社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－MPS－60 (3)
2006／6／26

可変長ワイルドカード領域を持つ極小な頻出配列パターンの抽出

加 藤 智 之[†] 北 上 始[††] 高 木 允[†]
田 村 慶 一[††] 黒 木 進[††] 森 康 真[††]

著者らは，アミノ酸配列データベースからモチーフの候補である頻出パターンを抽出するために，極小な可変長ワイルドカード領域を持つ頻出パターンを導き出す方法を提案する．この方法では，k-頻出パターン (長さ k の頻出パターン) から (k+1)-頻出パターンを生成するパターン成長アプローチを拡張し，k-頻出パターンごとにスコープデータベースを作成する．スコープデータベースの有効性を示すために，PROSITE から Leucine Zipper モチーフを含むデータセットを取り出し，可変長の頻出パターンを抽出する能力の評価を行ったので，その結果について報告する．

Mining of Frequent Sequential Patterns with Minimum Cover on Variable-Length Wildcard Regions

TOMOYUKI KATO,[†] HAJIME KITAKAMI,[††] MAKOTO TAKAKI,[†]
KEIICHI TAMURA,[††] SUSUMU KUROKI[††] and YASUMA MORI [††]

We propose a method for extracting frequent sequential patterns with minimum variable-wildcard regions, in order to extract candidates of a motif from amino acid sequence databases. A scope database defined by each frequent k-length pattern is constructed by the extension of Projected Database that generate frequent (k+1)-length patterns from a frequent k-length pattern in pattern growth approach. Moreover, we report experimental results that our extended method was evaluated using a dataset that includes the Leucine Zipper motif.

1. Introduction

We relate to the analysis of sequence databases in the field of bioinformatics. The focus in the field has been on extracting the candidates of motifs from the sequence databases. Each one of these motifs that has been discovered by biologists has a particular sequence pattern related to one function of some protein. The motifs discovered by many biologists appear in PROSITE [1] [2] and Pfam [3] and are regarded as a protein function that has been conserved in the process of molecular evolution.

We have proposed a pattern extraction method [5] [6] based on the prefix-projected pattern-growth approach [4]. The literature [5] reports the fast extraction of frequent patterns with fixed-length wildcard regions in the application of a projected database. In the pattern-growth approach, the extraction of frequent (k+1)-length patterns is achieved by adding all pairs of a wildcard region together with an alphabet letter to a frequent k-pattern. The literature [6] reports the extraction of variable-length wildcard regions instead of the above fixed-length wildcard regions in the same approach.

However, new problems arise in the variable-length pattern-extraction method.

(1) Variable-length wildcard redions without minimum cover
Any variable-length wildcard region in the frequent pattern does not represent the minimum cover for occurrences corresponding to the evidence of the frequent pattern.

(2) Redundant frequent patterns
The redundant frequent patterns, in which the wildcard regions located at the same place are dif-

[†] 広島市立大学大学院
Graduate School of Information Sciences, Hiroshima City University
[††] 広島市立大学
Faculty of Information Sciences, Hiroshima City University

ferent from each other, are extracted. For example, if two different wildcard regions, $x(i_1,j_1)$ and $x(i_2,j_2)$, in two patterns, $\langle A\text{-}x(i_1,j_1)\text{-}F\rangle$ and $\langle A\text{-}x(i_2,j_2)\text{-}F\rangle$, have a relation $[i_1,j_1]\subset[i_2,j_2]$, the former pattern must be removed because it is redundant for the latter pattern.

In order to solve the problems, we propose a new methodology for extracting non-redundant frequent sequential patterns that are constructed by variable-length wildcard regions with the minimum cover in the prefix-projected pattern-growth approach.

