Extraction of Ambiguous Sequential Patterns with Least Minimum Generalization from Mismatch Clusters

Kotaro Araki Keiichi Tamura Tomoyuki Kato Yasuma Mori Hajime Kitakami
Graduate School of Information Sciences, Hiroshima City University,
3-4-1 Ozukahigashi, Asaminami-ku, Hiroshima, 731-3194, Japan
{kotaro,ktamura,kato,mori,kitakami}@db.its.hiroshima-cu.ac.jp

Abstract

An ambiguous query in sequence databases returns a set of similar subsequences, called a mismatch cluster, to the user. The inherent problem is that it is difficult for users to identify the characteristics of very large similar subsequences in a mismatch cluster. In order to support user comprehension of mismatch clusters, it is important to extract a set of ambiguous sequence patterns with the least minimum generalization in the mismatch cluster. The extraction of the ambiguous sequential pattern set requires an enormous amount of computational time, since we have to discover generalized patterns with minimum covers for the mismatch cluster from candidate generalized patterns.

The present paper proposes an iterative refinement method to extract ambiguous sequence patterns with minimum cover for a mismatch cluster selected from a sequence database and proposes to use the method together with a domain segmentation method to achieve an efficient pattern extraction. Moreover, a prototype implementing the two proposed methods has been applied to three datasets included in PROSITE in order to evaluate their usefulness. The proposed methods resulted in a high capability to extract ambiguous sequential patterns from mismatch clusters that are provided by an ambiguous query in the sequence database.

1. Introduction

Ambiguous Query means similarity search in the sequence databases and is important in many application domains, such as finding similar molecular subsequences called motifs, finding patients whose lung lesions have similar evolution characteristics, detecting stocks that have similar growth patterns, and so forth. The motifs discovered by many biologists appear in PROSITE [1] and Pfam and are regarded as a protein function that has been conserved in the process of molecular evolution. A great deal of research [2] [3] [4] [5] [6] has been carried out on ambiguous queries in the fields of computer science and bioinformatics with the goal of finding similar subsequences in sequence databases. Each ambiguous query includes a length-k query subsequence (query k-subsequence), defined as a query backbone, and the number r of allowable mismatches with respect to the query k-subsequence.

A large number of similar k-subsequences, collectively referred to as a mismatch cluster, can be obtained from the sequence databases by a query given by the users. However, a problem arises that the user finds it difficult to understand the characteristics of very large similar subsequences in the returned mismatch cluster. To support the user’s comprehension of the mismatch cluster, the extraction of the set of ambiguous sequence patterns with the least minimum generalization is important. The extraction of this set requires a large computational time, since the time complexity required to find the set from candidate sequence patterns is $O(2^l k)$, where l is the size of the candidate ambiguous letter domain, and k is the number of ambiguous letter positions located at the ambiguous sequence patterns.

In order to efficiently extract ambiguous sequence patterns with the least minimum generalization, the following methods are here proposed:

(1) Iterative refinement

This method may be used to refine any generalized sequence patterns using negative information. First, the method is applied to compute the set of negative generalized patterns without subsuming any element in the mismatch cluster by refining the most general patterns. The method is then applied to the computation of the set of positive generalized patterns without subsuming the negative generalized patterns computed by refining the most general patterns. The positive generalized patterns are ambiguous sequential patterns with the least minimum generalization for the mismatch cluster.
(2) Optimization using domain segmentation method

We use the iterative refinement method together with the domain segmentation method. This method performs the segmentation of each ambiguous letter domain appearing in the most general sequential pattern generated by the mismatch cluster if a distance matrix of alphabet characters is defined as the user’s background knowledge. Domain segmentation is carried out using the distance matrix, and an attempt is made to avoid the generation of meaningless ambiguous patterns.

The remainder of the present paper is organized as follows. A discussion of related research is presented in Section 2. The terminology using in the present paper is described in Section 3. The proposed method is described in Section 4. An outline of the performance evaluation is reported in Section 5, and the results of the present study are summarized in Section 6.

