Mining Maximal Frequent Contiguous Sequences in Biological Data Sequences

Tae Ho Kang*
Department of Computer and Communication Engineering
Chungbuk National University, Cheongju, Korea.

Jae Soo Yoo
Department of Computer and Communication Engineering
Chungbuk National University, Cheongju, Korea.

Hak Yong Kim
Department of Biochemistry
Chungbuk National University, Cheongju, Korea

Byoung Yup Lee
Department of Electronic Commerce
Paichai University, Daejeon, Korea

ABSTRACT

Biological sequences such as DNA and amino acid sequences typically contain a large number of items. They have contiguous sequences that ordinarily consist of more than hundreds of frequent items. In biological sequences analysis(BSA), a frequent contiguous sequence search is one of the most important operations. Many studies have been done for mining sequential patterns efficiently. Most of the existing methods for mining sequential patterns are based on the Apriori algorithm. In particular, the prefixSpan algorithm is one of the most efficient sequential pattern mining schemes based on the Apriori algorithm. However, since the algorithm expands the sequential patterns from frequent patterns with length-1, it is not suitable for biological datasets with long frequent contiguous sequences. In recent years, the MacosVSpan algorithm was proposed based on the idea of the prefixSpan algorithm to significantly reduce its recursive process. However, the algorithm is still inefficient for mining frequent contiguous sequences from long biological data sequences. In this paper, we propose an efficient method to mine maximal frequent contiguous sequences in large biological data sequences by constructing the spanning tree with a fixed length. To verify the superiority of the proposed method, we perform experiments in various environments. The experiments show that the proposed method is much more efficient than MacosVSpan in terms of retrieval performance.

***Keywords**: Sequence Pattern Mining, Sequence Analysis, Motif Finding Problem, Bioinformatics.*

I. INTRODUCTION

Nowadays, development of bioinformatics has accumulated the amount of biological information rapidly, and required information technologies for quicker and more effective analysis of huge biological information. Analysis and prediction of genetic function by bioinformatics can curtail the exorbitant expenses required by the existing biological technique, and lessen the time taken for experimental verification. Analysis of genetic functions includes predictions of regulator binding sites, of promoter regions and of specific protein function region and the comparison of DNA and amino acid sequence homology, etc. The results of this analysis can offer important clues for new genetic discovery and functional analysis. Comparative and analytic methods of the homology of biological data sequences are typical bioinformatics technologies for the analysis of huge biological data. Biological sequence data have two kinds of sequences - DNA sequence and amino acid sequence. Generally, these

Corresponding author. E-mail : thkang@netdb.cbnu.ac.kr
Manuscript received May 29, 2007 ; accepted June 22, 2007

This work was supported by the Korea Research Foundation Grant funded by the Korean Government(MOEHRD)" (The Regional Research Universities Program/Chungbuk BIT Research-Oriented University Consortium) and the research grant of the Chungbuk National University in 2006 (H.Y.Kim).

sequence data are huge and spread over several databases and each sequence is very long as it has hundreds or thousands of items. These biological sequences have regions or specific patterns, in which have been conserved from genetic mutation or variation, while conserved regions or specific patterns coexisting in several biological sequence data can be applied as a phylogenetic foundation and also have close relation with functions. Searching motif and domain is divided greatly into two. The first is finding a supposed motif & domain or sequence to represent a set from the given sequence data set. The second is searching the existing open database to know if the given sequence has a specific motif or domain.

Of them, frequent maximal contiguous sequence mining is the first, which tries to find the frequent maximal contiguous sequence pattern from sequences of more than two[1-3]. The problem of finding the frequent maximal contiguous pattern is very important in bioinformatics and is widely used[4-7]. When more than two DNA sequences are given, their commonly-included sequence pattern can be used in order to understand their biological correlation. Although in the beginning, researchers tried to discover frequent contiguous patterns based on the Apriori algorithm[8][9], lately they have attempted to improve performance by a method that uses a projected database and the PrefixSpan Algorithm[10][11]. However, when mining long frequent contiguous sequences, the method to recursively grow sequential patterns item by item is inefficient in terms of time and space.