2. Definitions and statement of the problem

A sequence database is denoted DB, and $DB=\{t_1,t_2,\cdots,t_n\}$. Any element t_i is represented by the form $\langle sid, s_i\rangle$, where $1\leq i\leq n$, "n" is the number of elements, and "sid" is a sequence identifier. The set of sequence identifiers in the sequence database is represented as $\Omega=\{1,2,3,\cdots,n\}$. Each s_{sid} is defined as a sequence for which the sequence identifier has the value of sid. The j^{th} letter from the head of the sequence s_{sid} is represented as $s_{sid}[j]$, where $1\leq j\leq\|s_i\|$. Table 1 represents a $DB=\{t_1,t_2,t_3,t_4,t_5\}$ and $\Omega=\{1,2,3,4,5\}$, where $t_1=\langle 1, FKYAKWLCDN\rangle$, $t_2=\langle 2, SFVKTAEHNQC\rangle$, $t_3=\langle 3, ALR\rangle$, $t_4=\langle 4, MSKPL\rangle$, and $t_5=\langle 5, FSKFLMAWEH\rangle$. When we realize that the character "A" is included in such elements as t_1, t_2, t_3 and t_5, we can then find $s_1[4]=s_2[6]=s_3[1]=s_5[7]=$"$A$". If the number of alphabet letters for a sequence has a value of k, the sequence is called a k-length sequence. For example, the first sequence in Table 1 is a 10-length sequence.

Finite sub-sequences composed of a letter of the alphabet and the wildcard letter (hereafter, called the wildcard) are called a string, where both ends of the string have to be letter of the alphabet. The wildcard is a sign that shows an arbitrary letter of alphabet. If the number of letters in a string has the value of k, the string is called a k-string. For example, a string $\langle F{*}K{*}A\rangle$ including the wildcard letter is a 3-string and a string $\langle FLMA\rangle$ without the wildcard letter is a 4-string.

2.1 Relationship between a pattern and occurrence

The pattern is a form which represents a set of k-strings occurring more than one time in sequences of the database DB, where $k\geq 1$. The set of k-occurrences is defined as a ternary relation $\{(k\text{-}string, i, j)|k\text{-}string\ existing\ in\ the\ j^{th}\ position, \langle i, s_i\rangle\in DB\}$. For $\alpha_i\in\sum$ and $1\leq i\leq k$, a k-length pattern $\langle pat^k\rangle$ with k letters of the alphabet $\sum$ is a way to represent a set of k-occurrences and has the following form:

$$\langle pat^k\rangle=$$
$$\langle\alpha_1\text{-}x(i_1,j_1)\text{-}\alpha_2\text{-}x(i_2,j_2)\text{-}\cdots\text{-}x\ (i_{k-1},j_{k-1})\text{-}\alpha_k\rangle \quad (1)$$

The symbol $x(i,j)$ in expression (1) denotes that the number of wildcard signs in the wildcard region has a range from i to j, where $0\leq i\leq j$. The "-" symbol means that the next element is continued. Sometimes, this symbol is omitted. If $i<j$, then this region is called a variable-length wildcard region. If $i=j$, it is the same as a fixed-length wildcard region, and this region can be represented as $x(i)$. The expression (1) is equal to '$*$'. Henceforth, $\epsilon=j-i$ is called the error count of region $x(i,j)$. It is clear that $x(i,j)$ can be represented as $x(i,i+\epsilon)$. If all wildcard regions included in a k-pattern are fixed-length wildcard regions, it is called a fixed pattern or rigid pattern. If a k-pattern has at least one variable-length wildcard region, it is called a variable-length pattern.

2.2 Statement of the problem

Our goal is to extract all the frequent sequential patterns from a sequence database that satisfy the input parameters, such as the minimum support count (denoted $mini\_sup$), the maximum wildcard count (denoted wc_{max}), and the maximum error count (denoted ϵ_{max}), given by the user. The maximum wildcard count, wc_{max}, is defined as the maximum length for each of the wildcard regions

表 1 例の配列データベース
Table 1 An example of a sequence database

sid	sequence
1	FKYAKWLCDN
2	SFVKTAEHNQC
3	ALR
4	MSKPL
5	FSKFLMAWEH

included in the frequent patterns, where any wildcard region $x(i,j)$ in the frequent patterns satisfies the relation $0 \leq i \leq wc_{max}$. When the set P_k of frequent k-patterns has m elements, P_k is shown as follows:

$$P_k = \{\langle pat_1^k \rangle : cnt_1, \langle pat_2^k \rangle : cnt_2, \cdots, \langle pat_m^k \rangle : cnt_m\} \qquad (2)$$

where $\langle pat^k \rangle$:cnt denotes a k-pattern $\langle pat^k \rangle$ with a support count, "*cnt*". When we define a set P as all the frequent patterns extracted from the sequence database, the set P is shown as $P_1 \cup P_2 \cup \ldots \cup P_q$, where q corresponds to the maximum length of the extracted frequent pattern.

