ISSN: 0975-766X
CODEN: IJPTFI
Available Online through *Research Article*
www.ijptonline.com

AN ENHANCED ALGORITHM FOR FREQUENT PATTERN MINING FROM BIOLOGICAL SEQUENCES

Kuruva Lakshmanna[1,a], Rajesh Kaluri[2,b], G Thippa Reddy*[3,c], G Nagaraja[4, d], Dhenesh V Subramanian[5,e]
[{1, 2, 3, 4}] Asssistant Professor, VIT University, Vellore, India.
[5]Lecturer in Computing Science and Information Systems, USP, Suva, Fiji.
Email: thippareddy.g@vit.ac.in

Received on 10-05-2016 *Accepted on 09-06-2016*

Abstract

Bio-data analysis deals with the most vital discovering problem of similarity search and finding relationship among bio sequences and structures. In this paper, we are trading the problem of discovering the most recurrently occurring patterns in a given DNA or protein sequence. Several on hand tools need the user to spell out gap constraints in advance in turn to find specific patterns. Practically it is not possible for the user to provide the gap constraints. So the need arises of budding an algorithm to obtain the patterns easily on its own without the need of user intervention in the form of mentioning of gap constraints. We have got two analytical methods to find out the recurrent subsequences and guesstimate the maximum support for data with complexity O(|T|.Sup) where |T| stands for text sequence length and Sup represents the number of occurrences of the pattern. We are proposing an altered version of the previously proposed algorithm with complexity O(|T|).

Keywords: Recurring patterns, mining, biological sequences, wild-cards.

1. Introduction

Patterns' appearing recurrently in a data set is known as frequent patterns. Take an example of a transaction data set in which a set of items that appears recurrently is known as recurrent item-set. A substructure occurring recurrently is known as structured pattern [3,13,23]. While the sequence data availability is escalating everyday due to the achievement of genome sequencing of various organisms, our acquaintance about the mystifying gene coding is still very inadequate [3,10]. Due to the public availability of genome sequence data a lot of research efforts are being done to extract the information of genome sequences.

Finding out the pattern hidden in the data set is of fore most importance for biologists as it may help them out to comprehend and observe the associations among different genes [2,11]. As for an example, it is well known that the

distribution of nucleotides is biased and not random and there is a very strong base pair (bp) pattern in the genomes of prokaryotes and lower eukaryotes [5,7].

These patterns are covering around half of the genomes present in prokaryotes and might be major a impenetrable network of protein interaction sites in chromosomes [5,8,9]. It is nearly impossible for molecular biologists to realize genes and their functionalities without the information of promoter and so is the need to find the promoter information arises in order to discover the gene information [1,17].

In the prediction of protein structure [6,9], sighting of local amino acids interaction pattern along the protein primary structure is a very vital objective. Two amino acids present in a region can interact with another amino acid present in the same region.

In this paper, we are study the problem of mining recurring data patterns from biological sequence with supple gap constraints, with a particular focus on biological DNA sequences.

2. Literature Survey

The discussion of mining of recurring patterns from biological sequences with wildcards problem is discussed in [2,11]. The Apriori Algorithm [3] in sequential pattern mining lays a general framework for Algorithm 1[2]. The inputs for this algorithm are a sequence T and a support threshold min (sup). The main goal is of generation of all frequent candidate subsequences in a breadth first search order by incrementing the subsequence length by one in every iteration.

The iterations will prolong until no more longer frequent subsequences are there in the given text [2],[1]. The maneuver of existing algorithm is in following two stages.

2.1 Candidate generation

Generating and pruning the candidate subsequences of length k are done by applying a procedure called Apriori-gen[3]. The main aim is to generate a candidate k subsequences by means of any two frequent (k-1) subsequences and then to prune the recently obtained candidate k-subsequence if any of its (k-1) subsequences is not recurrent [2],[1].

2.2 Maximum support generation

In this method two different heuristics are used in order to guesstimate the maximum support for a certain subsequence. We have two linear time heuristic, one is one-way scan and another is two way scan, to compute the maximum no. of non-overlapping occurrences for a subsequence [1,20].

