AN EFFICIENT SPACE REDUCTION TRANSFORMATION [SRT] TECHNIQUE FOR STRING MATCHING IN BIOLOGICAL SEQUENCES

[1]G.Kousalya, [2]Dr.Latha Parthiban
[1]Research Scholar, Department of Computer Science, [2] Research Guide,
[1]Bharathiyar University, Coimbatore
[1]Tamil Nadu, India

ABSTRACT:

DNA sequencing is done by stream matching techniques where String matching methods are often used to find out DNA pattern,which manages the investigating of various types of techniques for dissecting, storing and recovering natural information, for example, protein successions and nucleic corrosive (DNA/RNA), designs, capacities, pathways, and hereditary relations. In bioinformatics, genomic grouping examination has prompted the development of efficient example revelation calculations to deliver seek over expansive DNA sequences. The most relevant factor for efficiency of a matching algorithm is depending on the amount of character equivalence. The proposed Signal Assistance Transformation algorithm was reliable on different dimension of DNA sequences data set. The proposed algorithm shows great efficiency in terms of total amount of character equivalence it requires than the most existing algorithm. The experimental result shows that our proposed SAT algorithm works better when length of pattern is increased.

KEY WORDS: String matching, DNA Sequencing, SRT, Pattern Matching.

I.INTRODUCTION

A.STRING MATCHING:

String matching which is likewise characterize as string looking is found as the most vital and customary issue in the kfield of software engineering. Single or different examples are to be looked inside some random string. It has a key impact in different true issues and applications. **[1]** A couple of its rewarding applications are bio informatics, spam separating, infringement ID framework, computerized logical test, counterfeiting ID, spell checker and redress, web search tools, and data recuperation frameworks and so on **[1] [2]**. With the start of cutting edge grouping advancements, there are loads of genomic groupings of substances of similar species exhibited.

B.ANALYZING GENOMIC DATA

Under normal procedure every time a compressed sequence data is to be used, it has to be for processing. This process of decompressing and analyzing the compressed sequences reduces the efficiency of compressed system,**[3]**with additional overhead of computational space and time complexity. Hence it is expedient to analyze the compressed sequence for motifs, tandem repeats, and other patterns across a sequence as well as among sequences. Though a variety of works have been conducted in compressed pattern matching in textual data, nothing promising has been explored for genomic data.**[3]**Many authors have attempted to compress as well as search the compressed sequences using the methods devised for textual data. These attempts does not take the full advantage of the peculiarities of genomic data starting from the reduced character set of 4 instead of 26 in the case of textual data.**[3]** The genomic data overrides the possibility of numeric digits among the data characters as opposed to textual data.

C.DNA SEQUENCE:

The original PFAC was introduced as the general purpose library for any kind of applications, which are based on finding multiple patterns within the given large input. However, our implementation is designed especially for DNA sequence matching and compared with the usage of the original PFAC for DNA sequence matching. Within our PFAC implementation,**[4]** we concentrated on cache optimizations of GPGPU and application specific modifications for achieving better performance than original PFAC for DNA sequence matching.

II.LITERATURE SURVEY

Aamir et al propose an optimal algorithm for compressed pattern matching in text data. The compression technique used in run-length encoding and time complexity is O(c|T|) and a compressed pattern matching complexity is O(c|T|+ |P|). The algorithm works based on the periodicity class of a pattern under consideration. Periodicity of two dimension arrays is defined as the ability of an array to overlap itself without mismatch. Using witness tables **[3],[5]** candidate pattern is identified as compatible or not. The algorithm runs in two phases of pattern pre processing and text pre processing. A pattern pre processing takes an m x m pattern as input analyze it to identify pattern characteristic seam and produce a witness table. The text analysis phase is initiated with a candidate selection process which limits the initial number of candidates. In the second phase, identify the candidate as compatible or not. In the final verification step, the compatible candidate is chosen **[3],[5].**

Dimopoulos et al. [4] have discussed a modification to the serial Aho-Corasick algorithm for gaining memory efficiency to detect intrusions. They implemented a split Aho-Corasick algorithm with domain specific characteristics of intrusion detection for minimizing the memory usage of the finite state machine of the algorithm. As the domain specific characters, they observed that most patterns are subset of 256 characters, out of these 256 characters some are used almost in every states while other characters are used infrequently, and split finite state machines have smaller memory sizes according to the domain than the large finite state machine. The modified algorithm has been run on a Field-Programmable Gate Array (FPGA) chip as an improved version of the Aho-Corasick algorithm with the point of view of the memory usage.