Therefore, this paper reduces another data space and decreases the frequency of database accesses dramatically by removing unnecessary data generation and proposes an algorithm that is able to discover frequent subsequence pattern from several sequence data. For this, first of all, all sequence data of the database is read in the same way as the fixed length and fixed-length spanning tree is constructed. And the fixed-length contiguous sequences satisfying a specified minimum support threshold are produced by using the constructed fixed-length spanning tree, and the sequence candidate set of up to the maximal length is produced by expanding these sequences 1 by 1 in length. To show the superiority of the proposed algorithm, experiments are performed under a variety of environments.

The rest of this paper is organized as follows. Section 2 explains the recent related researches and suggests their problems. Section 3 proposes a new algorithm in order to solve the problems and analyzes its characteristics. In section 4, we compare the performance of our proposed method with the existing methods. Finally, Section 5 presents the conclusion.

II. RELATED WORK

Many researches have been performed to find the frequent maximal contiguous subsequence from more than two sequences. Most of Existing methods for mining sequential patterns are Apriori-like, ie., based on the Apriori property proposed in association mining[4], in which states that *any super-patterns of nonfrequent patterns cannot be frequent.* A typical Apriori-like algorithm such as GSP[5] adopts a multiple-pass, candidate-generation-and-test approach in sequential pattern mining. However, their running time increases exponentially with increasing average sequence length, and thus such high-dimensional data render most current algorithms impractical. Based on the idea of Apriori, a more efficient algorithm, called PrefixSpan[6], has been proposed in recent years, which presents the projection-based sequential pattern-growth approach. Its general idea is to examine only the prefix subsequences and project only their corresponding postfix subsequences into projected databases. In each projected database, sequential patterns are grown by exploring local length-1 frequent patterns. However, because every expansion of sequence pattern needs a recursive process using the projected database, it is inefficient in long sequence data like the DNA or amino acid sequences. Last, the MacosVSpan[7] algorithm produces, once every item data, the projected database consisting of subsequences sequentially concatenating each data item, gets the maximal subsequence of each data item by fixed-length span method, and looks for the frequent maximal contiguous subsequence by using the suffix tree. In order to explain the algorithm, first of all, we suppose that there is a DNA sequence database as shown in Tab. 1 and assume that minimum support is 2.

Table 1. DNA Sequence Database

ID	sequence
10	ATCGTGACT
20	CATCGTT
30	CATCGTGAAG
40	TCGTGATTG
50	GCGTGATT

The set of items in the database is {A, C, T, G}. A sequence <ATCGTGACT> is 9-sequence since there are 9 instances appearing in this sequence. The whole sequence <ATCGTGACT> contributes 2 to the support of <A>. Since both sequences 10 and 30 contain contiguous subsequence s=<ATCGTGA>, s is a frequent contiguous subsequence of length 7.

MacosVSpan method first construct four projected databases: <A>-, <C>-, <T>- and <G>- projected database. The projected database of item A is <TCGTGACT>, <CT>, <TCGTT>, <TCGTGAAG>, <AG>, <G>, <TTG> and <TT>. Fig.1 shows spanning tree of 3 of fixed length for getting the maximal sequence satisfying 2(the minimum support threshold) starting with A by using the span method from the A-projected database.

The internal node in the tree represents the overlapping prefix and the leaf node in the tree represents the subsequence. The fixed-length span method with a root ATCG satisfying the minimum support threshold, of the subsequences of leaf node, is executed again as shown in the right side of Fig.1. In the result, the maximal contiguous sequence starting with A and coming through the process is <ATCGTGA>. Then the same process is recursive for C, T and G in the root ATCG. Finally, producing the suffix tree of the maximal sequences of each item enables the last maximal sequence to be found.

