Mining Maximal Adjacent Frequent Patterns from DNA Sequences using Location Information

Moin Mahmud Tanvee Shaikh Jeeshan Kabeer Tareque Mohmud Chowdhury
Asif Ahmed Sarja Md. Tayeb Hasan Shuvo
Department of Computer Science and Engineering
Islamic University of Technology
Gazipur, Bangladesh

ABSTRACT
The rapid development of bioinformatics has resulted in the explosion of DNA sequence data which is characterized by large number of items. Studies have shown that biological functions are dictated by contagious portions of the DNA sequence. Finding contiguous frequent patterns from long data sequences such as DNA sequences is a particularly challenging task and can pave the way towards new breakthroughs. Apriori based techniques were among the first to be used in frequent contagious pattern mining. Later improved approaches like GSP, Prefix Span were also applied but the approaches required either large number of sequence scans, generated large number of candidates or required higher number of intermediate sequential patterns. In this paper an improvement of the positional based approach for contagious frequent pattern mining is DNA sequences is proposed. The proposed algorithm improves the existing positional based approach by introducing a new amalgamated sorting and joining technique which helps to reduce time and space complexity. The proposed approach outperforms traditional existing contiguous frequent mining approaches.

General Terms
Bioinformatics, Maximal Contagious Frequent Pattern Mining, Data Mining

Keywords
DNA Sequence Data, Maximal Contagious Frequent Pattern Mining, Sequence Information.

1. INTRODUCTION
The relentless development of bioinformatics; particularly that of the Human Genome Project have lead to a wealth of biological datasets including DNA sequence datasets. The study of DNA sequences have led researchers to believe that the biological state of individuals is a direct ramification of the DNA sequence state and how they interact. But it is a challenge for researchers as how to process this vast volume of data and identify important patterns and sequences [1].

Large data sequences are usually very large and contain hundreds or thousands of items. DNA sequence data however has only four repeating items A, T, C, and G. Biological

sequences such as DNA sequences contain specific patterns which can used as a phylogenic foundation. Using frequent pattern mining, similarities between DNA sequences can be used to analyze their underlying ties [2].

Maximal contiguous sequence mining is the method of finding frequent maximal contiguous patterns from sequences which are more than two [3, 4]. Although the application of sequential pattern mining in bioinformatics is relatively new, it has been widely applied in other fields [5, 6].

Many of the earliest sequential mining were inspired by the very popular Apriori Algorithm [7]. A very similar approach also based on the Apriori is the GSP [6] which continuously scans the data sequence to generate candidate solutions and then tests them. A more efficient modification of GSP was the PrefixSpan [8] which uses projected databases to reduce the number of scans by employing projected databases to grow sequential patterns. Another recent method, MacosFSpan was proposed in [9], which is based on the PrefixSpan to overcome the recursive growing of sequential pattern problem. Kang et al. [2] proposed an efficient modification of PrefixSpan using fixed length spanning tree. A more recent method was proposed Zerin et al. [10] to improve [2] using positional information to improve the efficiency of frequent pattern mining.

This paper proposes an efficient improvement of the position based technique [10] for mining maximum length concatenate frequent patterns form the large DNA sequence datasets. The positional based approach is improved by introducing a new sorting technique which sorts the candidates according to their position in the database. Then the sorted subsequences are joined and the maximum length frequent pattern is generated in less number of iterations which reduce both the time and memory complexity. The proposed approach outperforms other related approaches.

This paper is organized as follows, Section 1 was the Introduction. Section 2 provides an overview on the related works. Section 3 illustrates the proposed framework in detail. Section 4 shows the results of the proposed framework on multiple DNA sequences, while Section 5 concludes the study with future scopes and conclusion.