2. Related Work

Many studies [2] [3] of ambiguous query processing for sequence databases are being carried out in the field of computer science. However, these studies focus on the development of methods to find subsequences that satisfy query conditions from sequence databases, and little attention is given to methods that organize these subsequences as ambiguous sequence patterns. Recently, the focus of studies on ambiguous query processing has shifted to the determination of motifs and frequent sequence patterns in the field of bioinformatics and data mining. Studies focusing on the determination of motifs and frequent sequence patterns have produced new algorithms, such as SPELLER [4], WINNOWER [5], and MITRA [6].

The above previous studies [2] [3] [4] [5] [6] focused on finding possible k-subsequences within a mismatch count given by user. However, they were not concerned with the computing of an ambiguous sequence pattern expression that appears in PROSITE [1] from all possible k-subsequences. A method for extracting ambiguous sequential patterns from a mismatch cluster referred to in previous studies as a clique is proposed as a solution to the above problems.

3. Definitions and Statement of the Problem

A sequence database is denoted as \( DB \), and \( DB = \{ t_1, t_2, t_3, \ldots, t_n \} \). Any element \( t_i \) is represented by the form \(< sid, s_i >\), 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 \( \Omega = \{ 1, 2, 3, \ldots, n \} \). 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 \( t_1 = < 1, FKYA\text{K}WLC\text{D}N >, t_2 = < 2, SFVK\text{TA}EH\text{N}QC >, t_3 = < 3, ALR >, t_4 = < 4, MSK\text{PL} >, \) and \( t_5 = < 5, FSK\text{F}M\text{A}WEH > \). The letters of the first position in elements \( t_1, t_2, t_3, t_4, \) and \( t_5 \) are represented as \( s_1[1] = "D", s_2[1] = "S", s_3[1] = "A", s_4[1] = "M", s_5[1] = "F" \) respectively. If the number of alphabet letters for a sequence has a value of \( k \), the sequence is called a length-k sequence or, merely, a \( k \)-sequence. For example, the first sequence in Table 1 is a 10-length sequence, or a 10-sequence.

3.1. Ambiguous sequence pattern

For an alphabet \( \Sigma_i \subseteq \Sigma \) where \( 1 \leq i \leq k \), a length-k pattern \(< pat^k \) with k alphabet letters \( \Sigma \) can be used to represent the set of k-instances and has the following form:

\[
< pat^k > = < \Sigma_1 - x(i_1, j_1), \Sigma_2 - x(i_2, j_2), \ldots, \Sigma_k - x(i_k, j_k) > \tag{1}
\]

The alphabet \( \Sigma_i \) in Equation (1) is called an ambiguous letter domain, and the position of \( \Sigma_i \) in the equation denotes the allocation of any alphabet letter included in the domain \( \Sigma_i \), where \( ||\Sigma_i|| \geq 2 \) and \( 1 \leq i \leq k \). Equation (1) is called a ambiguous k-sequence pattern if there is at least one ambiguous domain \( \Sigma_i \) such that \( ||\Sigma_i|| \geq 2 \) in \( 1 \leq i \leq k \). The “...” symbol means that the next element is continued; however, this symbol is sometimes omitted. The symbol \( x(i, j) \) in Equation (1) denotes the number of wildcard signs in the wildcard region ranges from i to j, where \( 0 \leq i \leq j \). 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 \( x(1) \) is equal to the wildcard “*”. The wildcard indicates an arbitrary alphabet letter included in the alphabet \( \Sigma \). The expression \( x(0) \) denotes an empty string.

