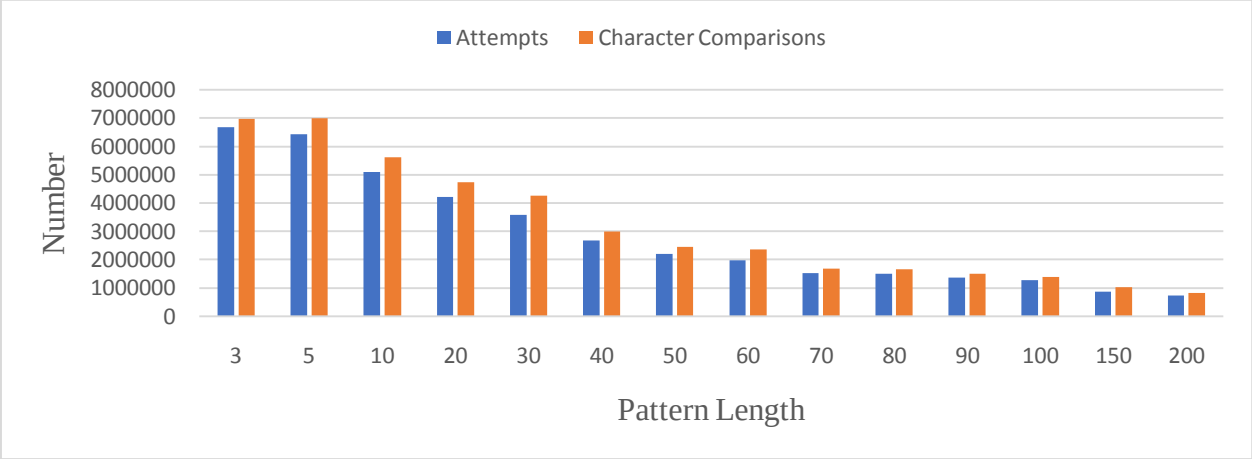


Fig. 5.6: Number of shifting and comparisons using Protein sequence

6. Conclusions

The development of numerous algorithms has resulted in a vast field of string matching. The number of time-saving string matching techniques is steadily increasing. The most prevalent motive of string matching is to reduce the shifting and comparisons which minimize the execution time. The proposed hybrid EPSA algorithm reduces the shifting and comparisons compared with the BFM and MS algorithm. As a result, this algorithm would undoubtedly be an efficient choice for tasks that necessitate a large number of text searches. One limitation of our algorithm is that the shifting and comparison increase when the alphabet set and pattern length are shorter.

7. Conflict of Interest:

All the authors in this research project hereby declare that there are no conflicts of interest.

8. Acknowledgement

First, we would like to express profound gratitude to the supreme of the universe the Almighty Allah. We would also like to express our deepest appreciation to all those who provided immense support and guidance for successfully completing this research work.

9. Funding of the Research

This work is funded by the Research Grant Committee (RGC), KhwajaYunus Ali University, Enayetpur, and Sirajganj, Bangladesh.

10. Authors Contributions

The concept of this present research was initiated by Mahmud P. All the authors participated in designing a questionnaire for the purpose of collecting and editing data. Thereafter, edited data was tested and analyzed with the cooperation of all authors. Finally, Mahmud P. took part in writing the manuscript and the rest of the authors approved the final manuscript after careful readings.

11. References

1. Al-Faruk, MD Obaidullah, et al. “BFM: a forward-backward string matching algorithm with improved shifting for information retrieval.” International Journal of Information Technology (2019): 1-5
2. Hakak, Saqib Iqbal, et al. “Exact String Matching Algorithms: Survey, Issues, and Future Research Directions.” IEEE Access 7 (2019): 69614-69637
3. Gurung, Dipendra, Udit Kr Chakraborty, and Pratikshya Sharma. “Intelligent predictive string search algorithm.” Procedia Computer Science 79 (2016): 161-169.
4. Xylogiannopoulos, Konstantinos F. “Exhaustive exact string matching: the analysis of the full human genome.” 2019 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM). IEEE, 2019.
5. Kalsi, Petri, HannuPeltola, and JormaTarhio. “Comparison of exact string matching algorithms for biological sequences.” International Conference on Bioinformatics Research and Development. Springer, Berlin, Heidelberg, 2008.

