Sequential Pattern Mining with Constraints on Large Protein Databases

Joshua Ho, Lior Lukov, Sanjay Chawla

School of Information Technologies, University of Sydney
Sydney NSW 2006 Australia
joshua.ho@student.usyd.edu.au, lior@it.usyd.edu.au, chawla@it.usyd.edu.au

Abstract

Sequential pattern mining in protein databases has the potential of efficiently discovering recurring structures that exist in protein sequences. This in turn may provide an understanding of the functional role of proteins which support such structures. In this paper we generalize a well known sequential pattern mining algorithm, SPAM [1], by incorporating gap and regular expression constraints along the lines proposed in SPIRIT [2]. However the advantages of using a depth-first algorithm like SPAM is that (a) it allows us to push the constraints deeper inside the mining process by exploiting the prefix antimonotone property of some constraints, (b) It uses a simple vertical bitmap data structure for counting and (c) it is known to be efficient for mining long patterns.

Our work onextending SPAM is motivated by its role in two concrete applications: (1) as a "feature factory" for a secondary structure classification problem in integral membrane proteins and (2) as an underlying engine for answering secondary stucture queries on protein databases. A detailed set of experiments confirm that the incorporation of gap and regular expresssion constraints allows us to extract more specific and biologically relevant information.

1 Introduction

In the post-genomic era scientists are constantly facing the challenge of effectively managing, querying and mining useful information from the vast amouont of biological sequence data that is now readily available. Analysis and knowledge retrieval from large biological sequence database is a common theme in many bioinformatics application, like gene finding, sequence homology search and protein secondary structure prediction.

In this paper, we concentrate on two distinct but related biological applications that involve mining of large protein databases: extraction of features for accurate transmembrane helix prediction and answering secondary structure queries. We propose that these two problems can be solved efficiently using sequential pattern mining techniques that incorporate user-specified constraints. We begin with a brief overview of the two bioinformatics problem followed by a review of the different sequential pattern mining algorithms that are available.

1.1 Extracting Transmembrane Helix Features

Integral Membrane Proteins (IMPs) control a broad range of events essential to the proper functioning of cells, tissues and organisms, and are the most common target of prescription drugs [3]. Despite their biological importance, there are only a small number of IMPs whose 3D structure information is available. Predicting the secondary structure of IMPs using machine learning techniques is an active area of research within bioinformatics. IMPs consist of transmembrane helix regions and non-transmembrane regions. The machine learning task is to identify the location of these regions based on primary structure of proteins, i.e., the given sequence of amino acids. This makes the identification of helical regions as an instance of the sequential classification problem. In our previous work [4] we have used the framework of Conditional Random Fields (CRFs) [5] to predict the location of transmembrane helical regions.

The abstract secondary structure classification problem can be stated as follows. We denote with x the given observation sequence of amino acids and with y the corresponding unobserved sequence from the set $\{1, 0\}$ where 1 denotes a transmembrane helical region and 0 a non-transmembrane region. The relationship between the sequence x and y is modeled

International Conference on Management of Data
COMAD 2005b, Hyderabad, India, December 20–22, 2005

by the conditional probability formula

$$P(y|x) = \frac{1}{Z} \exp(\sum_{i=1}^{n} \alpha_i F_i(x,y)) \qquad (1)$$

The success of the classification task is crucially dependent on the availability of a "good" set of features $F_i^\prime s$. The $F_i^\prime s$ capture the relevant biological information which characterize the properties of transmembrane helical regions. The $F_i^\prime$ can also be viewed as *constraints* on the conditional probability distribution $P(y|x)$. *We want to use sequential pattern mining in order to discover which constraints or features dominate the different secondary structure regions.* Thus we see sequential pattern mining with constraints as a "Feature Factory" which will give biologists the flexibility to generate candidate features and then test their effectiveness through the model specified in Equation 1.

1.2 Querying Protein Secondary Structure Database

The secondary structure is the local three dimensional structure of a protein sequence. There are mainly three type of secondary structures: alpha helix, beta sheet and loop region. The secondary structures is important in determining the 3D structure and the function of a protein [6].