Consider the following Krringle motif:

\[
< [FY] - C - [RH] - [NS] - x(7, 8) > = < [FY] - C > \tag{2}
\]
\[ \text{FOR} \text{example}, \text{EV AL}(\langle \Sigma_1\Sigma_2\cdots\Sigma_{i-1}\{\alpha_1\}\Sigma_{i+1}\cdots\Sigma_k \rangle) = \text{EV AL}(\langle \Sigma_1\Sigma_2\cdots\Sigma_{i-1}\Sigma_i\cdots\Sigma_k \rangle) - \text{EV AL}(\langle \Sigma_1\Sigma_2\cdots\Sigma_{i-1}\{\alpha_i\}\Sigma_{i+1}\cdots\Sigma_k \rangle) \] 

\[ \text{For example, EV AL}(\langle [AB][CD] \rangle) \text{ generates all possible instances } \{< AC >, < AD >, < BC >, < BD > \}. \] 

The left-hand side \( \text{EV AL}(\langle [AB][CD] - \{D\} \rangle) \) of Equation (5) becomes \( \text{EV AL}(\langle [AB]C \rangle) = \{< AC >, < BC > \} \), and the right-hand side \( \text{EV AL}(\langle [AB]C \rangle) - \text{EV AL}(\langle [AB]D \rangle) \) of the equation becomes \( \{< AC >, < BC > \} \). Therefore, the left-hand side is equivalent to the right-hand side in Equation (5). Hereinafter, the ambiguous \( k \)-sequence pattern subsuming several instances is called a generalized \( k \)-sequence pattern or simply a generalized \( k \)-pattern in the sequence database.

### 3.3. Set of generalized patterns

Consider a set \( I^k \) of \( k \)-instances that are length-\( k \) subsequences originating in the sequence databases. For \( 1 \leq j \leq k \), we define an alphabet \( \Sigma_j \) as follows:

\[ \Sigma_j = \{\text{inst}[j] | \text{inst} \in I^k\} \] 

The generalized \( k \)-pattern \( \langle \Sigma_1\Sigma_2\cdots\Sigma_{i-1}\Sigma_i\cdots\Sigma_k \rangle \) constructed by Equation (6) is called the most general \( k \)-pattern \(< \text{mgpat}^k >\) for the set \( I^k \) of \( k \)-instances. Moreover, the set of generalized \( k \)-patterns with minimum cover for \( I^k \) is called a set of least minimum generalized \( k \)-patterns.

Consider the mismatch cluster \( \text{MIS} = \{< \text{mis}_1^k >, < \text{mis}_2^k >, \ldots, < \text{mis}_m^k > \} \) including \( m \) \( k \)-instances. If the least minimum generalized \( k \)-patterns to never subsume any element of \( \text{MIS} \) are computed by refining the most general \( k \)-pattern \(< \text{mgpat}^k >\), these patterns are called negative generalized \( k \)-patterns. The method of computation is described in the next section. If generalized \( k \)-patterns to never subsume any of the negative generalized \( k \)-patterns are computed by refining the most general pattern \(< \text{mgpat}^k >\), these patterns are called positive generalized patterns, i.e., the set of least minimum generalized patterns for set \( I^k \) of \( k \)-instances.

### 3.4. Problem statement

Given a sequence database, a query \( k \)-subsequence, a number \( r \) of allowable mismatches compared to the query \( k \)-sequence, and a minimum support threshold (denoted \( \text{mini}_\text{sup} \)), for the mismatch cluster, the problem of ambiguous sequential pattern extraction is not only to find the mismatch cluster satisfying the support count from the sequence database but also to extract ambiguous sequential \( k \)-patterns with the support count from the mismatch cluster. The support count of a sequential \( k \)-pattern is defined as the number of distinct sequence identifiers in the set of sequences including the sequential \( k \)-pattern.

### 4. Extraction of Least Minimum Generalized Patterns

The least minimum generalized \( k \)-patterns are extracted from the set of candidate generalized \( k \)-patterns. We consider representing the most general \( k \)-pattern \(< \text{mgpat}^k >\) as \( \langle \Sigma_1\Sigma_2\cdots\Sigma_{i-1}\Sigma_i\cdots\Sigma_k \rangle \). Then, the most general \( k \)-pattern \(< \text{mgpat}^k >\) subsumes \((2^l - 1)^k - l^k\) elements, which correspond to both candidate generalized \( k \)-patterns and \( k \)-instances, where \( l = |\Sigma_i| \) and \( 1 \leq i \leq k \). The number of \( k \)-instances is \( l^k \), and the number of candidate generalized \( k \)-patterns is \((2^l - 1)^k - l^k\). Therefore, candidate \( k \)-patterns are included in a very large set.

