Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 4, Issue. 8, August 2015, pg.369 – 374

RESEARCH ARTICLE

An Optimized Algorithm for k Length Frequent Pattern Search over DNA Sequence

Vandna Gureja[1], Nidhi Sharma[2], Alok Sharma[3]

M.Tech Scholar, Computer Engineering, MDU, India
Assistant Professor in CSE Dept, MDU, India
Assistant Professor in CSE Dept, MDU, India
vandnagureja@gmail.com, nidhisharma1725@gmail.com, malok1231@rediffmail.com

Abstract-Genomic sequence analysis in bioinformatics has lead to the evolution of efficient pattern discovery algorithms to generate search over large DNA sequences. This paper proposes a tabular structure based k length Frequent Pattern (FP) extraction using side by side removal techniques applicable at Non Frequent Patterns (NFP).The proposed model consists of two phases, in the first phase the 2 Dimensional vector is formed as the result of preprocessing over the DNA Sequence based on the next character information and in the second phase pattern extraction of pattern is performed by parallel removing the spurious patterns. The direct mapping of DNA Sequence in table form has reduced the complexity of the combined process.

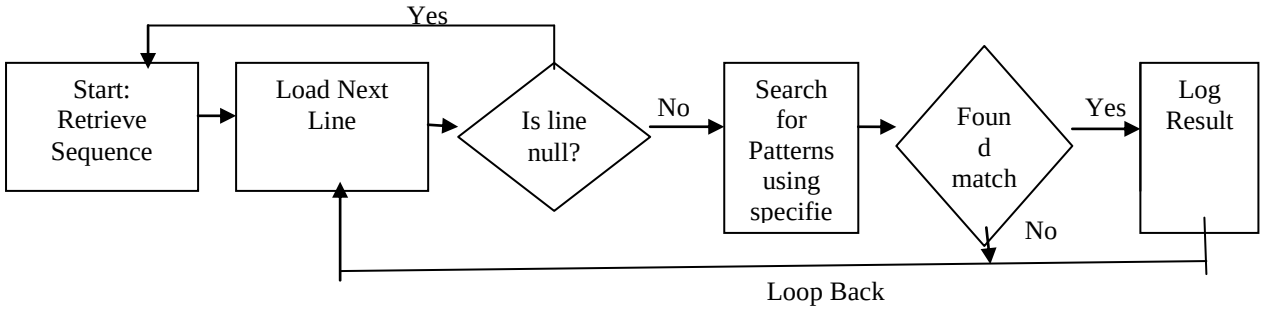
Index Term: Frequent Pattern, Sequence, Mapping, Spurious cells, complexity, Frequency

I. INTRODUCTION

String Matching or Pattern Searching is a technique to discover a pattern P with length m, the any given sequence T with length n and storing the frequency and shift of the pattern occurrences. The Pattern searching process is generally divided into two categories (i) Exact Pattern Searching (EPS): in which the exact occurrences of Pattern are searched only giving the output as successful search or unsuccessful search while in second category (ii) Approximate Pattern Searching (APS): is the process in which the result is said to be successful even if there is 80% matching is done. For example Let U is an alphabet; the basics of U are called symbols or characters. For example, if U= {A,G} then AGAG is a string. The pattern is denoted by P (1…M) the string denoted by T (1…N) [1].The pattern occurs in the string with the shifting operation.

DNA Sequence analysis has been an important research area since last few decades in the field of bioinformatics. Evaluation of similarity between the sequences for disease identification, gathering hereditary information, bio molecular processing and genomic sciences is now a crucial field in Medicine. These methods are based on alignment, word frequency and geometric representation respectively, each of which has its advantage and disadvantage [2]. The search process on any DNA string can be further divided into 3 categories: (i) when the sequence is given and the pattern is given (ii) when the sequence is given and the length of pattern is given.(iii) when the sequence is given and no information about the pattern has provided.