6. Hakak, Saqib Iqbal, et al. "Exact string matching algorithms: Survey, issues, and future research directions." IEEE Access 7 (2019): 69614-69637.
7. Markić, Ivan, Maja Štula, MarijaZorić, and DarkoStipaničev. "Entropy-Based Approach in Selection Exact String-Matching Algorithms." Entropy 23, no. 1 (2021): 31.
8. Rekha, J. UJWALA. "Approximate multiple string matching algorithm." Journal of Theoretical and Applied Information Technology 98.11 (2020).
9. Singla, Nimisha, and Deepak Garg. "String matching algorithms and their applicability in various applications." International journal of soft computing and engineering 1.6 (2012): 218-222.
10. Yang, Peng, et al. "Fast multi-pattern string matching algorithms based on q-grams bit-parallelism filter and hash." Proceedings of the 2012 International Conference on Information Technology and Software Engineering. Springer, Berlin, Heidelberg, 2013
11. Mohammad, Ababneh, Oqeili Saleh, and Rawan A. Abdeen. "Occurrences algorithm for string searching based on brute-force algorithm." Journal of Computer Science 2.1 (2006): 82-85.
12. Ibrahim, Amin MubarkAlamin, and Mustafa Elgili Mustafa. "Comparison criteria between matching algorithms texts application on (horspool's and brute force algorithms)." Journal of Advanced Computer Science & Technology 4.1 (2015): 175.
13. Boyer, Robert S., and J. Strother Moore. "A fast string searching algorithm" Communications of the ACM 20.10 (1977): 762-772
14. Ojugo, Arnold Adimabua, and David Ademola Oyemade. "Boyer Moore string-match framework for a hybrid short message service spam filtering technique." IAES International Journal of Artificial Intelligence 10.3 (2021): 519.
15. Al-Dabbagh, Sinan Sameer Mahmood, and NawafHazimBarnouti. "A New Efficient Hybrid String Matching Algorithm to Solve the Exact String Matching Problem." British Journal of Mathematics and Computer Science, pp. 1-14, 2017.
16. Horspool, R. Nigel. "Practical fast searching in strings." Software: Practice and Experience 10.6 (1980): 501-506.
17. Charras Christian, Thierry Lecroq. "Handbook of exact string matching algorithms." King's College; 2004.
18. Sunday, Daniel M. "A very fast substring search algorithm." Communications of the ACM 33.8 (1990): 132-142.
19. Al-Dabbagh, Sinan Sameer Mahmood, et al. "Parallel quick search algorithm for the exact string matching problem using OpenMP." Journal of Computer and Communications 4.13 (2016): 1-11
20. Feng, Zhu Rui, and Tadao Takaoka. "On improving the average case of the BoyerMoore string matching algorithm." Journal of Information Processing 10.3 (1987): 173- 177.
21. Raita, Timo. "Tuning the boyer- moore- horspool string searching algorithm." Software: Practice and Experience 22.10 (1992): 879-884.
22. Kadhim, HakemAdil, and NurAiniAbdulRashid. "Maximum-shift string matching algorithms." Computer and Information Sciences (ICCOINS), 2014 International Conference on. IEEE, 2014
23. *http://www-igm.univ-mlv.fr/~lecroq/string/berryravindran.html*,accessd February, 2022
24. (2022) National Center for Biotechnology Information. [Online]. Available: *https://www.ncbi.nlm.nih.gov/gene/?term= Helianthus annuus*
25. *http://pizzachili.dcc.uchile.cl/texts.html*,accessed February, 2022

Citation: Mahmud P, and Moinuddin M. (2022).An Effective Sequential Hybrid Single Pattern Searching Approach for Bioinformatics**.** *KYAU Journal.*5 (**1**):36-44