In order to efficiently extract least minimum generalized patterns with minimum cover for the mismatch cluster, we developed the following process:

1. Find the mismatch cluster from the sequence database under the condition that the cluster satisfies the minimum support count and any element in the cluster is a \( k \)-subsequence within \( r \) mismatches, as compared to the query \( k \)-subsequence.
2. Generate the most general \( k \)-pattern from the mismatch cluster.
(3) Generate the set of segmented $k$-patterns from the most general $k$-pattern using the domain segmentation method.

(4) Apply the iterative refinement method as follows:

- The set of segmented $k$-patterns is refined to the set negative generalized $k$-patterns without subsuming any $k$-instance included in the mismatch cluster. As a result, the negative generalized $k$-patterns are represented as patterns with minimum cover for $EVAL(<\text{smaller patterns}>\{-k\text{-instance}\mid k\text{-instance}\in\text{the mismatch cluster}\}$.

- The set of segmented $k$-patterns is refined to the set of positive generalized $k$-patterns without subsuming any of the negative generalized $k$-patterns. The positive generalized $k$-patterns denote a generalized $k$-pattern minimum cover for $EVAL(<\text{smaller patterns}>\{-\text{EV AL}\}<\text{negative generalized } k\text{-patterns}>\}$.

Specifically, the positive generalized patterns correspond to the least minimum generalized patterns.

In this section, we first describe the computation of the minimum cover and the iterative refinement and domain segmentation methods.

### 4.1. Computation of the minimum cover

Consider a generalized $k$-pattern $<\text{pat}^k>$ and the $k$-instance set $I^k = \{<\text{inst}^1_{\text{pat}}>, <\text{inst}^2_{\text{pat}}>, \cdots, <\text{inst}^n_{\text{pat}}>\}$. Here, we compute a set of negative $k$-instances that is defined as $EVAL(<\text{pat}^k>) \setminus \{<\text{inst}^b_{\text{pat}}>\}$. The set of generalized $k$-patterns with minimum cover for the set of negative $k$-instances is computed by the following equation:

$$COVS(<\text{pat}^k>, <\text{inst}^k>) = \{<\Sigma_1\Sigma_2\cdots\Sigma_{i-1}(<\alpha_i)>\cdots\Sigma_k > | 1 \leq i \leq k\},$$

where,

$$<\text{inst}^k> = <\alpha_1 - \alpha_2 - \cdots - \alpha_{k-1} - \alpha_k> \quad (7)$$

For $\alpha_i \in \Sigma_i$ and $1 \leq i \leq k$, the set $COVS(<\text{pat}^k>, <\text{inst}^k>)$ is represented as the set of generalized $k$-patterns. For $\alpha_i \notin \Sigma_i$ and $1 \leq i \leq k$, the set $COVS(<\text{pat}^k>, <\text{inst}^k>)$ becomes $\{<\text{pat}^k>\}$.  

**Example 1**

Consider computing generalized 3-patterns $COVS(<\text{ABC}][\text{DEF}][\text{GHI}]>, <\text{ADG}>)$ with minimum cover for negative 3-instances $EVAL(<\text{ABC}][\text{DEF}][\text{GHI}]>)\{-<\text{ADG}>\}$. Equation (7) yields the following results:

$$COVS(<\text{ABC}][\text{DEF}][\text{GHI}]>, <\text{ADG}>) = (<\text{BC}][\text{DEF}][\text{GHI}]>, <\text{ABC}][\text{DEF}][\text{GHI}]>)$$