Figure1. Flowchart of a general pattern match



In this paper a working model is proposed which consists of two phases, in the first phase the 2 Dimensional vector is formed as the result of preprocessing over the DNA Sequence based on the next character information and in the second phase pattern extraction of pattern is performed by parallel removing the spurious patterns. The mapping of DNA Sequence in table form has reduced the complexity of the combined process.

The rest of the paper is organized as follows: The II section describes the concepts and definitions associated with the work, the III and IV section contains the detailed working of the proposed algorithm, the V section explains the working example of algorithm, and the VI, VII section has Results & Analysis, Conclusions respectively.

II. CONCEPTS AND DEFINITIONS

Let Σ = {A, C, G, T} be a set of DNA alphabets, where A, C, G, and T are called DNA characters or four bases; A is for adenine, C for cytosine, G for guanine, and T for thiamine. A DNA sequence S is an ordered list of DNA alphabets. S is denoted by <c1, c2… cl> where ci ∈ Σ and │S│ denotes the length of sequence S sequence with length n is called an n-sequence.

Sequence-it is the large amount of data submitted to the algorithm as input in which the patterns of different size are to be searched. Length of sequence is 1 to n.

Pattern- it is the subset of sequence which is to be searched , basically it is output of any algorithm with the possible shift occurred in the sequence. Length of pattern is 1 to k. Generally k is always less than m.

PHASE 1 : TABLE FORMATION

DNA sequence is a large symbol sequence formed using DNA codes. To perform the knowledge extraction over such larger sequence is a challenging task. There are different form of pattern identification and knowledge extraction. To perform this extraction, an effective pattern search algorithm is suggested in this work. In this work a structured form is presented in which the complete DNA input string is at first transformed to the table structure. This structure generation is the primary task applied only once. This structure will provide the direct and relational awareness to the DNA characters based on the pattern. This prior knowledge of next character existence provides the direct jump over the table to map the DNA pattern. This direct mapping provides the efficient tracking of DNA pattern over the DNA string.

This work model is applied only one time as the pattern search begins. The complexity of this algorithm model requires processing each character of the sequence at least once and later on identifying the position in the maximum mapped character of the sequence. The Table formation process is described in table 1 given below.

Table 1. 2 Dimensional formation algorithm of DNA sequence

```
TableStructureGen(dnaseq)
/*dnaseq is the actual sequence on which the dna pattern is required to search*/
{
1.      dnacharfreq=GetFreq(dnaseq,dnachars) // [Obtain the Frequency of each DNA character over the sequence]
2.      size=Max(dnacharfreq) //[Obtain the maximum length of any of DNA characters]
3.      Generate 2D structure of size (4,size) //[The rows here represents the dna characters and columns represents the
        length of the maximum frequency character]
4.      For i=1 to length(dnaseq) //[Process the sequence characters]
        {
5.      char=dnaseq(i)
        nextchar=dnaseq(i+1)//[Obtain the character and next character from DNA sequence]
6.      [freq,pos]=GetFreq(nextchar) //[Get the frequency and position of next DNA character in the sequence]
7.      Len=GetLength (structure (char, :)) //[Get the length of character in the structure]
8.      structure(char,length+1)=nextchar
        structure(char,length+2)=freq
```



```
structure(char,length+3)=pos
[Store the pattern component in the structure]
}
Return structure; }
```

III. PHASE 2: PATTERN EXTRACTION AND SPURIOUS PATTERN REMOVAL

After forming this structure, the pattern search is applied. This pattern search is applied on table structure itself without involvement of raw DNA sequence. The search algorithm has separated the work in terms of single character, two characters and multiple character searches. In case of single character search, the column length of particular DNA code row is taken. This length represents the frequency of pattern character count. To perform, two character pattern search, the frequency of second character in particular DNA character row is identified. This frequency represents length of occurrence of DNA character pair. To perform the multiple character search, the column based switching is performed for each character.