3. Existing method

The existing method finds the wildcard region together with one letter added to a frequent k-pattern from the sequence database whenever a set of the (k+1)-pattern is generated by the growth of a frequent k-pattern. The letter of the alphabet found by the method exists between the start and end scan positions in each sequence including the region. In order to avoid scanning useless subsequences, the start scan position must be the position next to the place where the rightmost letter of the frequent k-pattern is in the sequence data.

It is convenient to make the projected database of each k-pattern to find the start scan position efficiently for the scanning. The projected database $PDB(\langle pat^k \rangle)$ for k-pattern $\langle pat^k \rangle$ is as follows.

$PDB(\langle pat^k \rangle) = \{(i,j) |$ *The rightmost letter of each k-string included in the set of occurrences represented as* $\langle pat^k \rangle$ *is located at the* $(j\text{-}1)^{th}$ *position of* $s_i \in DB$, $1 \leq j \leq \|s_i\|\}$ (3)

When we apply the variable-length pattern-extraction method [6] to the sequence database shown in Table 1, we can obtain the enumeration tree shown in Figure 1, where *mini_sup*, wc_{max}, and ϵ_{max} have a value of 3.

Consider the extraction of $\langle F\text{-}x(0,3)\text{-}K\text{-}x(1,3)\text{-}A \rangle$ in Figure 1. The variable-length wildcard region between $\langle F \rangle$ and $\langle K \rangle$ is represented as $x(0,3)$, which does not represent the minimum cover for the set of 3-occurrences, {(⟨F-x(0)-K-x(1)-A⟩,1,1),(⟨F-x(1)-K-x(1)-A⟩,2,2),(⟨F-x(0)-K-x(1)-A⟩,5,1)}. The minimum variable-length wildcard region between them must be $x(0,1)$, which is deduced from s_1, s_2,

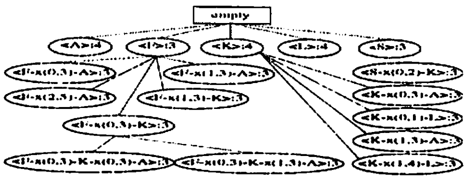

図 1 可変長パターン抽出法により抽出されるパターン
Fig. 1 Patterns extracted by variable-length pattern-extraction method

and s_5 in Table 1. Thus, the projected database does not have any capability to deduce the minimum variable-length wildcard region between $\langle F \rangle$ and $\langle K \rangle$ when both $x(1,3)$ and $\langle A \rangle$ are extracted by the pattern growth of $\langle F\text{-}x(0,3)\text{-}K \rangle$ using the database.

In addition, when we focus on the variable-length wildcard regions between $\langle F \rangle$ and $\langle K \rangle$ in Figure 1, the number of wildcards between $\langle F \rangle$ and $\langle K \rangle$ in 2-pattern including the both is 0 and 3, and these wildcards exist on the 2-occurrences found from s_1, s_2, and s_5. In other words, the minimum variable-length wildcard region is $x(0,3)$. A region $\langle F\text{-}x(1,3)\text{-}K \rangle$ in Figure 1 is redundant to the region $\langle F\text{-}x(0,3)\text{-}K \rangle$.