Consider replacing the $k$-instance $<\alpha_1 - \alpha_2 - \cdots - \alpha_{k-1} - \alpha_k>$ with a generalized $k$-pattern $<\Gamma_1\Gamma_2\cdots\Gamma_k>$ in Equation (7), where $\Gamma_i \subseteq \Sigma$ and $1 \leq i \leq k$. Then, the extension of the following equation is reliable:

$$COVS(<\Sigma_1\Sigma_2\cdots\Sigma_k>, <\Gamma_1\Gamma_2\cdots\Gamma_k>) = \{<\Sigma_1\Sigma_2\cdots\Sigma_{i-1}(<\alpha_i> \cdots \Sigma_k > | 1 \leq i \leq k\} \quad (8)$$

For $\Gamma_i \neq \Sigma_i$ and $\Gamma_i \cap \Sigma_i \neq \varphi$, the set $COVS(<\text{pat}^k>, <\text{inst}^b>)$ is represented as the set of generalized $k$-patterns, where $1 \leq i \leq k$. For $\Gamma_i = \Sigma_i$ and $1 \leq i \leq k$, the set $COVS(<\text{pat}^k>, <\text{inst}^b>)$ becomes an empty set.

**Example 2**

Consider computing generalized 3-patterns $COVS(<\text{ABC}][\text{DEF}][\text{GHI}]>, <\text{BC}][\text{DEF}][\text{GHI}]>)$. Equation (8) yields the following result:

$$COVS(<\text{ABC}][\text{DEF}][\text{GHI}]>, <\text{BC}][\text{DEF}][\text{GHI}]>) = (<\text{ABC}][\text{DEF}][\text{GHI}]>)$$

### 4.2. Iterative refinement method

Supposing a set $S$ with $n$ elements, such as instances and generalized patterns, the refinement for the most general $k$-pattern $<\text{mpat}^k>$ is carried out using the function $COVS(<\text{the set of generalized patterns with the refinement of } i - 1 \text{ times}>, <\text{the } i^{th} \text{ element included in } S>)$ described previously. For $i = 1$, $COVS(<\text{mpat}^k>, <\text{the first element in } S>)$ generates a set $G_1$ of more refined $k$-patterns without subsuming the first element. For $i = 2$, $COVS(<\text{each } k\text{-pattern in } G_1>, <\text{the } 2^{nd} \text{ element in } S>)$ generates a part of a more refined $k$-pattern set $G_2$ without subsuming the first and second elements. We can obtain the set of generalized $k$-patterns without subsuming any of the $n$ elements in $S$ after conducting iterative refinement of $n$ times.

**Example 3**

Consider computing generalized 3-patterns $COVS(<\text{ABC}][\text{DEF}][\text{GHI}]>, <\text{BC}][\text{DEF}][\text{GHI}]>)$. When Equation (8) is iteratively applied to $<\text{ABC}][\text{DEF}][\text{GHI}]>)$, we obtain the following results:
The set of least minimum generalized k-patterns for the mismatch cluster \( MIS = \{ < mis^k_1, < mis^k_2, \ldots, < mis^k_m > \} \) including k-subsequences is computed by the following two-step procedure:

1. Extract the set of negative generalized k-patterns to equal a minimum cover for the negative instances \( EVAL(< mgpat^k_1 >) – MIS \). The extraction is achieved by refining the most general k-pattern \( mgpat^k > \) using \( MIS \).
2. Extract the set of positive generalized k-patterns to equal a minimum cover for the positive instances \( MIS \). The extraction is achieved by refining the most general k-pattern \( mgpat^k > \) using \( MIS \) set with minimum cover for the negative instances.

Consider the set of k-patterns \( \{ < pat^k_1 >, < pat^k_2 >, \ldots, < pat^k_n > \} \) generated by Step (1) or Step (2). If a k-pattern \( < pat^k_1 > \) subsumes another k-pattern \( < pat^k_2 > \), then the set is redundant. Therefore, each step removes the k-pattern \( < pat^k_1 > \) from the set in order to eliminate redundancy in the set. For example, consider the set of 3-patterns, \( \{ < D[BE][CF] >, < DE[CF] >, < D[BE][C] > \} \). Both \( < DE[CF] > \) and \( < D[BE][C] > \) are redundant with the first pattern \( < D[BE][CF] > \). The redundant elimination results in the set \( \{ < D[BE][CF] > \} \).