The following table represents the k length frequent pattern search algorithm applicable on the structure formed after phase I. Every unique pattern encountered is added to the UniquePatternList and every repeated pattern leads to increase in the frequency count of that pattern. At last the maximum repeated pattern of length k is obtained with the frequency and shifts. The maxfreq is the temporary storage in which the frequency of maximum repeated pattern is stored any pattern having freq<maxfreq will be counted as spurious pattern and will not be stored. This search algorithm is shown in table 2 .

Table 2. The Pattern generation algorithm from 2D structure formed by phase I

```
TableStructureGen(dnaseq,k)
/*dnaseq is the actual sequence on which on which the automated form of maximum pattern identification is done.*/
{
1.      For len=k //[Check for different patterns of  size=k]
{
2.      For i=1 to dnaseq.length-len+1 //[Process the pattern extraction of given size on dna sequence]
{
3.      pattern=ExtractPattern(dnaseq,i,len) //[Extract the pattern from the sequence]
4.      count=CountExistence(dnaseq,pattern) //[Check if the pattern exist over the dna sequence]
5.      if (count>1) //[If pattern exist more than once]
{
6.      if(uniquePattern.Exist(pattern)=0) //[Check the pattern is not already recorded]
{
7.      uniquePattern.Add(pattern) //[Include the pattern in the list]
8.      maxpattern=FindMaxLenth(uniquePattern) //[Identify max size pattern]
9.      returnmaxpattern; }
```

IV. WORKING of ALGORITHM WITH EXAMPLE

The working of the proposed model can be explained through given example as :

Let the DNA SEQUENCE: CCATCGTTTAATCGATCG

Length of pattern, k=4

Table 3. A random sequence of DNA alphabets

Pos:	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Seq:	C	C	A	T	C	G	T	T	T	A	A	T	C	G	A	T	C	G
Freq:	1	2	1	1	3	1	2	3	4	2	3	5	4	2	4	6	5	3

Maximum frequency= 6

A.Phase I

i. Table formation:

In the first phase the DNA sequence is converted into a tabular structure based on the next character information. Here two variables are created namely CHAR and NXTCHAR. For example the 1st char of DNA sequence is CHAR=C with pos=1 and freq=1 and the NXTCHAR =C freq=2. The Table below formed can be read as: At the 1st occurrence of (CHAR=A) its position in the sequence is 3rd with NXTCHAR=T & freq of NXTCHAR=1 or [A,1= 3(T,1)]

ii. Searching Process:

Suppose the pattern to be searched P= CAT len=3

The control flows as→ in the row where CHAR=C , NXTCHAR=2(A ,1) lies in the 2nd occurrence of C at position 2. Now CHAR=A and NXTCHAR=T , we have to check whether the 1st occurrence of A is followed by T or not. So at the 1st occurrence of A A,1=3(T,1). If there would have been C,A or G in place of T in the table the pattern is stated as not occurred at the 2nd occurrence of C. Now we have to check the 3rd ,4th 5th occurrence of C , fortunately the A is not the next character for any of these occurrences of C, So we don't need to check further. There by reducing the time of processing.

Table 4 . The 2D structure formation during phase I

	1	2	3	4	5	6
A	3(T,1)	10(A,3)	11(T,5)	15(T,6)		
→C	1(C,2)	2(A,1)	5(G,1)	13(G,2)	17(G,3)	
T	4(C,3)	7(T,3)	8(T,4)	9(A,2)	12(C,4)	16(C,5)
G	6(T,2)	14(A,4)	18(X,X)			

B.Phase II

i. Pattern Generation and Spurious Pattern removal:

In this phase a UniquePatternList is generated which generates all the possible patterns of length k.