2.2.1 One-Way Scan

Scanning from left to right and incrementing the counter for a text T is to be done until the incidence of P in T appears for a specified subsequence P. Any anon occurrences of P can't overlie with any formerly found incidence of P as affirmed by the one-off condition.

2.2.2 Two-Way scan

This method will scan a particular text T from left to right as well as from right to left iteratively to find out the incidence of a specified subsequence P[2]. To state the case more precisely, this method will scan starting from left most side t_1 of T to find out the incidence of p_1. Then it will go straight to the right most end t_n (n=|T|) after finding some $t_{i1}=p_1$ so as to find out the occurrence of last subsequence character p_m (m=|P|) in T such that $t_{im}=p_m$. This iterative scan process is continued until an occurrence of P is found and is done until there is no more occurrences of P is remaining in the text T. Time complexity of two way scan method is O(n.sup) where n is the text sequence length |T| and sup represents support for P. Consider an example of text T= ababaabb and subsequence P=abab, the two way scan method will find out the two occurrences of P in T as {1,2,6,8} and {3,4,5,7} respectively. From these occurrences[2], a pattern can very well be detected as a[0,0]b[0,3]a[1,1]b.

The two way scan method doesn't follow the ordering property which states that every character in a later occurrence of the pattern always succeeds the character with the same position index in an earlier occurrence of the pattern. The enhanced version of two way scan method will simply reorganize the occurrences of characters in different occurrences of P. Considering the previously quoted example with the application of ordering property, the third and fourth characters can be exchanged in both the occurrences of P in order to obtain a new set of occurrences of P as {1,2,5,7} and {3,4,6,8} and thus the pattern obtained for new set of occurrences is a[0,0]b[1,2]a[1,1]b which is much compressed then the earlier obtained pattern[1,19,20]. No alteration of generated patterns is needed in one way scan method because ordering property always holds true for it.

Algorithm-1: Pattern Mining(SequenceT, min sup)

1: generate L1

2: for k:=2; Lk=1/= ϕ; k++ do so

3: Ck:= ϕ; // ϕ represents an empty set

4: Ck:= Apriori-gen(Lk □1);

5: Lk:= ϕ;

6: for any P ϵ Ck do so

7: (sup1, P1):=scanoneway(P, T);

8: (sup2, P2):=scantwoway(P, T);

9: reverse T and P to try again

10: sup oneway:=max(sup1,sup1 rev);

11: sup twoway:=max(sup1,sup1 rev);

12: max sup:= max(sup oneway ,sup twoway);

13: if(max sup ≥ min sup)

14: output that subsequence in pattern format;

15: insert P into Lk;

16: end for;

17: end for;

3. The Enhanced Algorithm

3.1 Method

The proposed work is going to work on the concept of scanning the sequence text from left to right only. The steps of the adapted algorithm are:

(1) To detect the no. of elements in patterns and to count out the no. of occurrences of the element in the specified pattern.

(2) Take positions in the text sequence and to detect the no. of elements by scanning the text sequence from left to right as well as keep a count of different elements in the mentioned pattern.

(3) Measure the ratio of text count to the pattern count for each different element being detected in the pattern.

(4) Now take the maximum of the ratio obtained for each different element.

(5) Ratio value number of positions in proper order is to be taken for different elements obtained from the pattern in step (2).

(6) Step (5) is to be repeated until all the positions are filled. Complexity for this method is O(n) where n symbolizes text sequence length. This adapted algorithm is mentioned as ALGORITHM 2.

3.2 Complexity Analysis

Two way scan and one way methods are having a time complexity of O(|T|. Sup) where Sup stands for support by the two way and one way scan methods [1,2].

Time complexity for the adapted algorithm is O(n) where n stands for the text sequence length.