**Example 4**

Suppose that the most general pattern \( mgpat^3 > \) is represented as \( [AD][BE][CF] \) and a mismatch cluster \( MIS = \{ < ABF >, < AEC >, < AEF >, < DBF >, < DEC >, < DEF > \} \). Consider computing the positive generalized length-3 patterns after computing the negative generalized length-3 patterns.

- The first element \( < ABF > \) included in \( MIS \) is applied to the refinement of \( [AD][BE][CF] \), and the refinement gives the following negative generalized length-3 patterns:
  \[ COVS([AD][BE][CF] >, < ABF >) = ([D[BE][CF] >, < [AD][BE][CF] >, < [AD][BE][C] >] \]

- The second element \( < AEF > \) in \( MIS \) is applied to the refinement of the first pattern \( [D[BE][CF] > \), the second pattern \( [AD][BE][CF] > \), and the third pattern \( [AD][BE][C] \) included in the above, and the resulting refinement gives the following negative generalized 3-patterns:
  \[ COVS([AD][BE][CF] >, < AEF >) = ([DE[CF] >, < [AD][BE][CF] >, < [AD][BE][C] >] \]

- The third element \( < DEC > \) in \( MIS \) is applied to the refinement of the second 3-pattern \( < [AD][BE][CF] > \), the result becomes empty.

- The fourth element \( < DBF > \) in \( MIS \) gives the following set of 3-patterns:

- The fifth element \( < DEC > \) in \( MIS \) is applied to the refinement of \( < DE[CF] > \) and \( < D[BE][C] > \), both results become empty.

- The computation of the last element \( < DEF > \) is unnecessary since it is never subsumed by any residual generalized patterns.

In order to summarize the above processing, we obtain the negative generalized 3-pattern set \( [AD][BC] \). Figure 1 illustrates the above processing.

- The negative generalized 3-pattern set \( [AD][BC] \) gives the following set of positive 3-patterns:
  \[ COVS([AD][BE][CF] >, [AD][BC] >) = ([AD][BE][CF] >, [AD][BE][F] >] \]

Therefore, the set \( [AD][BE][CF] >, [AD][BE][F] >] \) becomes the positive least minimum generalized 3-pattern set with minimum cover for the mismatch cluster \( MIS = \{ < ABF >, < AEC >, < AEF >, < DBF >, < DEC >, < DEF > \} \).

### 4.3. Optimization using domain segmentation method

In order to reduce the computational time of the iterative refinement method with the worst case time complexity \( O(2^k) \), we propose to use the iterative refinement together with the domain segmentation method. The domain segmentation method is a method to decompose each ambiguous letter domain included in the most general pattern on the condition that application domain knowledge is defined.
as a distance matrix representing the dissimilarity among alphabet letters. Since the ambiguous letter domain of size \( l \) can be decomposed into small sub-domains by the domain segmentation method, the most general pattern is decomposed into a set of generalized patterns specialized by the method. The decomposition results in reducing the computational time of the iterative refinement method. The proposed processing algorithm to use the iterative refinement together with the domain segmentation method is as follows:

(1) Classify alphabet letters included in the each ambiguous letter domain of it using an agglomerative hierarchical clustering algorithm and decompose the most general pattern into a set of sub-domains using the classified alphabet letters.

(2) Decompose the most general pattern into a set of generalized patterns using the each set of sub-domains.

(3) Extract generalized patterns with the minimum cover of the mismatch cluster using the iterative refinement method, whose input data are generalized patterns and the mismatch cluster.