For Example the 1st pattern of len=4 is CCAT, now the searching process is applied at the whole sequence to check whether there is any repetition of pattern or not if not then the pattern is added to the List and maxfreq=1 and position=shift of the pattern. The next pattern searched would be CATC and then ATCG, since it has the freq=3 then the maxfreq=3 and all the pattern having freq<maxfreq will be counted as spurious patterns and will be removed. No new pattern will be added to the list until it has freq>=maxfreq. In the below table all the patterns generated are shown without removal of spurious patterns for the sake of convenience.

Table5. UniquePatternList formed during phase II with position and frequency

S.NO	PATTERN	POSITION	MAXFREQUENCY
1.	CCAT	1	1
2.	CATC	2	1
3.	ATCG	3,11,15	3
4.	TCGT	4	1
5.	CGTT	5	1
6.	GTTT	6	1
7.	TTTA	7	1
8.	TTAA	8	1
9.	TAAT	9	1
10.	AATC	10	1
11.	TCGA	12	1
12.	CGAT	13	1
13.	GATC	14	1

V. RESULTS & ANALYSIS

The results obtained can be summarized in table 7 after implementing the proposed algorithm

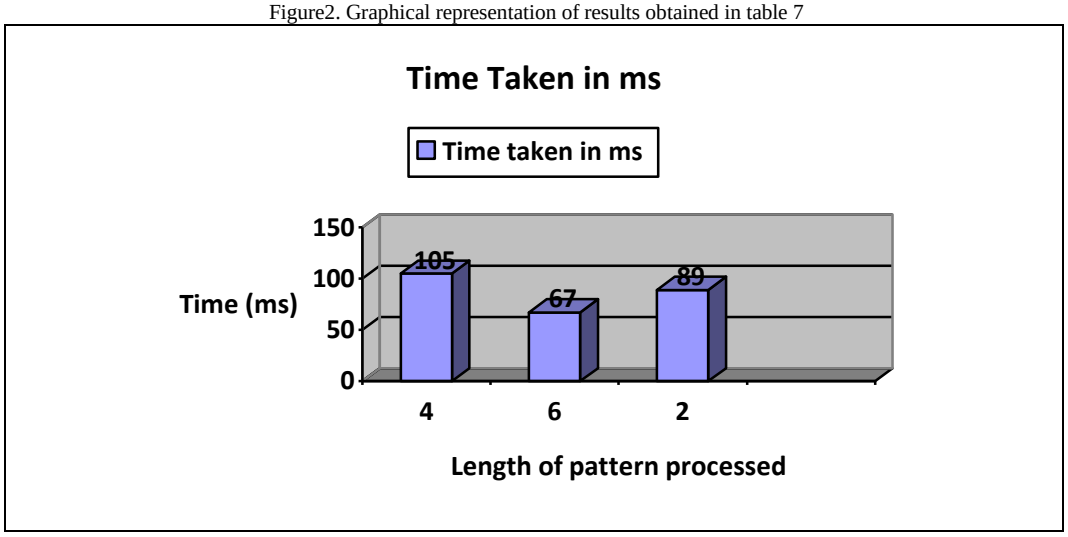
Text Sequence= CCATCGTTTAATCGATCG

Table 7. Results obtained after implementing the algorithm

Pattern submitted	ATCG	TTTAAT	AT
Length of pattern	4	6	2
No. of occurrences	3	1	3
Position of occurrences	3,11,15	7	3,11,15
Number of character comparison	12	6	7
Time Taken (ms)	105	67	89

In table 7 above it is seen that the number of comparison per pattern match is drastically reduced or it can be said that the number of character comparisons can be equal to (length of pattern to be searched * times of occurrences) in the best case.

Figure2. Graphical representation of results obtained in table 7



The complexity of this algorithm is given by O (nm). Here n is the length of the DNA sequence and m is the length of maximum occurred DNA character or the column length of table. But this complexity will be included to the overall algorithmic complexity only once because this process model will be applied only once during the structure formation. After that only the structure traversing will be performed.