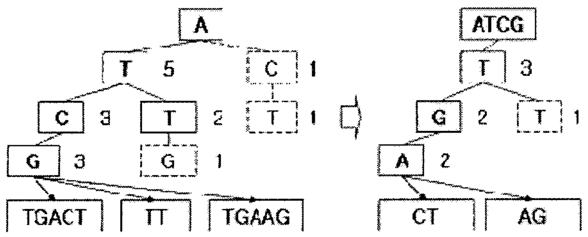

Fig. 1. Fixed Length Span Method

The algorithm significantly decreases the recursive execution process for expanding sequence patterns, when using the PrefixSpan method. However, MacosVSpan also has the problem of producing and using the projected database. The longer the data sequence becomes, the larger the projected database gets, the faster the size of the projected database increases in comparison with the original database. Its detailed explanation is treated in the discussion of the problems of the projected database in section 4.1. The projected database is very big over the original data in size. The more the kinds of items, the greater the expense of obtaining the projected database of each item, for example, in the case of an amino acid sequence consisting of 20 items.

III. THE PROPOSED MAXIMAL CONTIGUOUS SEQUENCE MINING ALGORITHM

In this section, we propose an algorithm to efficiently find the frequent maximal contiguous sequences from several biological data sequences without producing the projected databases.

Fig.2 shows the maximal contiguous sequence algorithm proposed in this paper. The algorithm is divided into 3 stages. The first stage recursively reads the subsequence with the same length as the fixed-length window from the given sequences, and constructs a fixed-length suffix tree. The second stage produces subsequences(satisfying minimum support threshold) from the constructed tree through depth first search, and expands them into the whole candidate sequence. The third stage searches the maximal sequence satisfying the real support threshold by finally comparing the produced candidate sequence and the database. At this time, comparison is executed from long candidate sequences to short ones, and stops when satisfactory results are achieved. More detailed explanation on algorithm execution is done through examples.

```
Algorithm
Input : Sequences Set S(S1, S2.. SN), FixedLength W, Minimum Support threshold Min_Sup
Output : fixed length suffix trie T, MFCSeq
Step 1: extract fixed length subsequence from database and construct spanning tree
1. for(i=0; i< N; i++)
2.    WS = extractFixedLengthSubseq(W, Si); //extract subsequences
3. memTree = constructTree(W); // construct spanning tree
4. for(i=0; i< WS; I++)
5.    insertSequences(WSi, memTree); //insert sequence and frequency count
Step 2: search tree and extract candidate sequence
6. candidateSeq = searchTree(memTree, Min_sup); //extract candidate sequence
7. resultSeq = makeSeqCandidate(candidateSeq); // expand sequence
Step 3: compare candidate sequence with Database
8. MFCSeq = searchDatabase(resultSeq);
```

Fig. 2. The Maximal Contiguous Sequence Mining Algorithm

```
ATCGTGACT
ATCG
 TCGT
   ...
     GACT
CATCGTT
CATC
   ...
   CGTT
```

Fig. 3. Fixed Length Scan Method

The algorithm is based on DNA sequence database of Tab. 1. Its fixed length is supposed as 4 and its minimum support threshold as 2. Above all, the sequence database with length-4(in the size of data fixed-length window) is moved one by one from the first position and is read in the same length as the fixed-length window, and then a spanning tree of 4(in fixed length) is constructed. Fig.3 shows that sequence data are read in the same length as the fixed length. Fig. 4 shows that the spanning tree is constructed by reading the whole sequence in the same length as the fixed-length specified in the same way as Fig.3. Each node of the tree includes a frequency and maintains the frequency of subsequence overlapping.