Algorithm-2: PatternMining(Sequence T,MinSup)

1: for each different Pattern element in Pattern(P) do

2: element size(Pattern element)=no of occurrences of element in Pattern

3: end for

4: for each pattern element in text T do

5: Pattern element array(Pattern element array count++)=pattern element position in T

6: end for

7: for each pattern element in element size array

8: temp=(Pattern element array count /

elementsize(Pattern element))

9: if min > temp then

10: min = temp

11:end if

12: end for

13: define Pattern matrix as ((min ×Pattern size))

14: for i=0 to min

15: index=0

16:for each Pattern element in Pattern do

17: for k=0 to Pattern element array count

18: if index==0 and deleted

pattern element array(k) == false then

19: Pattern matrix(i,index++)=Pattern element array(k)

20: deleted pattern element array(k)=true; break

21: else

22: if deleted pattern element array(k)==false and Pattern matrix(i,index-1) <Pattern element array(k) then

23: Pattern matrix(i,index++) =pattern element array(k)

24: deleted pattern element array(k) =true;

break

25: end if

26: end if

27: end for

28: end for

29: end for

30: if index==Pattern size then

31:break

32: end if

33: i-1 Pattern occurrences found

34: if (i-1) ≥ MinSup then

35: output that subsequence in pattern format;

36: end if

Experimental Result

In this work we have taken same input for existing algorithm as well as for the improved algorithm. The output that is average accuracy and efficiency which came by giving the same input to both the algorithms is shown in Table 1 and bar chart in Fig 1. These three sets of frequent patterns have shown in table 2 with their distributed length. We compare the number of occurences for each pattern by the two counting heuristics, and the enhanced algorithm and the results are shown in Fig 2. It is examined that on an average, an improved algorithm outperforms a two way scan, and performs similarly as the one way scan.

Table 1: Comparison table in terms of accuracy and execution time.

Algorithm	**Time(m.s)**	**Accuracy(%)**
One-way scan	7.9	98
Two-way scan	8.66	63

Enhanced algorithm	6.04	100

Table 2: Distribution of Patterns for AX829174, |T | = 1000 and Minimum support = 100

Pattern Size |P|	3	4	5	6	7	8
One-way scan	877	727	534	297	102	12
Two-way scan	624	310	70	0	0	0
Enhanced algorithm	877	731	534	297	105	14

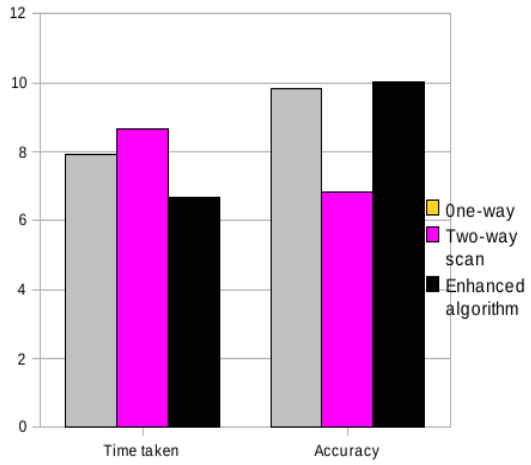

Fig 1. Performance Comparison Chart.

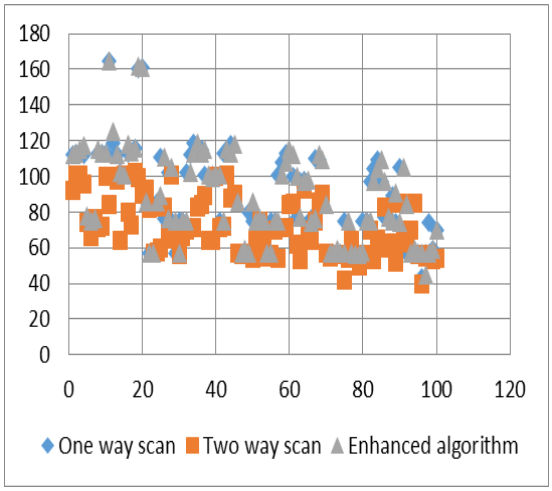

Fig 2. Comparison of the Two heuristics and the Enhanced algorithm on 100 patterns of AX829174, |T| = 1000.

4. Conclusion

In this paper we have taken a challenging task of frequent pattern mining from sequences with wild cards. The present algorithm has a complexity of O(|T|.Sup) where Sup denotes the number of pattern occurrences whereas the improved algorithm has an complexity of O(n) where n denotes the sequence length (n = |T |) . Our test results on bio-sequence indicate that our improved algorithm is found accurate in 90% of the cases. It is examined that on an average, an improved algorithm outperforms a two way scan, and performs similarly as the one way scan. We are looking for the future work that can find the possibility to come out with best solution to this problem.