If the length of search pattern is k then the complexity of the algorithm will be O(m+k). Here m is the column length of generated structure. As the direct positional map is applied in this work, it provided the effective DNA character sequence search. The presented work model is applied on different sequence and pattern sets. The complexity analysis of the algorithm is given in the table 8 below:

Table8. Complexity comparison of the given algorithm with the existing algorithms

S. no	Algorithm	Comparison Order	Pre processing complexity	Matching complexity	Characterstics
1	Naïve	Not relevant	No preprocessing	O(mn)	It uses single character shift.
2	Rabin Karp	Left to right	O(m)	O(mn)	It uses Hashing function, useful in multiple pattern matching.
3	Knuth-Morris-Pratt(KMP)	Left to right	O(m)	O(m+n)	It is independent of alphabet size, knowledge of previous mismatch for further matching increases performance.
4	Boyer Moore	Right to Left	O(m+n)	O(mn)	It is based on 2 rules • Good Suffix • Bad character shift
5	Needleman Wunsch	Left to right	-	O(mn)	Global alignment of protein and nucleotides
6	Smith-Waterman	Left to right	-	O(mn)	Align local sequence in segments of proteins and nucleotides
7	Proposed	Top down or bottom up	O(n)	(m+k)*	Based on column-switching

*m represents maximum frequency of any character in the DNA sequence to be processed.
*k represents the size of pattern to be searched.

VI. CONCLUSION

In this paper, a two phase structure adaptive model is presented for improving the DNA pattern search for pattern of defined length k. This work model is divided in two main stages. In first stage, the tabular structure is formed over the DNA sequence. In second stage, the search is applied over the structure and generation of patterns. The experimentation is applied on different size DNA sequences and different patterns. The results show that the presented work model has provided the search in effective time. This work model can be improved in future for identification of relative behavior

search to perform the classification and for searching the maximal frequent pattern over the DNA sequence. The characteristic oriented classification or disease oriented classification can be applied using this proposed model.

REFERENCES

[1] Pandiselvam.P, Marimuthu.T and Lawrance. R, "A COMPARATIVE STUDY ON STRING MATCHING ALGORITHMS OF BIOLOGICAL SEQUENCES", International Journal of Soft Computing and Engineering, ISSN: 2231-2307, Volume-I, Issue-6, January 2012

[2] Md. Rezaul Karim, Md. Mamunur Rashid, Byeong-Soo Jeong and Ho-Jin Choi, "An Efficient Approach to Mining Maximal Contiguous Frequent Patterns from Large DNA Sequence Databases", Genomics & Informatics Vol. 10(1) 51-57, March 2012

[3] Xiaojing Xie, Jihong Guan, and Shuigeng Zhou , "Similarity evaluation of DNA sequences based on frequent patterns and entropy", 10th International Symposium on Bioinformatics Research and ISBRA-14) Zhangjiajie, China. 28-30 June 2014

[4] Ms. V. V. Kamble and Mr. S. V. Kamble , "Discovery of Frequently Occurring Approximate sub sequences with distance" , International Journal of Scientific and Research Publications, Volume 5, Issue 5, May 2015

[5] S.Nirmala Devi , Dr.S.P Rajagopalan and Dr.V.Anuradha ,"Index Based Multiple Pattern Matching Algorithm Using Frequent Character Count in Patterns", International Journal of Advanced Research in Computer Science and Software Engineering Volume 3, Issue 5, May 2013 ISSN: 2277 128X

[6] Thomas H. cormen ,Charles E. Leiserson and Ronald L. Rivest,"Introduction to algorithms , Eastern Economy Edition2nd Edition ,Prentice-Hall of India Publications, 2001

[7] Sartaj K. Sahni and Ellis Horowitz," Analysis and Design of Algorithms, " 2nd Edition, 2001

[8] Nimisha Singla andDeepak Garg, "String Matching Algorithms and their Applicability in various Applications", International Journal of Soft Computing and Engineering (IJSCE) ISSN: 2231-2307, Volume-I, Issue-6, January 2012

