International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622
International Conference on Industrial Automation and Computing (ICIAC- 12-13th April 2014)

RESEARCH ARTICLE OPEN ACCESS

Mining Elementary Repeats from DNA using Suffix Tree in Linear Time

Vishal B. Rathod *,Prof.A.V.Deorankar, Dr. P.N.Chatur
*,***(Department of
** (Department of Information Technology, Government College Of Engineering, Amravati, India

ABSTRACT-
In genome study identifying the elementary repeats in DNA sequence is typical problem. There are various approaches designed to find Elementary Repeats and boundaries of elementary repeats in the given sequence. As genome data use to be very large , we require fast and efficient algorithm to identify such repeats. This paper presents the Ele(mentary)Rep(eats) algorithm which identify the repeats from sequence in linear time. We use the definition of the Elementary repeat given by Zheng and Stefano . EleRep uses modified suffix tree to store required information in suffix tree node. EleRep avoids the time consuming process of checking occurrence list required in RepSeeker algorithm. Thus, enhances the performance of EleRep. EleRep provieds the starting and ending of the occurrences of the repeats in given DNA sequence.
Keywords - Elementary repeat ; EleRep; Suffix tree; Ukkenon'algorithm;

I. INTRODUCTION

DNA is well known for carrying the genetic information of living species and their information encoding . Study of DNA sequence is required to recognize different genetic properties and thee genetic diseases. In genome study , as data of DNA sequences is increasing it becomes challenging to handle such huge data and extract information out DNA sequences. To extract the information from DNA we identify the repeats. DNA sequence composed of more number of repeats. For example genome of humane has more than 50% various kinds of repeats [8]. It required to identify repeats in sequence. Genome study shows that many genetic diseases are caused because the irregularity in the repeats length, such as Fragile – X chromosome syndrome. Some studies shows that, in gene expression some repeats are important. It is required to study those repeats for understanding their function. There are many repeats which are not well defined and functions of those are not clearly understood. Identifying such repeats becomes challenging problem.

Sequential data mining is having importance in many field such as genome study. There are various algorithm sequential pattern mining which include mining the pattern with constraints .Applying these algorithms where continuous pattern mining is required, is not appropriate such as in DNA sequence mining for finding repeats . As size of data on which these algorithm applied is very large. For example, in human genomes there are 3 billion DNA base pairs. These genome contains most of the repeats. Therefore these repeats can be considered as basic block of sequence. Hence, first we require to identify repeats.We require method to identify repeats in DNA Sequence fast and efficiently. RepeatMasker and Reputer are two well known approaches to recognize repeats .RepeatMasker has library of repeats and uses the library to identify repeats . If library of regions is not known then this approach don't workfor new repeats [5]. REPuter identifies the maximal length repeats. Frequencies of occurrences of the repeats is not considered din the both repeats [6]. Definition of Elementary repeats was proposed Zheng and Stefano [2] which different from finding the maximal repeat. They proposed algorithm with time complexity O(n2)where n is length of sequence. This problem was studied by Dan He [3] .The algorithm proposed in [3] to find the elementary repeat is having time complexity O(n2/f2) where n is sequence length and f is the minimum frequency .For finding elementary repeats RepSeeker algorithm [1] designed which consider the frequencies and also provide approach to extend and classify repeats. RepSeeeker checks occurrence list of the repeats which is time consuming process .RepSeeker has frequency O(nNf) where N is number of different repeats ,n is sequence length and f is average frequency of occurrence.

In this paper we present , EleRep algorithm for mining the DNA sequence. We consider the problem of mining the Elementary Repeats from sequences .We consider the elementary repeats definition defined by Zheng and Stefano. We use suffix tree constructed using Ukkonen's algorithm.

This suffix tree is modified ,such that at each suffix tree node we store the additional information required for extracting Elementary

Repeats. We extract only possible nodes which may be the Elementary repeats . We perform operation on only extracted possible nodes .For this we use suffix links which are added to the node while constructing suffix tree using Ukkonen's algorithm. This enhances the performance of EleRep. Our algorithm EleRep correctly extracts the elementary Repeats in O(n) where n is sequence length. With this, our algorithm can recognize the boundaries of repeats where the occur in the DNA sequence. Therefore this algorithms is fast,efficient and practical for mining the elementary repeats in DNA sequences where each sequence is having long length.