4. Proposed method

The construction of a scope database is being proposed as a solution to the previously stated problems in the prefix-projected pattern-growth approach. When the set of (k+1)-patterns are generated by the growth of a k-pattern, all the minimum and non-redundant wildcard regions included in the (k+1)-patterns and their support count can be computed from the scope database defined by the k-pattern. The scope database defined by a k-pattern $\langle pat^k \rangle$ is defined as follows:

$SDB(\langle pat^1 \rangle, [r, r+\epsilon_{max}]) =$
$\{(nil, i, j, wc, s_i[j_{new}]) |$ *One letter of* $\langle pat^1 \rangle$ *is located at* $(j\text{-}1)^{th}$ *position from the head of* $s_i \in DB$, $wc \in [r, r+\epsilon_{max}]$, $j_{new} = j + wc$, *and* $1 \leq j \leq \|s_i\|$, $j_{new} \leq \|s_i\|\}$ (4)

$SDB(\langle pat^k \rangle, [r, r+\epsilon_{max}]) =$
$\{(List^k, i, j_{next}, wc_{next}, s_i[j_{new}]) | (List^{k-1}, i, j, wc, \alpha) \in SDB_{\alpha}(\langle pat^{k-1} \rangle, [r, r+\epsilon_{max}])$, $List^k = Append(List^{k-1}, [wc])$, $wc_{next} \in [r, r+\epsilon_{max}]$, $s_i \in DB$, $j_{next} = j + wc + 1$, $j_{new} = j_{next} + wc_{next}$, $1 \leq j \leq \|s_i\|$,

$j_{next} \leq \|s_i\|$, and $j_{new} \leq \|s_i\|\}$ (5)
where $k \geq 2$, $\langle pat^k \rangle = \langle pat^{k-1}\text{-}x(r, r+\epsilon)\text{-}\alpha \rangle$, $\epsilon \leq \epsilon_{max}$, and *Append*$(X, Y)$ indicates adding list Y to list X.
$SDB_\alpha(\langle pat^k \rangle, [r, r+\epsilon_{max}]) =$
$\{(List^k, i, j, wc, \alpha) | (List^k, i, j, wc, \alpha) \in SDB(\langle pat^k \rangle,$
$[r, r+\epsilon_{max}])\}$ (6)

$List^1$ is an empty list because a 1-string does not have an adjoined letter. For $k \geq 2$, $List^k$ consists of (k-1)-elements, and each element stores the number of wildcards included in each of (k-1) wildcard region. The (k-1)-elements in $List^k$ are ordered by the address located in the sequence. For example, $List^3$ of 3-string $\langle F{*}K{*}{*}{*}A \rangle$ is represented as [1, 3].

The scope database $SDB_\alpha(\langle pat^k \rangle, [r, r+\epsilon_{max}])$ constructed from both the frequent k-pattern $\langle pat^k \rangle$ and the sequence database includes the set of (k+1)-occurrences corresponding to the evidence of the (k+1)-pattern with the rightmost letter α. Therefore, the scope database defined by the minimum variable k-pattern $\langle pat^k \rangle$ is useful in finding all minimum variable-length wildcard regions located in $\langle pat^{k+1} \rangle$ together with one alphabet letter α from the range $[r, r+\epsilon_{max}]$, starting at the position next to the rightmost letter of the variable k-pattern $\langle pat^k \rangle$ whenever we extract variable (k+1)-patterns by the growth of the variable k-pattern $\langle pat^k \rangle$.

4.1 Operation of the scope database

To make all $\langle pat^{k+1} \rangle$ from $\langle pat^k \rangle$, it is necessary to make $\langle pat^{k+1} \rangle$ by using the scope database constructed from $\langle pat^k \rangle$. We prepared one operation for the construction of the scope database and three operations for the use of one. We describe each operation as follows.

(Operation1) Construct the scope database $SDB(\langle pat^k \rangle, [r, r+\epsilon_{max}])$ from the expressions (4) and (5) when a wildcard count "r" as an input parameter of the operation is selected from $[0, wc_{max}]$.