2. RELATED WORK
A lot of research has already been done on maximal contiguous frequent pattern mining in DNA sequences as mentioned in the previous section. Most of the early works were based on the Apriori Algorithm [7] as already mentioned. Apriori is a classic algorithm for finding frequent item sets in databases [1]. Apriori is based on the principle 'super pattern of a non frequent pattern cannot be frequent'. Following the footsteps of Apriori came another algorithm called the GSP [6].GSP was based on continuously scanning the database generating candidate solutions. It starts by

generating single item candidate solutions, in the second pass it will generate two item sequences and will do another pass to check the support of the candidates. It continues in this manner first by generating an item candidate and then doing another pass to check it. GSP is inefficient since it required large number of database scans and the generation of so many candidate solutions introduces memory constraints.

An efficient version of GSP was introduced and it was known as PrefixSpan which uses projected databases. PrefixSpan only takes into account the prefix subsequences and generates or projects only their postfix subsequence. A recursive process of growing the projected database sequential patterns is done by exploring local length-1 frequent patterns. However PrefixSpan was not suitable for mining long sequences since growing in this recursive manner of one item at a time is extremely inefficient [9].

To overcome the recursive growing nature of PrefixSpan another approach called the MacosFSpan was introduced which uses fixed length span approach to reduce the recursive growing problem of PrefixSpan and it also adapts Suffix Tree to tackle the maximal problem [9]. However MacosFSpan does still use the projected databases and the size of databases grows very fast as the algorithm progresses and thus is not suited or large data sequences such as biological data.

Kang et al [2] has proposed an efficient algorithm which reduces the frequency of data sequence scans by reducing unnecessary data generation using a spanning tree of fixed length and the frequent patterns are generated using the spanning tree. The candidate sets are incrementally grown in steps of one until the maximal frequent sequence is generated. But the approach still requires multiple scans of the database when higher length candidates were generated and the database also needs to be scanned for checking the support.

Zerin et al. [10] proposes to improve the performance of [2] using position information of the pattern in the data sequence. Zerin et al proposes that the candidates of higher lengths can be generated without the need for scanning of the data sequence. The proposed approach [10] incrementally increases the size of the frequent pattern in steps of one but it was found and it will also be shown that the maximal pattern can be generated without a stepwise increase in steps of one using an innovative sorting technique. As mentioned before based on the sorted candidate sequences the joining and generation will be done in steps of more than one and hence the maximal contiguous frequent pattern can be generated in fewer steps and hence the overall computational and memory efficiency will increase.

3. PROPOSED METHOD

In this section the proposed approach for finding maximal frequent pattern generation will be elaborated. The overall approach is shown in Figure 1.

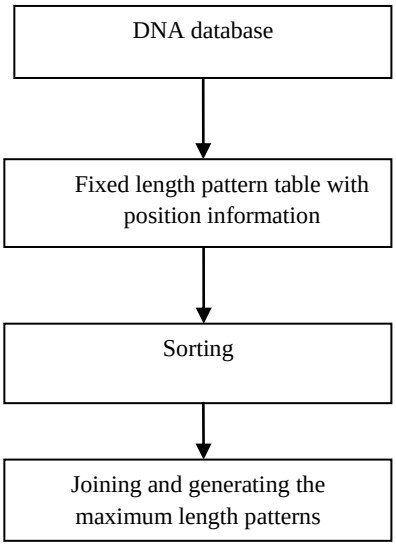

Fig 1: Proposed Approach

The proposed algorithm starts working with the DNA database containing the DNA sequences. To get the frequent fixed length patterns from the database a pattern table was generated as shown in [10]. As mentioned before the advantage of keeping the position information is that candidates can be generated without scanning the database repeatedly. Table 1 shows the sample data sequences. From the table it can be seen that 4-length frequent patterns in a single database scan like position based approach [10] considering minimum support 2. Minimum support 2 means only those 4-length patterns are stored which occur at least twice in the database.

Table 1: Sample DNA Sequence database

ID	***Sequence***
10	ACTATTGTAGAGTA
20	AGTATTAATCGAT
30	ACTAGTCGATCG
40	CTAGTGCGATCTATGCTTAA
50	GAGTGCTTAATCG

Table 2 shows all 4-length patterns along with their position information, which have satisfied the minimum support =2. Patterns found less than 2 times are considered infrequent.