II. RELATED WORK

A.Finding Sequential Patterns

It is the problem in database mining which is motivated by the decision support problem faced by most large retail organizations[3]. An example of such pattern is customer is going for shopping of computer first then printer and there after camera within period of two months. So the problem is "what items are bought together in a transaction".While related, the problem of finding what items are bought together is concerned with finding intra-transaction patterns, whereas the problem of finding sequential patterns is concerned with inter-transaction patterns[3]. A pattern in the first problem consists of an unordered set of items whereas a pattern in the latter case is an ordered list of sets of items. The problem of finding all sequential patterns is solved in five phases: i) sort phase, ii) litemset phase, iii) transformation phase, iv) sequence phase, and v) maximal phase[3].

B. Finding Frequent closed itemset:

Most of the previously developed sequential pattern mining methods follow the methodology of Apriori which may substantially reduce the number of combinations to be examined[1]. However Apriori still encounters problems when a sequence database is large and when sequential patterns to be mined are numerous and long. A novel sequential pattern mining method,called Prefixspan (i.e., Prefix-projected-Sequential Pattern mining), which explores prefix projection in sequential pattern mining[3]. Prefixspan mines the complete set of patterns but greatly reduces the efforts of candidate subsequence generation. Moreover; prefix-projection substantially reduces the size of projected databases and leads to efficient processing.[1].

C. Ukkonen's Edit Distance Calculating Algorithm:

EDIT DISTANCE MEASURES THE SIMILARITY BETWEEN TWO STRINGS (AS THE MINIMUM NUMBER OF CHANGE, INSERT OR DELETE OPERATIONS THAT TRANSFORM ONE STRING TO THE OTHER)[5]. AN EDIT SEQUENCE S IS A SEQUENCE OF SUCH OPERATIONS AND CAN BE USED to represent the string resulting from applying s to a reference string. A modification to Ukkonen's edit distance calculating algorithm based upon representing strings by edit sequences[5]. We conclude with a demonstration of how using this representation can improve mitochondrial DNA query throughput performance in a distributed computing Environment[5].

D. Delta Closed Patterns and Noninduced Patterns from Sequences:

Discovering patterns from sequence data has significant impact in many aspects of science and society, especially in genomics and proteomics. Here they consider multiple strings as input sequence data and substrings as patterns. In the real world, usually a large set of patterns could be discovered yet many of them are redundant, thus degrading the output quality[6]. Their paper improves the output quality by removing two types of redundant patterns. First, the notion of delta tolerance closed itemset is employed to remove redundant patterns that are not delta closed. Second, the concept of statistically induced patterns is proposed to capture redundant patterns which seem to be statistically significant yet their significance is induced by their strong significant subpatterns[6]. It is computationally intense to mine these nonredundant patterns (delta closed patterns and noninduced patterns). To efficiently discover these patterns in very large sequence data, two efficient algorithms have been developed through innovative use of suffix tree. Three sets of experiments were conducted to evaluate their performance.[6]

E.PrefixSpan:

Prefix span is based on recursively constructing the patterns, and simultaneously, restricting the search to projected databases. An a-projected database is the set of subsequences in the database that are suffixes of the sequences that have prefix . At each step, the algorithm looks for the frequent sequences with prefix a in the corresponding projected database. In this way, the search space is reduced at each step, allowing for better performances in the presence of small support thresholds. In general, pattern growth methods can be seen as depth-first traversal algorithms because they construct each pattern separately in a recursive way. PrefixSpan is a more efficient algorithm for mining sequential patterns compared to GSP and Apriori. PrefixSpan is also capable of dealing with a very large database. PrefixSpan mainly employs the method of database projection to make the database for the next pass much smaller and can make the

algorithm faster; also in PrefixSpan there is no need for candidates generation only recursively project the database according to their prefix. In PrefixSpan, there are three different projection methods, level-by-level projection, bi-level projection and pseudo-projection. In recent years, mining sequential patterns from a multidimensional sequential database is proposed in [18]. An algorithm for mining approximate sequential patterns has been developed.