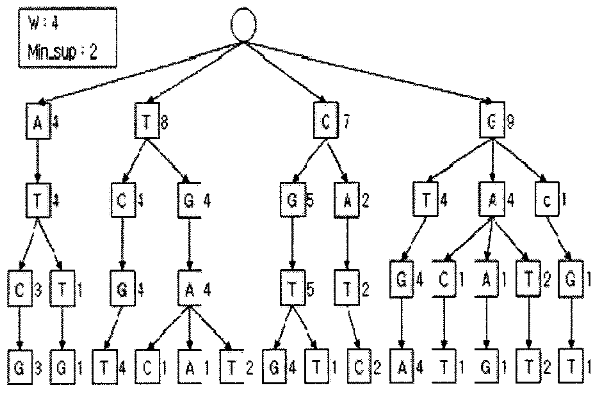

Fig. 4. Fixed Length Spanning Tree

Fig.4 shows a fixed-length spanning tree that includes the frequency information of all subsequences with length-4 occurring from each sequence data of the whole sequence database. When the tree is constructed like Fig.4, retrieval of the tree can obtain a contiguous subsequence with length-4, satisfying the minimum support threshold through frequency search. The obtained contiguous sequences are <ATCG>, <TCGT>, <TGAT>, <CGTG>, <CATC>, <GTGA> and <GATT>. These data are all subsequences with length-4, occurring over the minimum support threshold in all the compared sequences, and become candidate sets with the possibility of making maximal contiguous sequence. That is, the maximal contiguous sequence satisfying the minimum support threshold are made up of the above sequences. It results from the fact that any super patterns consisting of nonfrequent sequence patterns cannot be frequent in the Apriori algorithm.

Next, binding and expanding sequences with a possibility of concatenating each other in frequent contiguous sequences with length-4 produce the candidate set of the maximal contiguous sequence. In order to bind sequences, like <ATCG> and <TCGT>, they can be concatenated and expanded the same as <ATCGT> in the case when the first-A-excluded remainder from <ATCG> and the last-T-excluded remainder from <TCGT> accord with each other. The result becomes <ATCGT>, <TCGTG>, <TGATT>, <CGTGA>, <CATCG> and <GTGAT>, and then the process is recursive in the same way to expand until it is impossible to expand further. Tab. 2 shows the whole candidate set of the maximal item group finally obtained through the process. Tab. 2 predicts the sequences able to expand maximally from a contiguous subsequence with length-4.

Table 2. Candidate Subsequences

Length 4	Length 5	Length 6	Length 7
ATCG TCGT TGAT CGTG CATC GTGA GATT	ATCGT TCGTG TGATT CGTGA CATCG GTGAT	ATCGTG TCGTGA CATCGT CGTGAT GTGATT	CGTGATT CATCGTG ATCGTGA TCGTGAT

Table 3. Actual Sequences Satisfying Minimum support threshold

Length 4	Length 5	Length 6	Length 7
ATCG TCGT TGAT CGTG CATC GTGA GATT	ATCGT TCGTG TGATT CGTGA CATCG GTGAT	ATCGTG TCGTGA CATCGT CGTGAT GTGATT	CGTGATT ATCGTGA

Table 3 is the result of examining the actual sequences satisfying the specified minimum support threshold. It can be confirmed that there is no big difference between candidate sequence in Table 2(gradually expanding from subsequences with length-4 satisfying support threshold) and the actual subsequence satisfying the minimum support threshold in Table 3. Therefore, it can be known through expansion that the resultant candidate subsequence is very similar to the actual subsequence, and that it is possible to analogize the maximal candidate sequence through it. Comparison of Table 2 and Table 3 can confirm that the method of producing the candidate sequence is considerably exact.

The finally-produced candidate set is confirmed through real database searches from the maximal candidate sequences. Then, the produced results make it unnecessary to search a shorter candidate set. As a result, the real frequent maximal contiguous sequence in sequence database of Table 1 is <ATCGTGA> and <CGTGATT>.

The fixed-length spanning tree of the proposed algorithm can meet various minimum support thresholds very effectively and flexibly. Especially, when the minimum support threshold is more than 5, the proposed method can find a maximal contiguous sequence only by tree search, without producing a candidate subset.

IV. PERFORMANCE EVALUATION

This section presents that performance evaluation reveals the excellence of the proposed algorithm. Before that, we explain the problems of the projected databases separately produced for searching contiguous subsequences in the existing methods.

1. The Problems of Projected Database

The projected database generally has a relatively large capacity, compared with the original sequence database. To prove this fact, the projected database was produced through the following sequence data, and was compared in its size. In Table 4, after each projected database of the original sequence data of different size made, their data size is compared. The used data are a randomly-produced DNA sequence data and the actual DNA sequence data, with data size of 5,000 bytes and 8,932 bytes, respectively. In the result of producing the projected database of each item (A, C, G and T) from each sequence, there is a difference in the size of data. Fig. 5 compares the differences shown in Table 4 by using a graph.