Table 2:4-length patterns with position information

Pattern Index	Pattern	Position information(seq_id, start position)
1	CGAT	(20,10) (30,7) (40,7)
2	CTAG	(30,2) (40,1)
3	CTAT	(10,2) (40,11)
4	CTTA	(40,16) (50,6)
5	ACTA	(10,1) (30,1)
6	AATC	(20,7) (50,9)
7	AGTG	(40,3) (50,2)
8	ATCG	(20, 8)(30,9) (50,10)
9	AGTA	(10,11) (20,1)
10	GCTT	(40,15) (50,5)
11	GTGC	(40,4) (50,3)
12	GAGT	(10,10) (50,1)
13	GATC	(30,8) (40,8)
14	TCGA	(20, 9) (30,6)
15	TAAT	(20, 6) (50,8)
16	TAGT	(30, 3) (40, 2)
17	TATT	(10, 3) (20,3)
18	TGCT	(40, 14) (50, 4)
19	TTAA	(20,5)(40,17)(50,7)

3.1 Sorting fixed length patterns

The new sorting technique where all the patterns will be sorted based on its last occurring position. In existing traditional approaches like position based approach, the fixed length frequent patterns are sorted alphabetically which is time consuming. But if the patterns are sorted by looking at its last occurring position it will make the joining step much easier to produce higher length pattern. Figure 2 shows the sorting process of the patterns. To make the proposed approach robust a hash table is created as shown in Table 3. In the hash table, there are 4 rows each of which contains the relative indexes of the pattern in the table starting with A, C, T and G respectively.

Pattern Index	Pattern	Position information(seq_id, start position)
1	CGAT	(20,10) (30,7) **(40,7)**
2	CTAG	(30,2) **(40,1)**
3	CTAT	(10,2**) (40,11)**
4	CTTA	(40,16) **(50,6)**
5	ACTA	(10,1) **(30,1)**
6	AATC	(20,7) **(50,9)**
7	AGTG	(40,3) **(50,2)**
8	ATCG	(20, 8)(30,9) **(50,10)**
9	AGTA	(10,11) **(20,1)**
10	GCTT	(40,15) **(50,5)**
11	GTGC	(40,4) **(50,3)**
12	GAGT	(10,10) **(50,1)**
13	GATC	(30,8) **(40,8)**
14	TCGA	(20, 9**) (30,6)**
15	TAAT	(20, 6) **(50,8)**
16	TAGT	(30, 3) **(40, 2)**
17	TATT	(10, 3) **(20,3)**
18	TGCT	(40, 14) **(50, 4)**
19	TTAA	(20,5)(40,17),**(50,7)**

*** Bold locations represents the last occurring position of the Pattern*

Pattern Index	Pattern	Position information(seq_id, start position)
1	ATCG	(20, 8)(30,9) (50,10)
2	AATC	(20,7) (50,9)
3	TAAT	(20, 6) (50,8)
4	TTAA	(20,5)(40,17)(50,7)
5	CTTA	(40,16) (50,6)
6	GCTT	(40,15) (50,5)
7	TGCT	(40, 14) (50, 4)
8	GTGC	(40,4) (50,3)
9	AGTG	(40,3) (50,2)
10	GAGT	(10,10) (50,1)
11	CTAT	(10,2) (40,11)
12	GATC	(30,8) (40,8)
13	CGAT	(20,10) (30,7) (40,7)
14	TAGT	(30, 3) (40, 2)
15	CTAG	(30,2) (40,1)
16	TCGA	(20, 9) (30,6)
17	ACTA	(10,1) (30,1)
18	TATT	(10, 3) (20,3)
19	AGTA	(10,11) (20,1)