The speed-up of Aho-Corasick pattern matching machines by rearranging states has been done by Nishimura et al. **[5].** They have rearranged the states of the finite state machines for improving the memory access via cache memory. Afterconstructing the basic goto graph, all the states of the graph have been rearranged according to the breadth-first order. Then all the memory accesses were more cache friendly and the performance gain is 55%.

The PFAC was introduced by Lin at el.[2]. PFAC has been run on a graphic processing units without failure links. They have run failure links less Aho-Corasick algorithm using thousand of threads of the GPGPU. Then it shows big performance improvement over the original serial Aho-Corasick algorithm. They published the second paper **[3]** of improving the PFAC by introducing a hash function method.

III.METHODOLOGY

A.STRING MATCHING:

String matching calculations can be partitioned into two kinds, single pattern matching and Multiple pattern matching. In single matching just a single pattern is to be looked into the given content though in the multiple pattern matching numerous patterns are given to be looked. String matching which is likewise characterizes as string looking is found as the most vital and customary issue in the k field of software engineering. Single or different examples are to be looked inside some random string. It has a key impact in different true issues and applications. **[1]**

B.PROPOSED METHODOLOGY

The proposed SRT technique generally evaluates with two main phases in string matching, they are as follows

- Pre-processing phase
- Penetrating Phase

(i)Pre-Processing phase:

Preprocessing stage is utilized to preprocess the pattern or given string which can limit the looking time. Here we preprocess the given string so as to limit the quantity of endeavors and number of character examinations. The proposed technique uses the index value of given string/DNA sequence for pre-processing and this phase has two stages.

a. Pre-process the given input string:

At first it tries to define all indices of the given string, then it works with DNA sequence and it only contains four characters **"{A, C, G, T}" [8][10].** For this purpose it is necessary to **[1]** maintain four array index tables. In our proposed SAT algorithm, the ASCII index system from **[8]** to define the subscript value of each character. The ASCII index system is measured by this formula, [((G[p])-64)%5]

Here, we consider four characters

X,Y,P,*Q*

ASCII values are provided correspondingly, Now the following table describes the ASCII value

Table.1.Subscript values of DNA characters

S.No	DNA Characters	ASCII Value	[((G[p])-64)%5]
1.	A	65	1
2.	C	67	3
3.	G	71	2
4.	T	84	0

The above formula provides the value"0,1,2,3,4" of the characters. This subscript value are using in the two dimensional array by assuming the string S be with length of x and Pattern be P with **[1]** length of y and store position of each character in the given text.

Value = Length (of the pattern) − Index(of character)

b. Compute the substring:

In second phase of pre-processing stage we need to partition the given info string into various number of substrings. The **[1]** length of substring is equivalent to the length of pattern and furthermore the primary character of substring is equivalent to the principal character of pattern. It ought to pursue the accompanying criteria:

- The substring distance end to end is exactly equal to pattern distance end to end.
- The process begins to figure substring from the content that matches with the patterns first character as we definitely realize the file estimation of content.
- Last substring index will generate according to the following.