Table 4. Projected Database Size

Data Type	Original Data	A-Projected Data	C-Projected Data	G-Projected Data	T-Projected Data
Random Sequence (Byte)	5000	66529	61286	64734	59901
DNA sequence (Byte)	8932	1034951	920142	967499	1015148

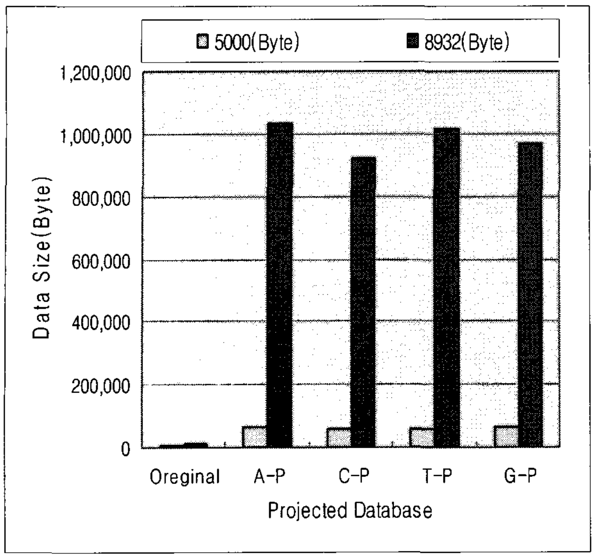

Fig. 5. Size of Projected Databases

Table 4 and Fig. 5 show that the larger the capacity of the original sequence data, the larger the exponential increase in the capacity of the projected database. That makes production of the projected database very expensive which has a major effect on the deterioration of retrieval performance. The longer the length of the sequence becomes, the more the capacity of the projected database increases. Therefore, in order to solve the problem, this paper has improved retrieval performance by making possible frequent contiguous subsequences without producing the projected database.

2. Retrieval Performance and Used Memory Amount

Fig. 6 shows the excellence of the retrieval performance of the proposed algorithm by comparing it with the retrieval performance of the existing MacosVspan algorithm. As for the MacosVspan algorithm, as mentioned in the related research, the projected database of each item has to be produced, and the spanning tree has to be constructed recursively several times, in order to retrieve the maximal common subsequence from long sequence data. However, the proposed algorithm needs only one tree construction and one database scan. But, as the proposed algorithm(in the case of the predicted candidate set) includes false positive data, so it has to confirm its real existence through database scan finally. However, the more the fixed length is expanded, the smaller the produced candidate sequence becomes; so this false positive data can be greatly reduced. The fact can be confirmed in Fig. 6 where the proposed algorithm enables the change of the fixed length to bring the change of retrieval performance. As shown in Fig. 6, the algorithms with a fixed length-7 has remarkably higher retrieval performance than those with a fixed length less than 7.

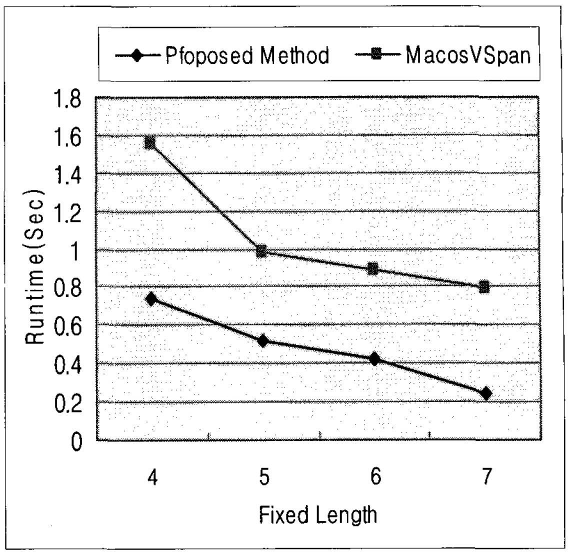

Fig. 6. Retrieval Performance According to Change of Fixed Length

Table 5 represents the used memory according to the change in fixed length. The tree was constructed by using the random data, while the fixed length changed from 5 to 10, and at the same time, the used memory amount was measured. Whenever the fixed length is expanded by about 1, twice as much memory is required. When the fixed length is 10, it can accept the amount of used memory, of 1Mbyte as sufficient.