Fig 2: Sorting Patterns

3.2 Joining

For joining frequent patterns the Apriori joining rule is adopted [10]. The patterns are joined from the top of the table. Using Apriori joining rule, by joining two length patterns we get the upper length pattern. Here Among the two patterns, first one is called prefix pattern and the next is suffix pattern. The joining rule is, if both the patterns are of length n, then the last (n-l) nucleotides of the prefix pattern will be same with the first (n-1) nucleotides of the suffix pattern [10]. For example, suppose a length-4 frequent pattern is CGAT. For generating length-5 pattern this pattern will need to be joined with some pattern that starts with G and the corresponding nucleotides are same. From the hash table the indexes for the pattern starts with nucleotide G can be found. A counter is also used that counts the frequency of the newly generated next length pattern. To increase the counter value the position information stored in the sorted pattern table is used. The counter value is only updated if both the prefix and suffix patterns are in same sequence and the starting position of the suffix pattern should be just after the starting position to the prefix pattern. If the frequency count is equal or more than the minimum support threshold, the newly generated pattern will be taken as frequent and it is stored it along with the first pattern. Next time if the first pattern needed to joined as the suffix pattern, first it will join it with the updated pattern and check the frequency of the newly generated pattern. If newly generated pattern has the frequency less than minimum support threshold then it will be joined in the second pattern. The joining process is illustrated in Figure 3.

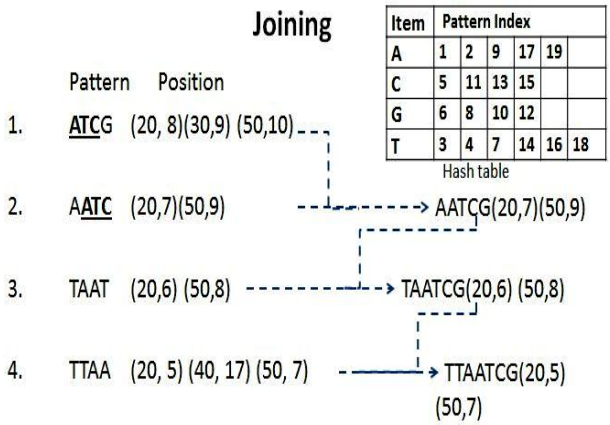

Fig 3: Joining Process

The top element of the table is ATCG. Considering this pattern as prefix pattern and the pattern which will be joined with this pattern to generate higher length pattern is suffix pattern. Now, to get a higher length pattern from the prefix pattern it needs to be joined with some suffix pattern starting with T and consecutive next 2 nucleotides C and G. T is searched in the hash table and TCGA is found in 16thposition of the table. Now the positions of ATCG and TCGA need to be matched to see whether the produced pattern is frequent. But after matching it is found that 5 length pattern is infrequent. Hence, the pattern AATC will now be prefix pattern. Now some pattern ATC will be searched in the hash table and the index was found to be 1. After joining these patterns 5-length pattern AATCG was obtained. Now the frequency of the newly generated pattern will be obtained by examining the position information of two input 4-length patterns from the table. For prefix pattern AATC, its corresponding positions in example DNA database are 7th position of sequence 20 and 9th position of sequence 50. For suffix pattern its corresponding occurring positions are 8th position of sequence 20, 9th position of sequence 30 and 10th position of sequence 50. Therefore, after joining the prefix and suffix pattern, a 5-length pattern AATCG is generated that actually occurs at 7thposition of sequence 20 and 9thposition of sequence 50 respectively. Since AATCG has support of 2 that satisfy the minimum support so it is stored along with its 4-length prefix pattern in the pattern table. Next time when AATC is used as suffix pattern, at first the prefix pattern TAAT will be joined with updated suffix pattern AATCG and it will be checked whether the 6-length pattern TAATCG is frequent or not. This way every time the updated higher length pattern will be used to reach the maximum length pattern that avoids generating all the intermediate length patterns. Existing approaches generate all the intermediate length patterns to generate the maximum length pattern while the proposed approach of this paper uses the updated solutions to get the maximum length pattern. Thus the desired maximum length pattern was obtained by generating less number of patterns which not only reduces the time but diminishes space complexity. Table 3 shows the outcome of the above mentioned process.