(Example1) Consider the application of this operation to Table 1, where $mini\_sup = 3$, $wc_{max} = 3$, and $\epsilon_{max} = 3$. For $k = 1$, the set of frequent 1-patterns is $\{\langle A \rangle{:}4, \langle F \rangle{:}3, \langle K \rangle{:}4, \langle L \rangle{:}4, \langle S \rangle{:}3\}$. If we then set the wildcard count "r" to 0 for extracting frequent 2-patterns with the prefix $\langle F \rangle$, the scope becomes [0,0+3]. Therefore, the scope database of the 1-pattern $\langle F \rangle$ is as follows:
$SDB(\langle F \rangle, [0, 3]) = \{$
$(nil,1,2,0,\langle K \rangle), (nil,1,2,1,\langle Y \rangle), (nil,1,2,2,\langle A \rangle),$
$(nil,1,2,3,\langle K \rangle), (nil,2,3,0,\langle V \rangle), (nil,2,3,1,\langle K \rangle),$
$(nil,2,3,2,\langle T \rangle), (nil,2,3,3,\langle A \rangle), (nil,5,2,0,\langle S \rangle),$
$(nil,5,2,1,\langle K \rangle), (nil,5,2,2,\langle F \rangle), (nil,5,2,3,\langle L \rangle),$
$(nil,5,5,0,\langle L \rangle), (nil,5,5,1,\langle M \rangle), (nil,5,5,2,\langle A \rangle),$
$(nil,5,5,3,\langle W \rangle)\}$ (7)

(Operation2) Generate all candidate (k+1)-patterns $\langle pat^{k+1} \rangle$ corresponding to the child node by the growth of the k-frequent pattern $\langle pat^k \rangle$ corresponding to the parents' node of the enumeration tree. Let us consider the selection of the value of "r" from $[0, wc_{max}]$ in ascendant order. In order to compute the k^{th} wildcard region included in the candidate (k+1)-pattern with the rightmost letter α from $SDB_\alpha(\langle pat^k \rangle, [r, r+\epsilon_{max}])$, we begin to compute the minimum value r_{min} and maximum value r_{max} for $\{wc \mid (List, i, j, wc, \alpha) \in SDB_\alpha(\langle pat^k \rangle, [r, r+\epsilon_{max}])\}$. If $r_{min} = r$, we generate the candidate (k+1)-pattern $\langle pat^k\text{-}x(r_{min}, r_{max})\text{-}\ \alpha \rangle$. After that, we check the redundancy between the (k+1)-pattern and those (k+1)-patterns with the same α which have already been extracted from the same k-pattern $\langle pat^k \rangle$. If the (k+1)-pattern is redundant with the other (k+1)-patterns, it is removed. If $r_{min} > r$, we do not generate the candidate (k+1)-pattern.

(Example2) Consider the application of this operation to $SDB(\langle F \rangle, [0, 3])$ under the same conditions as in Example 1 for r=0. We can obtain r_{min}=0, r_{max}=3 from $SDB_K(\langle F \rangle, [0, 3]) = \{(nil,1,2,0,\langle K \rangle), (nil,1,2,3,\langle K \rangle), (nil,2,3,1,\langle K \rangle), (nil,5,2,1,\langle K \rangle)\}$. Thus, we can obtain a candidate 2-pattern $\langle F\text{-}x(0, 3)\text{-}K \rangle$ with the prefix $\langle F \rangle$, as $r_{min}=r=0$ and $r_{max}=3$.

(Operation3) Compute the support count, "cnt", of each candidate (k+1)-pattern, $\langle pat^{k+1} \rangle$ using the set $SDB_\alpha(\langle pat^k \rangle, [r, r+\epsilon_{max}])$ that is a subset of the scope database $SDB(\langle pat^k \rangle, [r, r+\epsilon_{max}])$. The support count of $\langle pat^{k+1} \rangle = \langle pat^k\text{-}x(r, r+\epsilon)\text{-}\alpha \rangle$ is computed by enumerating all of the elements that belong to the set $\{i|$ $(List,i,j,wc,\alpha) \in SDB_\alpha(\langle pat^k\rangle, [r, r+\epsilon_{max}])\}$ of identifiers included in $SDB_\alpha(\langle pat^k\rangle, [r, r+\epsilon_{max}])$. If the support count of $\langle pat^{k+1}\rangle$ satisfies the minimum support count given by the user, $\langle pat^{k+1}\rangle$ becomes the frequent $(k+1)$-pattern.