Substring length = S[x − 1] − (P[y] − 1)

(ii)Penetrating Phase:

In SRT technique, the penetrating phase will follow the following step:

Here specifying SAT technique but this is kept from "An improved Algorithm for string matching using Index based shifting approach.

```
$M_c = S_{cp} * Sfs_c$
    $S_{cp} * Sfs_c = P$
//Matching begins in second character of pattern with the second character of first substring from left to right since the substring of first character and pattern first character always same.//
    if
        $M_c \neq P$
// does not match, the move to the third character of both pattern and substring//.
    Then
        $M = S_{cp}1, \ldots\ldots\ldots\ldots\ldots\ldots S_{cp}n$
// There is chance that there is any mismatch, it skirts matching that substring and example window goes to the following substring and proceed with the procedure //.
    if
        $M_c = P1$
    then
        Returns
        Lc
//If any match initiates, it returns to the location of that pattern in the given string//.
    If
        $S_{cp}$ is repeated

// There is a repetitions of pattern in the given string//
    Then
        $S_{cp} = p1 \ldots\ldots\ldots\ldots$
//when one matching is found it will continue its process until all substrings are matched//.
        $S_{cp} * Sfs_c = P = M_c$.

//when all substrings are matched//
    End.
```

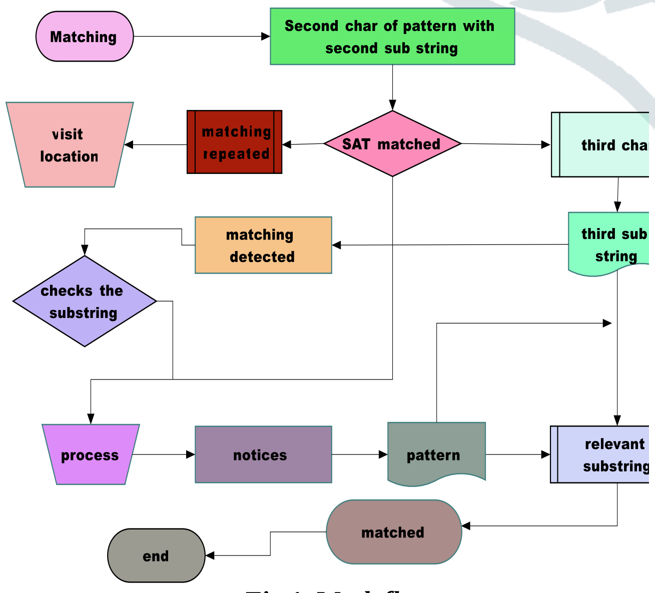


Fig.1. Work flow

B.SPACE REDUCTION ALGORITHM:

With the help of earliest pre-processing phase we solidly reduced the space of the search string. String matching which is likewise characterizes as string looking is found as the most vital and customary issue in the k field of software engineering. Single or different examples are to be looked inside some random string. It has a key impact in different true issues and applicationsWhen the string matching has been evaluated then there is need of reducing the space between the string and this process is seems here

```
$M_c = S_{cp} * Sfs_c$
    $S_{cp} * Sfs_c = P$
//Matching begins in second character of pattern with the second character of first substring from left to right since the substring of first character and pattern first character always same.//
    $N = K[S_{cp} * Sfs_c]$
// here N is numbers of sequence matched and K is the constant//
    If
        $N \neq K[P]$
// hence the number of sequence is not equal to pattern//
    $\sum_N = K[p]$
    $K[p] \ldots\ldots\ldots\ldots k[p]_n$
        $K[Q]$
// compressing the pattern on the basis of reducing the size till the n value and the compressed data is mentioned as Q//
    $I_m = IQ_1 \ldots\ldots\ldots\ldots IQ_n$
// initial match of the compressed data//
    $E_m = EQ_1 \ldots\ldots\ldots\ldots EQ_n$
// boundary match of the compressed data//
        $I_m \neq E_m$

// when initial match is not equal to the boundary match the reduce he space by the order of middle match and arrange the data in a sequence manner to notice its original size////
    $\sum_N = I_m \neq E_m$
    $K[Q] = IQ + EQ$
//now checking with the compressed size and the reduce space is noticed and data is found to be matched//
```

IV.RESULT AND DISCUSSIONS

A. DATASET:

This paper use only DNA sequences dataset **[9]** as our experimental data and this type involves nucleotides sequences with Adenine[X], Guanine[Y], Cytosine[P] and Thiamin[Q] **[10].** There are two basic highlights that are utilized to assess the execution of string coordinating calculations with various kinds of uses and these measures are recorded beneath. The proposed algorithm is developed using the language C# .