Table 3: After Joining Operation

Fixed length Patterns& position	Updated result & position
1.ATCG (20, 8)(30,9) (50,10)	X
2.AATC (20,7)(50,9)	AATCG(20,7)(50,9)
3.TAAT (20,6) (50,8)	TAATCG(20,6) (50,8)
4.TTAA (20, 5) (40, 17) (50, 7)	**TTAATCG**(20,5) (50,7)
5.CTTA (40,16) (50,6)	CTTAA(40,16) (50,6)
6.GCTT (40,15)(50,5)	GCTTAA(40,15) (50,5)
7.TGCT (40, 14) (50, 4)	**TGCTTAA**(40,14) (50,4)
8.GTGC (40,4)(50,3)	X
9.AGTG(40,3)(50,2)	AGTGC(40,3) (50,2)
10.GAGT (10,10)(50,1)	X
11.CTAT (10,2) (40,11)	X
12.GATC (30,8)(40,8)	X
13.CGAT (20,10) (30,7) (40,7)	CGATC(30,7) (40,7)
14.TAGT (30, 3) (40, 2)	X
15.CTAG (30,2) (40,1)	CTAGT(30,2) (40,1)
16.TCGA (20, 9)(30,6)	TCGAT(20,9) (30,6)
17.ACTA (10,1) (30,1)	X
18.TATT (10, 3) (20,3)	X

*** X represents the newly generated patterns are not frequent and patterns shown with the bold line are maximum length frequent pattern.*

As shown in Table 3, the proposed approach generates only 10 intermediate patterns to get the maximum 7 length patterns and hence the result has also be obtained with fewer steps. The proposed algorithm is shown below.

The following algorithm shows the working of the proposed algorithm.

Algorithm: Proposed Algorithm

Input: DNA database contains *N* DNA sequence (S_1, S_2,S_3….S_n), Minimum Support Threshold *Min_Sup*

Parameters: '*N*' is the number of sequences, fixed pattern length *W*

Output: maximum length frequent concatenate DNA subsequence

[1] extract fixed length DNA subsequence by scanning the DNA database and construct fixed length pattern table

[2] For *(i=0; i<N, i++)*

//extract fixed length subsequences with position information using position based approach [10]

//store all the fixed length subsequence in the fixed length pattern table

[3] sort the subsequences of the fixed length pattern table

//sort the sequence by observing its last occurring position

//create a hash table that contains that contains the positions of the subsequence starting with A, T, C and G respectively.

[4] joining the subsequence to get the maximum length frequent concatenate DNA subsequence

//start joining from the top of the pattern table

// take one subsequence as prefix pattern and search suffix pattern from the hash table.

// if the suffix pattern starts right next to the prefix pattern and their sequence id is same then the frequency of the newly generated pattern will be incremented by one

//if the frequency of the new subsequence is more than or equal to the Min_Sup, then store it along with its fixed prefix pattern.

// Next time if its fixed prefix pattern is needed to be used as suffix pattern then use the newly generated pattern and generate the higher length pattern and check its support count like previous.

// by this way generate the upper length pattern until reaching the maximum length possible.

4. EXPERIMENTAL & PERFORMANCE ANALYSIS

The proposed algorithm was applied on two datasets named HMR195 Dataset and BursetGuigo96 Dataset. HMR195 Dataset contains total 195 DNA sequences of human, mouse and rats. The mean length is 7096 bp. In BursetGuigo96 Dataset there are total of 570 sequences of vertebrate animals. Both of the DNA datasets were extracted from GenBank.

The algorithm was implemented in visual C++ 10 and run it on windows 7 platform with core2 duo 2.13 GHz CPU with a 4 GB of main memory.