Table 5. Memory Usage According to Change of Fixed Length

Fixed length Type	5	6	7	8	9	10
Memory Usage(Byte)	19112	64168	225692	535892	909892	1296474

Moreover, in the case that the length of the searched maximal common subsequence is shorter than or the same as the fixed length, we can obtain exact results at once, without an additional confirmation. In comparison with the existing method, as for the proposed method, the higher the frequent support threshold becomes, the more the candidate data lessens; it can result in much better retrieval performance.

Fig.7 compares the retrieval performance of the proposed method (according to minimum support threshold) with that of the existing method. A tree with a fixed length-5 was constructed using the above-mentioned random data. It is shown in Fig. 7 that when the minimum support threshold in 4, the difference in retrieval speed between the proposed method and the MacosVSpan is minimum.

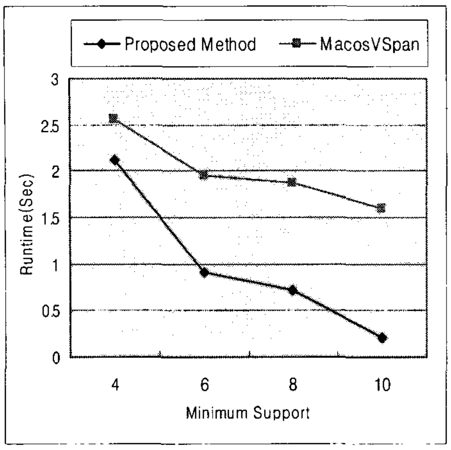

Fig. 7. Retrieval Performance According to Change of Minimum Support

Fig. 8 shows the used memory amount according to the change of sequence length when the number of DNA sequences in 100 and the length of the sequences is from 100 to 500. The data is used for the construction of a tree with a fixed length of 7. Fig. 8 shows that in the case of a relatively short sequence (length=100), the used memory amount of the proposed algorithm is larger than that of the existing method. The MascosVspan algorithm processes one projected database after another, and then produces, from the results, the last maximal search results by using a suffix tree, whereas the proposed algorithm requires slightly larger memory because it constructs the spanning tree by processing all of them at once. However, the above result appears in the case where the compared sequence is short. The MacosVspan algorithm needs large memory because the longer the sequence, the larger the suffix gets, while the fixed-length spanning tree of the proposed algorithm no longer increases after it reaches the necessary maximum.

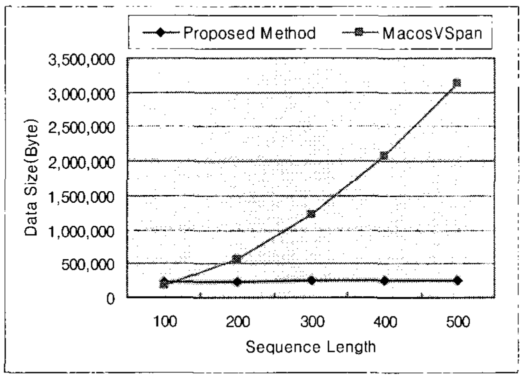

Fig. 8. Memory Usage According to Change of Sequence Length

3. Algorithm Analysis

The proposed algorithm has the following characteristics, compared with the previous researches. First, it can accept several values of minimum support threshold effectively by means of one time database access and construction of fixed-length spanning tree. Second, the longer the fixed length of the tree, the better the retrieval performance. Third, the proposed method can produce results only by tree search, without expansion for production of candidate set in the case of a high minimum support threshold and performance elevation according to fixed length. Fourth, the longer the sequence length becomes, the better performance it shows in comparison with any other algorithms. As a result, it can be applied not only to DNA sequence with a small number of items(dimension) but also amino acid sequence with a large number of items, and other multi-dimensional sequence data.