(Example3) Consider the application of this operation to $SDB_K(\langle F\rangle, [0,3])$ under the same conditions as in Example 2. We can obtain the set $\{1, 2, 5\}$ of identifiers included in $SDB_K(\langle F\rangle, [0,3])$ in order to extract the candidate 2-pattern $\langle F\text{-}x(0,3)\text{-}K\rangle$ with the variable-length wildcard region $x(0, 3)$ and the letter "K" from Table 1. This results in support count = 3 and the frequent 2-pattern $\langle F\text{-}x(0,3)\text{-}K\rangle$:3 can be extracted.

(Operation4) For $k\geq 2$, when a frequent $(k+1)$-pattern $\langle pat^{k+1}\rangle$ with a prefix $\langle pat^k\rangle$ is represented as $\langle pat^k\text{-}x(r, r+\epsilon)\text{-}\alpha\rangle$ and $\epsilon\geq\epsilon_{max}$, each variable-length wildcard region $x(i, j)$ that exists in k-prefix $\langle pat^k\rangle$ of $\langle pat^{k+1}\rangle$ is re-computed by using $List^k$ of $SDB_\alpha(\langle pat^k\rangle, [r, r+\epsilon_{max}])$ and looking for the minimum variable-length wildcard region $x(i', j')$. After it re-computed, the relation of $j'\leq j$ is approved to the variable-length wildcard region. If as many as one variable-length wildcard region that approves the relation of $i'\neq i$ exists, to prevent redundant regions from being generated, remove the $(k+1)$-frequent pattern.

(Example4) When a frequent 3-pattern $\langle F\text{-}x(0,3)\text{-}K\text{-}x(1,3)\text{-}A\rangle$:3 is generated from SDB_A $(\langle F\text{-}x(0,3)\text{-}K\rangle, [1,4]) = \{([0],1,3,1,\langle A\rangle),([1],2,5,1,\langle A\rangle),([1],5,4,3,\langle A\rangle)\}$ of Example 1, we can update the variable-length wildcard region $x(0,3)$ of the 3-pattern using Operation 4. Each of these lists, [0], [1], and [1], included in $SDB_A(\langle F\text{-}x(0,3)\text{-}K\rangle, [1,4])$, represents the number of wildcards placed between two letters, "F" and "K". Therefore, the first wildcard region $x(0,3)$ in the frequent 3-pattern $\langle F\text{-}x(0,3)\text{-}K\text{-}x(1,3)\text{-}A\rangle$:3 is updated to $x(0,1)$. As a result, we can obtain a minimum 3-frequent pattern $\langle F\text{-}x(0,1)\text{-}K\text{-}x(1,3)\text{-}A\rangle$:3.

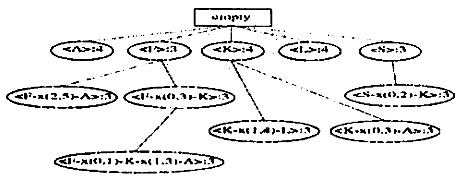

図 2 スコープデータベースにより抽出されるパターン
Fig. 2 Patterns extracted by scope database

5. Performance evaluation

The performance evaluation was achieved by comparing the existing variable-length pattern-extraction method with the proposed method using the scope database. The computer environment used for the evaluation was Intel PIV-2.4GHz, with 2GB memory, 2GB SWAP memory, 74.5GB HDD, and Microsoft Windows XP Professional as the operating system.

The dataset used in the performance evaluation was provided by PROSITE and includes the Leucine Zipper motif. The dataset has PS00036 as a registration number and 125 sequences. The form of the Leucine Zipper motif is $\langle[KR]\text{-}x(1,3)\text{-}[RKSAQ]\text{-N-}x(2)\text{-}[SAQ](2)\text{-}x\text{-}[RKTAENQ]\text{-}x\text{-R-}x\text{-}[RK]\rangle$. The input parameters to extract the above motif from this sequence database were the minimum support rate, wc_{max}, and ϵ_{max}. The first parameter had values of 37, 36, 30, and 25 percent as the minimum support rate. The later two parameters had a value of 2. Table 2 shows the results of the evaluation. Table 2 shows the number of frequent patterns extracted by the existing method and the proposed method for each minimum support rate.