Table 4 and Table 5 show the comparison of the run-time performance of different approaches to find out the maximum length pattern with various values of minimum support. The proposed algorithm was compared with 3 latest existing approaches. They are position based approach [10], position based approach with indexing, MacosVspan. From the table it can be seen that for minimum support threshold of 10%, the proposed algorithm took only 18.325 seconds where the latest Position based approach with indexing took more than 43.672 seconds. Other algorithms such as MacosVspan and Position based approach took 150.432 and 61.205 seconds to get the maximum length pattern respectively. The comparative analysis of the algorithms for BursetGuigo96 Dataset is shown in table 5. From both of the tables it can be seen that the proposed algorithm outperforms other algorithms.

Besides, the latest existing approaches like position based approach generally sort all the fixed patterns alphabetically. This sorting is very useful for the joining step from which higher length pattern was obtained. But sorting alphabetically is time consuming. The main contribution of our approach is, after getting the entire fixed length pattern; the patterns were sorted based on the last occurring position of that pattern. Since, the position information was stored initially, sorting is very easy and it saves time compared to the existing position based approach.

To make the process of comparison easier a graph based comparison is done. The comparison among different methods is graphically represented in Fig. 4 and, Fig 5.

Table 4: Performance comparison in terms of run time among proposed and other approaches for different support thresholds (for HMR195 Dataset)

Support threshold (%)	MacosVspan (seconds)	Position based approach (seconds)	Positioned based approach with indexing (seconds)	Proposed approach (seconds)
10	150.432	61.205	43.672	18.325
20	102.694	46.672	31.792	10.893
30	78.548	33.456	23.162	8.227
40	56.542	24.673	18.567	5.784
50	39.746	19.898	12.480	2.754

Table 5: Performance comparison in terms of run time among proposed and other approaches for different support thresholds (for HMR195 Dataset)

Support threshold (%)	MacosVspan (seconds)	Position based approach (seconds)	Positioned based approach with indexing (seconds)	Proposed approach (seconds)
10	357.759	157.895	97.859	41.892
20	250.592	109.483	76.474	23.163
30	183.529	81.540	57.582	18.354
40	143.686	63.469	46.361	11.784
50	92.758	47.983	26.543	5.155

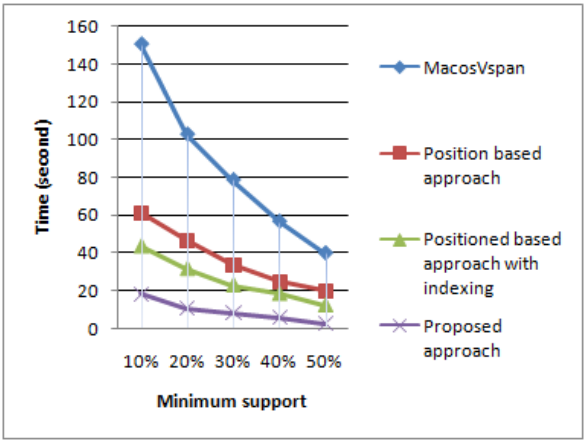

Fig 4: Performance comparison in terms of time among proposed approach and other approaches for different support thresholds (for HMR195 Dataset)

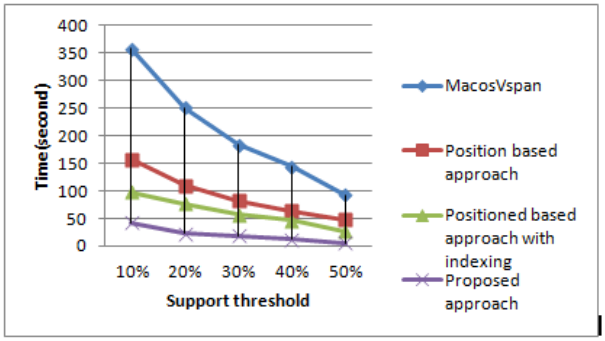

Fig 5: Performance comparison in terms of run time among proposed and other approaches for different support thresholds (BursetGuigo96 Dataset)