III. PRPOSED METHOD

The EleRep Algorithm

*Defination*s

Definition 1(subrepeat): Let S be the sequence ,P is repeat in S which occurs p times in S where p>=2. Let Q be the substring of P which occurs q times in Sequence S. We can say Q is subrepeat of P if

i) q=p

ii) Q occurs at position k in the P then every occurrence of Q will occur in P's occurrence only at position k.

Definition 2 (Elementary Repeat): For given sequence S, minimum length L ,minimum occurrence frequency F, P is elementary repeat if |P|>=L and f(p) i.e. frequency of

occurrence of string p in S>=F and every non trivial substring Q of P having |Q|>=L is subrepeat of P .

EleRep(S, L, F)

Input : S: String of length n
L: minimum length of elementary repeats,
F: minimum Frequency.

Output : Elementary Repeats which appears at least F times and there occurrence position

1.Construct Suffix Tree ST for String S
2. Annotate each node y with f(y) and |plabel(y)| .
3.E_P={}
4. Extract all nodes with |plabel(y)| >=L and f(y)>=F and |plabel(yp)|<L and add to E_P
5. Sort above nodes in ascending order of |plabel(y)| using counting sort.
6. For each above node y in E_P
if(ys exists and |plabel(ys)|>=L and |plabel(ysp)| < L&& f(x)==f(xs))

a. m_count(y)=m_count(y s)+1
b. c_node(ys)=y

7.ele_rep[]={0},B={};
8.call getEleREp(E_P)
9.call getBoundaries(ele_rep,B)

getEleRep(E_P: Extracted node set,L)
1..for each node y in E_P :
I. if c_node(y) does not exist then
a. print substring of plable(y) having first m_count(y)+L characters
b. Traverse the node below y and in ele_rep, at the start of suffixes indicated by leaf node below node y put the m_count(y)+L.

getBoundary(ele_rep:elemtary repeats array, B:Boundary array)
1. k=0;
2 .for i in 0 it |S|-1
a. if(ele_rep[i])
B[k]={i,i+ele_pos[i]};
3. return B;

Fig. 1. EleRep Algorithm.

Example 1: If we are given sequence S=" ABCDEFGIABCDEFGJABCDEFGKCDEFGMCD EF"L=4

and F=2. Elementary repeats are: ABCDE,CDEF,DEFG . Note that even though ABCDEFG is having frequency more than F ,we can not take it as Elementary repeat as CDEF is not subrepeat of ABCDEF. We can define our problem as given the sequence S, minimum length L and

minimum occurrence frequency F ,we have to find the maximal length elementary repeats having frequency is greater than F.

B. Description of the EleRep

EleRep algorithm is designed to get the maximal length elementary repeats from sequence .Given sequence S where S is having length n=|S|, minimum length L and minimum frequency of occurrence F, algorithm outputs the elementary repeat and start and end boundaries of the elementary repeat in sequence S. In Step1 we construct suffix tree ST using well known suffix tree construction Ukkonen's algorithm over sequence S. For a node v in suffix tree , plabel(v) denotes the string which is formed after concatenation of a path labels occurred while traversing from root node to node v. In suffix tree each leaf node indicates the one suffix from the sequence S. Each internal node v has at least 2 child. In suffix tree edge label has non empty string. Ukkonen's algorithm uses the suffix link in construction of suffix tree. In Step2, each node v in ST, is annotated with |plabel(v)| and number of leaf nodes below the given node.The number of leaf nodes below the given node is the frequency of occurrence of string plabel(v) in sequence S. For this we can use the depth first traversal. While visiting the node v first time we can annotate the length of plabel(v) I.e. |plabel(v)| . In suffix tree, maximum number of nodes will be 2n ,where n is sequence length. Therefore DFS will take O(n). In step 3 , we create set of extracted possible nodes denoted by E_P which is initially empty. This set is used to keep the nodes whose plabel can form the elementary repeat . In step 4, we extract the possible nodes which can form the elementary repeats. In this step we check that extracted node y will have |plabel(y)|>=L and f(y) i.e. frequency of string with |plabel(y)|>=F which is first necessary condition.