References

1. Wu X, Zhu X, He Y, Arslan AN. PMBC: Pattern mining from biological sequences with wildcard constraints. Computers in biology and medicine. 2013 Jun 1;43(5):481-92.
2. He Y, Wu X, Zhu X, Arslan AN. Mining frequent patterns with wildcards from biological sequences. InInformation Reuse and Integration, 2007. IRI 2007. IEEE International Conference on 2007 Aug 13 (pp. 329-334). IEEE
3. Jaiwei H, Kamber M. Data mining: concepts and techniques, Elsevier, Second, pages 227-267, 2006.
4. Klimke W, Agarwala R, Badretdin A, Chetvernin S, Ciufo S, Fedorov B, Kiryutin B, O'Neill K, Resch W, Resenchuk S, Schafer S. The national center for biotechnology information's protein clusters database. Nucleic acids research. 2009 Jan 1;37(suppl 1):D216-23.
5. Larsabal E, Danchin A. Genomes are covered with ubiquitous 11 bp periodic patterns, the" class A flexible patterns". BMC bioinformatics. 2005 Aug 24;6(1):1.
6. Qian N, Sejnowski TJ. Predicting the secondary structure of globular proteins using neural network models. Journal of molecular biology. 1988 Aug 20;202(4):865-84.
7. Rashid M, Karim M, Jeong BS, Choi HJ. Efficient mining of interesting patterns in large biological sequences. Genomics & informatics. 2012 Mar 1;10(1):44-50.
8. Chen Z, Li J, Wei L. A multiple kernel support vector machine scheme for feature selection and rule extraction from gene expression data of cancer tissue. Artificial Intelligence in Medicine. 2007 Oct 31;41(2):161-75.
9. Vityaev EE, Orlov YL, Vishnevsky OV, Belenok AS, Kolchanov NA. Computer system "Gene Discovery" to search for patterns in eukaryotic regulatory nucleotide sequences. Molecular Biology. 2001 Nov 1;35(6):810-7.

10. Exarchos TP, Tsipouras MG, Papaloukas C, Fotiadis DI. A two-stage methodology for sequence classification based on sequential pattern mining and optimization. Data & Knowledge Engineering. 2008 Sep 30;66(3):467-87.
11. Manber U, Baeza–Yates R. An algorithm for string matching with a sequence of don't cares. Information Processing Letters. 1991 Feb 18;37(3):133-6.
12. Zhang M, Kao B, Cheung DW, Yip KY. Mining periodic patterns with gap requirement from sequences. ACM Transactions on Knowledge Discovery from Data (TKDD). 2007 Aug 1;1(2):7.
13. Zhu X, Wu X. Mining complex patterns across sequences with gap requirements. A... A. 2007;1(S2):S3.
14. Newberg LA. Significance of gapped sequence alignments. Journal of Computational Biology. 2008 Nov 1;15(9):1187-94.
15. Agrawal R, Srikant R. Mining sequential patterns. InData Engineering, 1995. Proceedings of the Eleventh International Conference on 1995 Mar 6 (pp. 3-14). IEEE.
16. Kucherov G, Rusinowitch M. Matching a set of strings with variable length don't cares. InCombinatorial Pattern Matching 1995 Jul 5 (pp. 230-247). Springer Berlin Heidelberg.
17. Agrawal R, Srikant R. Fast algorithms for mining association rules. InProc. 20th int. conf. very large data bases, VLDB 1994 Sep 12 (Vol. 1215, pp. 487-499).
18. Pei J, Han J, Mortazavi-Asl B, Pinto H, Chen Q, Dayal U, Hsu MC. Prefixspan: Mining sequential patterns efficiently by prefix-projected pattern growth. Inicccn 2001 Apr 2 (p. 0215). IEEE.
19. Zaki MJ. SPADE: An efficient algorithm for mining frequent sequences. Machine learning. 2001 Jan 1;42(1-2):31-60.
20. Ji X, Bailey J, Dong G. Mining minimal distinguishing subsequence patterns with gap constraints. Knowledge and Information Systems. 2007 Apr 1;11(3):259-86.

Corresponding Author:
G Thippa Reddy*,
Email: *thippareddy.g@vit.ac.in*