(i)Number of character correlations

This factor for the most part alludes to the genuine correlations that are happening between given the content and an example. The calculations with less character examinations are considered as better.

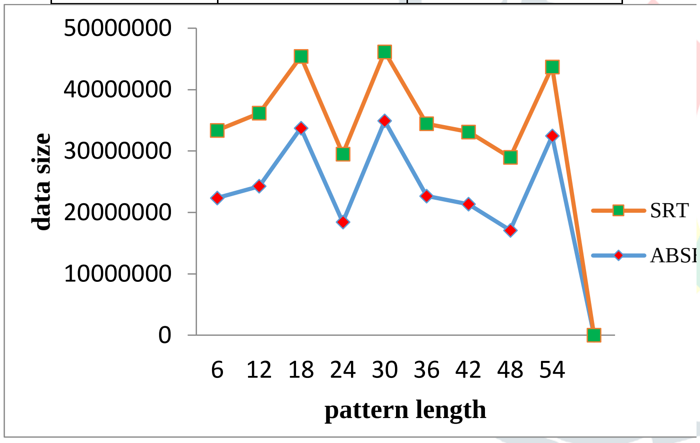
(ii)Number of evaluations:

This factor predominantly considers as the separation that an example moves alongside given string or content. Clearly, the execution of calculation is better when the quantity of endeavors is less.

The proposed SRT algorithm is compared with the ABSBMH algorithm **[9]** in terms a number of character comparison.

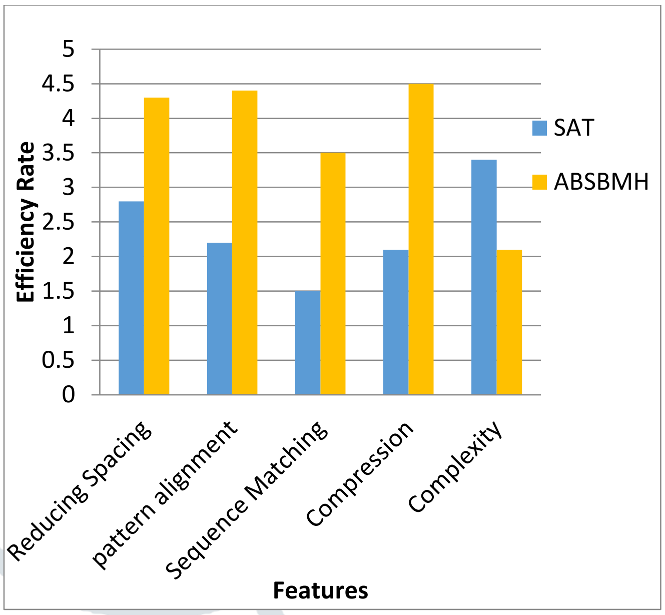
Table.1.Character Comparison using DNA sequences

Pattern Length	ABSBMH	SRT
6	22346789	11008707
12	24250524	11897651
18	33727763	11654732
24	18408266	11097368
30	34908765	11246801
36	22675489	11764931
42	21346844	11766543
48	17080181	11876503
54	32457123	11234598



Graph.1. Character comparison using DNA Sequences

The graph.1.discusses the character comparisons, Since the effectiveness is a quantifiable idea, quantitatively controlled by the proportion of valuable yield to add up to enter, among various matching process more and more efficiency is shown by the SAT and here it is compared with ABSBMH ALGORITHM and proved that it has the possibility to matching various large amount of data size.



Graph.2. Efficiency Comparison

The graph.2.discusses the efficiency analysis of sequencing matching, here we differentiate the basic function on pattern matching and the proposed SAT has the high efficiency in all the features and less complexity, When huge amount of data is implemented there is more difficult in reducing the spacing between the patterns and the proposed technique makes very simple thus it defines the compressed form among the data**.**

CONCLUSION

String matching is a very notable subject in bio informatics since it possesses a wide space in computer application. The proposed technique has exhibited another productive calculation which is known as SAT calculation. This calculation works quicker and gives a superior result when the quantity of character redundancies is restricted and length of the example is long. With the expanding of redundancies of character it likewise builds the quantity of substring. This calculation gives most exceedingly terrible outcome when utilizing English content and Protein grouping however gives better yield utilizing DNA sequences. This work is further extend in the perspective of reducing preprocessing time.