V. CONCLUSION

In this paper, we have proposed an algorithm to quickly and effectively process frequent maximal contiguous sequences, which are considered as very important in bioinformatics. Compared with the existing algorithms, the proposed method has improved the retrieval performance by greatly reducing the frequency of database accesses and removing the process of additional data production. In future research, we will optimize the proposed algorithm by considering a variety of environments with different parameters, such as the level of tree, various minimum support threshold, various lengths of sequence data and multi-dimensional sequence data, etc.

REFERENCE

[1] V. Chvatal and D. Sankoff , “Longest Common Subsequences of two random Sequences,” **Journal of Applied Probability**, Vol. 12, 1995, pp.306-315.

[2] R. Wanger and M. Fischer, “The string-to-string Correction Problem,” **Journal of ACM**, Vol. 21, No. 1, 1974, pp.168-173.

[3] D. Hirschberg, “Algorithms for the longest common subsequence problem,” **Journal of the ACM,** Vol. 24, No. 4, 1977, pp.664-675.

[4] S. Needleman and C. Wunsch, “A general Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins,” **Journal of Molecular Biology**, Vol. 147, 1970, pp.443-453.

[5] E.M. McCreight, “A space-economical suffix Tree construction algorithms” **Journal of the ACM**, Vol. 23, No. 2, 1976, pp.262-272.

[6] M. farach, “Optimal suffix tree construction with large alphabets,” **IEEE Symp. Found Computer Science**, 1997, pp. 137-143.

[7] R. Hariharan, “Optimal parallel suffix tree construction,” **IEEE Symp. Found Computer Science**, 1994, pp.290-299.

[8] R. Agrawal and R. Srikant, “Fast algorithms for mining association rules,” **Proceedings of VLDB**, 1994, pp.487-499.

[9] R. Srikant and R. Agrwal, “Mining Sequential Patternrs: Generalizations and performance improvements,” **Proceedings of EDBT**, 1996, pp.3-17.

[10] 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," **Proceedings of ICDE**, 2001, pp.215-214.

[11] Jin Pan, Peng Wang Wei Wang Baile Shi and Genxing Yang , "Efficient Algotithms for Mining Maximal Frequent Concatenate Sequences in Biological Datasets" **Procedings of Computer and Information Technology**, 2005, pp.98-104.

Tae Ho Kang
He received the B.S. degree in the department of Computer Communication Engineering from Howon University in 1999. And he received the M.S. degree in department of Computer and Industrial Engineering, Chungbuk National University in 2002. He is currently working towards PhD. degree on Bioinformatic system. His research interests also include Database System, Multimedia Database and Storage management system.

Jae Soo Yoo
He received the B.S. degree in Computer Engineering in 1989 from Chunbuk National University, Chunju, South Korea. And he received the M.S. and Ph.D. degrees in Computer Science in 1991 and 1995 from Korea Advanced Institute of Science and Technology, Taejeon, South Korea. He is now a professor in the department of Computer and Communication Engineering, Chungbuk National University, Cheongju, South Korea, where his research interests are the database system, multimedia database, distributed computing and storage management system.

Hak Yong Kim
He received the B.S. and M.S. degree in the departments of Agricultural Chemistry and Chemistry, Chungbuk National University in 1985 and 1987 respectively. He received the Ph.D. degree in the Molecular Cell Biology, University of Connecticut, U.S.A in 1994. He is now a professor in the Department of Biochemistry, Chungbuk National University, Cheongju, South Korea, where his research interests are the signal transduction, protein networks, and biodynamic model.

Byoung Yup Lee
He received the B.S. and M.S. degree in Computer Science in 1991 and 1993 from Korea Advanced Institute of Science and Technology, Taejeon, South Korea. And he received the Ph.D. degree in the Management Information Systems from Korea Advanced Institute of Science and Technology, Taejeon, South Korea in 1997. He is now a professor in the department of Electronic Commerce, Paichai University, Taejeon, South Korea, where his research interests are the datamining, XML, artificial intelligence, multimedia database, distributed computing.