5. CONCLUSION & FUTUREWORK

In this paper an efficient approach to mine the maximum length frequent pattern in DNA database is proposed where location information is used. A new sorting technique is introduced to mine maximum length frequent patterns within a single calculation. Compared to other existing techniques, the proposed approach generates less number of patterns and also takes less time to mine the maximum length frequent pattern. In the future, the proposed approach can be optimized by adjusting different parameters and applying the proposed algorithm in other biological sequence datasets.

6. REFERENCES

[1] Shuang Bai, Si-Xue Bai, "The Maximal Frequent Pattern Mining of DNA Sequence", GrC, pp 23-26, 2009.

[2] T.H Kang, J.S Yoo and H, Y Kim, "Mining frequent contiguous sequence patterns in biological sequences", in proceeding of the 7th IEEE International Conference on Bioinformatics and Bioengineering, pp 723-8, 2007.

[3] R. Wanger and M. Fischer "The string-to-string Correction Problem" J. of the ACM (JACM), Vol 21, No 1, pp.168-173, 1974.

[4] D. Hirschberg "Algorithms for the longest common subsequence problem" J. of the ACM (JACM). Vol 24, No 4, pp.664-675, 1977.

[5] M. Garofalakis, R. Rastogi, and K. Shim, "Spirit: Sequential pattern mining with regular expression constraints." In Proc. 1999 Int. Conf. Very Large Data Bases (VLDB'99), pages 223–234, Edinburgh, UK, Sept. 1999.

[6] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements." In Proc. 5th Int. Conf. Extending Database Technology (EDBT'96), pages 3–17, Avignon, France, Mar. 1996.

[7] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules." In Proc. 1994 Int. Conf. Very Large Databases (VLDB'94), pages 487–499, Santiago, Chile, Sept. 1994.

[8] J. Pei, J. Han, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M.C. Hsu, "PrefixSpan: Mining sequential patterns efficiently by prefix-projected pattern growth." InICDE'01, Germany, April 2001.

[9] J. Pan, p. Wang, W. Wang, B. Shi and G. Yang, "Efficient algorithms for mining maximal frequent concatenate sequences in biological datasets", in proceeding of the fifth International Conference on Computer and Information Technology (CIT), , pp 98-104, 2005.

[10] . Zerin SF, Ahmed CF, Tanbeer SK, Jeong BS, "A fast in-dexed-based contiguous sequential pattern mining tech-nique in biological data sequences." In: Proceeding of 2nd International Conference on Emerging Databases (EBD'10), 2010 Aug 30-31, Jeju.

[11] Rashid MM, Karim MR, Hossain MA, Jeong BS, "An ef-ficient approach for mining significant contiguous fre-quent patterns in biological sequences." In: Proceeding of 3rd International Conference on Emerging Databases (EBD'11), 2011 Aug 25-27, Incheon

[12] Rashid MM, Karim MR, Hossain MA, Jeong BS and -Jin Choi, "Efficient Mining of Interesting Patterns in Large Biological Sequences." Genomics & Informatics Vol. 10(1) 44-50, March 2012

[13] Jiawei Han & Micheline Kamber , "Data Mining Concepts and Techniques", Elsevier, 2006.

[14] Pan-Ning Tan, Vipin Kumar, Michael Steinbach , " Introduction to Data Mining", Pearson Education Inc, 2006

[15] Ye-In Chang, Chen-Chang Wu, Jiun-Rung Chen and Yin-Han Jeng, "Mining Sequence Motifs from Motif Databasesbased on a Bit Pattern Approach", International Journal of Innovative Computing, pp. 647-657,2012 .

[16] J. Yang, W.Wang, P.S Yu and J.Han, "Mining long sequential patterns in a noisy environment",SIGMOD, 2002