It is clear in the table that the extracted frequent patterns extracted by the proposed method were fewer than those extracted by the existing method. One reason is that the proposed method never extracted any patterns with redundant variable-length wildcard regions that were extracted by the existing method. Another reason is that the proposed method removed the redundant patterns produced by minimizing variable-length wildcard regions in the pattern-growth approach.

Moreover, the existing method broke off the computation due to a lack of memory when the support

表 2 Leucine Zipper モチーフを含むデータセットの計算結果
Table 2 Calculation result of a dataset that includes the Leucine Zipper motif

Comparison item / Minimum support rate		37%	36%	30%	25%
Existing Method	Number of frequent patterns	56821	64182		
	Calculation time (sec)	11.7	12.8		
Proposed Method	Number of frequent patterns	51739	57540	203077	700845
	Calculation time (sec)	92.6	101.5	468.4	2378.0

ratio was under 36 percent. Since the proposed method extracted fewer patterns than the existing method, it succeeded in extracting the frequent patterns until the support rate of 25 percent was reached.

The evaluation yielded the following results. The proposed method can extract the patterns with minimum cover and non-redundant wildcard regions and frequent sequential patterns using a smaller support count than that required by existing method. Moreover, the proposed method resulted in a high capability to extract non-redundant sequential patterns including minimum variable-wildcard regions.

6. Conclusion

In this paper, we have proposed a method for extracting frequent sequential patterns with minimum and non-redundant variable-wildcard regions in order to extract candidates of a motif from the sequence databases.

A scope database for each frequent variable length k-pattern is defined as an extension of a projected database generates frequent variable length (k+1)-patterns from a frequent k-length pattern in the prefix-projected pattern-growth approach. The scope database for a frequent k-length pattern consists of not only the existing projected database of the pattern but also the range of the scan and occurrences of the pattern.

The prototype has been applied to the evaluation of a dataset of sequences that included the Leucine Zipper motif. Our method had a higher capability to extract sequential patterns including both minimum and non-redundant variable-wildcard regions than the existing method.

Acknowledgment

This work was made possible in part by the support of a Grant-in-Aid for Scientific Research (C) (2) (No. 17500097) from the Japanese Society for the Promotion of Science and a Hiroshima City University Grant for Special Academic Research (general research fund - code number 3106).

References

1) Amos. Bairoch, Philipp. Bucher, and Kay. Hofman: The PROSITE Database. Its Status in 1995, Nucleic Acids Research, Vol. 24, pp.189-196, 1996.
2) PROSITE. http://kr.expasy.org/prosite
3) Erick.L.L. Sonnhamer, Sean. R. Eddy, and Richard. Durbin: Pfam: A Comprehensive Database of Proteins, Vol. 28, pp.405-420, 1997.
4) Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, and Helen Pinto: PrefixSpan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth, Proc. of International Conference on Data Engineering (ICDE 2001), IEEE Computer Society Press, p.215-224, 2001.
5) Hajime Kitakami, Tomoki Kanbara, Yasuma Mori, Susumu Kuroki, and Yukiko Yamazaki: Modified PrefixSpan Method for Motif Discovery in Sequence Databases, Proc. of the 7th Pacific Rim International Conference on Artificial Intelligence, Springer-Verlag, pp.482-491 , August 2002.
6) Shigetaka Tono, Hajime Kitakami, Keiichi Tamura, Yasuma Mori, Susumu Kuroki: Efficiently Mining Sequence Patterns with Variable-Length Wildcard Regions using an Extended Modified PrefixSpan Method, The 13th Annual International Conference on Intelligent Systems for Molecular Biology (ISMB2005), Poster No. G22, p.147, June 2005.