REFERENCES:

[1].Tania Islam,Kamrul Hasan Talukder (2017) “An Improved Algorithm for String Matching using Index Based Shifting Approach” *20th International Conference of Computer and Information Technology (ICCIT),* p.p:22-24.
[2]. V. SaiKrishna, A. Rasool and N. Khare,(2012).“String Matching and its Applications in Diversified Fields”, *IJCSI International Journal of Computer Science Issues, Vol. 9,* p.p:23-26,
[3].Keerthy A S,Dr. S. Manju Priya,(2017) “Pattern Matching in Compressed Genomic Sequence Data”2nd *International Conference on Communication and Electronics Systems (ICCES 2017)IEEE Xplore* ISBN:978-1-5090-5013-0.
[4].D.R.V.L.B Thambawita and Roshan G. Ragel and Dhammike Elkaduwe (2016) “An Optimized Parallel Failure-less Aho-Corasick Algorithm for DNA Sequence Matching”*IEEE Xplore* p.p:34-36.

[5] Amir, Amihood, Gary Benson, and Martin Farach. (1997)"Optimal two-dimensional compressed matching." *Journal of Algorithms* 24.2 p.p: 354-379.
[8] R. Bhukya and D. Somayajulu (2011), "An Index Based Sequential Multiple Pattern Matching Algorithm Using Least Count, International Conference on Life Science and Technology, *IACSIT Press*, Singapore, vol.3,.
[9].S. S. Mahmood, A. Dabbagh1, and N. H. Barnouti,(2017) "A New Efficient Hybrid String Matching Algorithm to Solve the Exact String Matching Problem, *British Journal of Mathematics and Computer Science*, pp. 1-14,
[10] Md. Aashikur Rahman Azim, Costas S. Iliopoulos, M. Sohel Rahman, and M. Samiruzzaman.(2015) Simplificpm: A simple and lightweight filterbased algorithm for circular pattern matching. *International Journal of Genomics, vol. 2015, pages 10, Article* ID 259320,
[11] M. Lothaire.(2005) Applied Combinatorics on Words. New York, NY, USA: Cambridge University Press, [12] K Fredriksson and S Grabowski. Average-optimal string matching. J Discrete Algorithms, 7(4):579–594, 2009.
[13] Kuei-Hao Chen, Guan-Shieng Huang, and Richard Chia-Tung Lee.(2014) Bitparallel algorithms for exact circular string matching. Comput. J., 57(5):731–743, 2014.
[14] Md. Aashikur Rahman Azim, Costas S. Iliopoulos, M. Sohel Rahman, and M. Samiruzzaman.(2014) A fast and lightweight filter-based algorithm for circular pattern matching. ACM Conference on Bioinformatics, Computational Biology, and Health Informatics,
[15] Md. Aashikur Rahman Azim, Costas S. Iliopoulos, M. Sohel Rahman, and M. Samiruzzaman(2015). A filter-based approach for approximate circular pattern matching. Bioinformatics Research and Applications, *ISBRA* 2015, Norfolk, VA, USA, June 7-10, 2015 Proceedings, pages 24–35,
[16] Carl Barton, Costas Iliopoulos, and Solon Pissis(2014). Fast algorithms for approximate circular string matching. Algorithms for Molecular Biology, 9(1):9,
[17] http://www.inf.kcl.ac.uk/research/projects/asmf/.
[18] CS Iliopoulos and MS Rahman.(2008) Indexing circular patterns. Proceedings of the *2nd International Conference on Algorithms and Computation*, pages 46–57,
[19] J Lin and D Adjeroh(2012). All-against-all circular pattern matching. Comput J, 55(7):897–906,
[20] S. M. Smith and J. M. Brady,(2015) "SUSAN - a new approach to low level image processing", International Journal of Computer Vision, 23(1): 45-78,