5. Performance Evaluation

An evaluation experiment was conducted using three datasets, Ribosomal with PS00051 as a registration number, Kringle with PS00021, and Zinc Finger with PS00028 included in PROSITE to confirm the effectiveness of the iterative refinement method and the domain segmentation method. The computer environment used for the evaluation was a 3.20-GHz Intel Pentium(R)4 with 2 GB of memory, 2 GB of SWAP memory, a 320 GB HDD, and CentOS V4.5 as the operating system. The characteristics of the three datasets in PROSITE are shown in Table 2. The three motifs appearing in the datasets are represented as follows:

(1) Ribosomal : \(<KRS> - <PTKS> - x(3) - [LIVMFG] - x(2) - [NHS] - x(3) - R - [DNHY] - W - R - [RS] >\)
(2) Kringle : \(<FY> - C - [RH] - [NS] - x(7, 8) - [WY] - C >\)
(3) Zinc Finger : \(< C - x(2, 4) - C - x(3) - [LIVMFWYC] - x(8) - H - x(3, 5) - H >\)

Beforehand, each dataset is stored in the disk-based suffix tree, called DynaCluster [7], which we have implemented as a tool to evaluate the proposed method. The mismatch cluster for an ambiguous query given by a user is first selected from the disk-based suffix tree stored with each dataset including sequences.

The performance evaluation was achieved by the following comparisons : (1) ambiguous query results and generalized patterns, (2) iterative refinement results with the domain segmentation method and without the domain segmentation method. In order to achieve the comparisons, we executed the iterative refinement with the domain segmentation method and without the domain segmentation method for subsequences included in a mismatch cluster selected by an ambiguous query. For the domain segmentation method, we used the substitution matrix PAM250 of the amino acid, and adopted the furthest neighbor method as the hierarchical clustering algorithm.

5.1. Experimental results

(1) Comparison of ambiguous query results with generalized patterns

Table 3 shows the input parameter, query subsequences, ambiguous query results and the iterative refinement method with the domain segmentation method. Comparing ambiguous query results and generalization results in Table 3, clarifies that the iterative refinement method successfully achieved fewer generalized patterns than the large number of elements in the mismatch cluster. For each dataset, the set of generalized patterns covers the set of elements in the mismatch cluster obtained by the ambiguous query. Generalized patterns extracted by the iterative refinement method

<table>
<thead>
<tr>
<th>Table 2. Characteristics of the three datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dataset</td>
</tr>
<tr>
<td>-----------------</td>
</tr>
<tr>
<td>Ribosomal</td>
</tr>
<tr>
<td>Kringle</td>
</tr>
<tr>
<td>Zinc Finger</td>
</tr>
</tbody>
</table>

Figure 1. An example of extracting the negative patterns from \{<ABF>, <AEC>, <AEF>, <DBF>, <DEC>, <DEF>\}
would provide more accurate information than merely inspecting all elements included in the mismatch cluster.

In order to verify the possibility of knowledge discovery, we computed the support count rate ranking of generalized patterns extracted by the iterative refinement method with the domain segmentation method. The experimental results obtained using the Ribosomal dataset are shown in Table 4. The iterative refinement method with the domain segmentation method extracts 48 generalized patterns, but some generalized patterns with high ranking include several solutions evaluated from the motif patterns. Therefore, this method allows us to access generalized patterns, including a part of the motif patterns, when we focus generalized patterns with high ranking. The other datasets such as the Kringle and Zinc Finger datasets, also yield the same experimental results.

(2) Comparison of the iterative refinement method with the domain segmentation method and without the domain segmentation method

Table 5 shows the results of the iterative refinement method without the domain segmentation and with the domain segmentation method. Table 5 shows that the domain segmentation method was faster than processing without the domain segmentation method. In the experiment using Kringle dataset, the execution of the iterative refinement without the domain segmentation method never terminated for three or more days. On the other hand, the execution of the iterative refinement method with the domain segmentation method resulted in high speed. Then, the most general pattern that had been computed from the mismatch cluster had 20 distinct amino acid letters in every ambiguous letter domain $\Sigma_i$, where $1 \leq i \leq 6$. Applying the domain segmentation method, each ambiguous letter domain was divided into \{C\}, \{I, L, M, V\}, \{D, E, H, Q\}, \{F, W, Y\}, \{K, N, R, S\}. The elimination of the combination of letters among segments is the reason that the segmentation method achieved high speed. In particular, if each ambiguous letter domains $\Sigma_i$ has several ambiguous letters, the effect of the domain segmentation method is quite remarkable.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Ambiguous Query including subsequences and mismatch count</th>
<th>Query results (Records)</th>
<th>Generalization result (Records)</th>
<th>Decrease rate (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ribosomal</td>
<td>$&lt;K - P - x(3) - L - x(2) - N - R - D - W - R - R &gt;$</td>
<td>6</td>
<td>109</td>
<td>48</td>
</tr>
<tr>
<td>Kringle</td>
<td>$&lt;Y - C - R - N - x(7, 8) - W - C &gt;$</td>
<td>4</td>
<td>3552</td>
<td>1152</td>
</tr>
<tr>
<td>Zinc Finger</td>
<td>$&lt;C - x(2, 4) - C - x(3) - L - x(8) - H - x(3, 5) - H &gt;$</td>
<td>1</td>
<td>2729</td>
<td>102</td>
</tr>
</tbody>
</table>