We denote parent node of node y with yp. While extracting node we check |plabel(yp)|<L. This condition is required to check as if |plabel(yp)| >=L indicates that there is substring s having |s|>=L such that f(ys) >f(y) . Therefore y can not be the elementary repeat because it has substring s which is not subrepeat of the string formed by palbel(y). In step 5 , if y is denotes the node E_P ,we sort the all the nodes extracted in E_P in ascending order of there plabel length .To perform sorting operation in linear time we use counting sort. We require all nodes in E_P to be sorted in ascending order because we first detect the smaller elementary repeats and we extend he smaller elementary repeat if possible using the suffix link. We perform this step in Step 6. Step 6 is an important step, in this step we try to extend the already known elementary repeat by adding the one extra character in the front. Let y be the node in E_P ,ys denotes the suffix link node of y and ysp denotes the parent node of ys. This step checks that f(y)==f(ys) which means that ys may only at those positions at y occurs. In other words, We try to extend ys with one character for getting maximal length Elementary Repeats .Above checked condition is necessary but not sufficient as ys may not be in the E_P.

For example,if S= "ABCDEIABCDEKBCD" . It can be observed that BCDE has same frequency as ABCDE. BCDE can not be extended to ABCDE as BCDE is not subrepeat because BCDE has a string which is not subrepeat. Therefore we check | plabel(ysp) |< L.

IV. CONCLUSION:

We have presented algorithm EleRep for finding exact repeats from DNA sequence depending on definition given by Zheng and Stefano. We use suffix tree constructed by Ukkonen algorithm. We store required information at each node .We first extract the nodes which may be give us elementary repeats .We perform operation on these nodes only. Our algorithm mines elementary repeats and recognizes the start and end of these repeats in the sequence. Experiment result shows that our algorithm outperforms the previously known algorithm which check the occurrence list. And practical result matches with the theoretical result.

REFERENCES

[1] Hongwei Huo ,Xiaowu Wang, Stojkovic, "An Adaptive Suffix Tree Based Algorithm for Repeats Recognition in a DNA" International joint conference Bioinformatics, Systems Biology and Intelligent Computing, 2009. IJCBS '09, pp. 181 – 184, Aug. 2009.

[2] J. Zheng and S. Lonardi. "Discovery of Repetitive Patterns in DNA with Accurate Boundaries" Proc. of Fifth IEEE Symposium on Bioinformatics and Bioengineering (BIBE'05), Minneapolis, Minnesota, pp. 105-112, 2005.

[3] Dan He, "Using Suffix Tree to Discover Complex Repetitive Patterns in DNA Sequences," IEEE International Conference of the Engineering in Medicine and Biology Society (EMBS 06), IEEE Press, pp. 3474-3477, Aug. 2006.

[4] Neil C. Jones and Pavel A. Pevzner, Introduction to Bioinformatics Algorithms, 1st ed., MIT Press: Cambridge, pp.311-337, 2004.

[5] Kurtz S, Choudhuri J. V, Ohlebusch E, Schleiermacher C, Stoye J, and Giegerich R, "REPuter: The Manifold Applications of Repeat Analysis on a Genomic Scale,"

Nucleic Acids Res., vol. 29, no.22, pp. 4633–4642, 2001.

[6] Kurtz S, Choudhuri J. V, Ohlebusch E, Schleiermacher C, Stoye J, and Giegerich R, "REPuter: The Manifold Applications of Repeat Analysis on a Genomic Scale," Nucleic Acids Res., vol. 29, no.22, pp. 4633–4642, 2001.

[7] Gusfield D, Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology, 1st ed, Cambridge University Press:Cambridge, 1997, pp87-168.

[8] Lander E.S, Linton L.M, Birren B, et al., "Initial Sequencing and Analysis of the Human Genome," Nature, vol. 409, pp. 860-921,Feb. 2001.

[9] R. Giegerich and S. Kurtz, "From Ukkonen to McCreight and Weiner: A Unifying View of Linear-Time Suffix Tree Construction," Algorithmica, vol. 19, no.22, 1997,pp. 331-353.

[10] Esko Ukkonen, "On-line construction of suffix tree," Algorithmica, vol. 14, no.3, pp.249-260, 1995.

[11] Lefebvre A, Lecroq T, Dauchel H. and Alexandre J., "FORRepeats: Detects Repeats on Entire Chromosomes and between Genomes,"Bioinformatics, vol. 19, no. 3, pp. 319-326, , 2003.