The iterative refinement with the domain segmentation method divided the generalized pattern into the following patterns:

- $<C - x(2) - C - x(3) - [ACFILMVY] - x(8) - H - x(5) - H >$: 8.99%.
- $<C - x(2) - C - x(3) - [AST] - x(8) - H - x(5) - H >$: 2.36%.

One is the generalized pattern that motif pattern subsumed into the Zinc Finger motif, and the other is the generalized pattern that junk pattern excluded from the Zinc Finger motif.

For both generalized patterns, the letters included in each domain have a strong relationship with each other. Therefore, the domain segmentation method would be effective for discovering new motif patterns, since the distance matrix representing dissimilarity that is needed for the method requires background knowledge in the application field.

6. Conclusion

We have proposed an iterative refinement method to extract ambiguous sequence patterns with minimum cover for
Table 4. Support count rate ranking of generalized patterns

<table>
<thead>
<tr>
<th>Rank</th>
<th>Pattern</th>
<th>Support count(%)</th>
<th>Number of instances</th>
<th>Number of solutions evaluated from the motif</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>$&lt; [KR] - T - x(3) - [IV] - x(2) - N - x(3) - R - [HN] - W - R - R &gt;$</td>
<td>56.10</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>2</td>
<td>$&lt; K - T - x(3) - [IMV] - x(2) - N - x(3) - R - H - W - R - R &gt;$</td>
<td>27.27</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>$&lt; [KR] - T - x(3) - V - x(2) - [HN] - x(3) - R - H - W - R - R &gt;$</td>
<td>20.45</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>$&lt; K - T - x(3) - V - x(2) - [HNS] - x(3) - R - H - W - R - R &gt;$</td>
<td>20.45</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>$&lt; P - [FL] - x(3) - L - x(2) - A - x(3) - K - Q - N - R - R &gt;$</td>
<td>11.36</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>6</td>
<td>$&lt; K - Q - x(3) - V - x(2) - W - x(3) - [KR] - T - N - R - R &gt;$</td>
<td>6.28</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>$&lt; R - [ST] - x(3) - I - x(2) - N - x(3) - R - H - W - R - R &gt;$</td>
<td>4.55</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 5. Performance of the iterative refinement method for two conditions

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Non-segmentation</th>
<th>Domain segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Number of generalized patterns</td>
<td>CPU time (seconds)</td>
</tr>
<tr>
<td>Ribosomal</td>
<td>45</td>
<td>3605.2</td>
</tr>
<tr>
<td>Kringle</td>
<td>--</td>
<td>--</td>
</tr>
<tr>
<td>Zinc Finger</td>
<td>38</td>
<td>2.03</td>
</tr>
</tbody>
</table>

a mismatch cluster selected from a sequence database, and proposed to use the method together with a domain segmentation method to achieve an efficient pattern extraction. A performance evaluation indicated a reduction in the number of subsequences in the mismatch cluster and faster extraction.

Future studies will describe the use of extensions of the proposed method to process variable wildcard regions and the development of a faster method for use with a very large mismatch cluster selected from the sequence database.

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 by a Grant-in-Aid for Young Scientists (B) (No.18700094) from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

References