The problem of posing secondary structure query on protein databases was described by Hammel and Patel [7]. The problem essentially states that given a protein database with each amino acid labeled with their secondary structure, find all the sequences that match a particular secondary structure pattern. A labeled protein sequence is of the following format:

```
GQISDSIEEKRGFFSTKRKKIEESDSSTTRKR...
hllllllllllhhheeeelllllllllleeeee...
```

The label **h** stands for alpha helix, **e** for beta sheet and **l** for loop region. The task is to retrieve all sequences that matches a query secondary structure pattern. An example of valid secondary structure query is <l 3 5><? 0 *><h 6 6>, which finds protein that contain a helix of length 3 to 5, followed at some point (with any number of gap) a loop of length 6. Hammel and Patel [7] make use of some segmentation technique to break up secondary structure labels into segments to allow for fast query. Currently, performance and flexibility are the main issues in devising a query engine. One challenge is to allow mixture of primary and secondary structure in the same query. In this paper, we present a sequential pattern mining approach to tackle this problem.

1.3 Sequential Pattern Mining

Both of the biological problems involve mining some kind of patterns in a large protein database. Therefore, we propose the use of the well characterized sequential pattern mining apporach to solve the problems. The support-confident sequential pattern mining problem was originally proposed by Agrawal and Srikant [8] in 1995. The problem was originally described for market basket analysis for customer transaction database. Since then, a number of Apriori [9] based algorithms have been invented [10, 2, 1, 11, 12]. GSP [10] generalized the problem by introducing the idea of time constraint, sliding time window and taxonomy. Subsequently, more researches have been done on pushing user-specified constraints into the sequence mining process to reduce the search space and gain output-selectivity. Ng *et al* [13] have explored the antimonotonic and succinct constraints and demonstrated how they can be used to prune the search space. SPIRIT [2] was the first family of algorithms that pushes regular expression constraints in sequential pattern mining. Constraints with prefix-monotone property was exploited in the context of mining with projected databases [11]. Sequential pattern mining for biological sequences has been explored by Wang *et al* [14]. They have deviced a scalable algorithm for mining general biological sequences (DNA and protein). However, they did not consider their algorithm in terms of solving specific biological problems and this is what we address.

One of the simplest and fastest algorithms for sequential pattern is called SPAM [1] which make use of a depth-first search strategy, efficient candidate pruning mechanism as well as a vertical bitmap data structure. Despite the space inefficiency, SPAM was shown to outperform other sequential pattern mining algorithms in runtime especially on large databases with long sequential patterns. Since runtime performance is essential for mining large protein databases and memory availability is usually not a big problem for most of the current databases, SPAM was chosen. The problem with SPAM is that it does not support pattern mining with user-specified constraints, which is essential for both of our biological applications. In transmembrane helix features extraction, being able to specify the minimum and maximum gap for output feature pattern is essential. The ability to specify regular expression constraints is also helpful in selecting desired features' pattern. In querying secondary structure on protein databases, the regular expression constraint is used for posing queries.

1.4 Our Contribution and Paper Organization

There are two main contributions of this paper. First, we have successfully incorporated the gap and regular expression constraints to SPAM. This modified SPAM algorithm is generic and can be applied to other sequential pattern mining problems. Second, we demonstrated the use of our modified SPAM algorithm to extract high quality transmembrane helix features as

	tid				
cid	1	2	3	4	5
1	{a,c}	{a}	{a}	{b}	{d}
2	{a}	{c}	{c}	{b}	{d}
3	{a}	{c}	{b}	{d}	
4	{a,b}	{b}	{b,c}	{c,d}	

Figure 1: Example of a transaction database

cid	sequence
1	({a,c},{a},{a},{b},{d})
2	({a},{c},{c},{b},{d})
3	({a},{c},{b},{d})
4	({a,b},{b},{b,c},{c,d})

Figure 2: Sequence view of the example transaction database

well as propose a framework for answering secondary structure queries using SPAM.

In particular, a three-phase framework for transmembrane helix features extraction using SPAM is implemented as a software called Pex-SPAM (**P**rotein Features **EX**tractor using **SPAM**). Our experimental results indicate that mining for transmembrane features with maximum gap constraint of five or below can significantly eliminate useless features.

The first part of the paper discusses how the gap and regular expression constraints are incorporated into SPAM. This includes section 2 which formally defines the problem of sequential pattern mining with constraints, and section 3 which describes how the constraints are pushed into SPAM. Section 4 includes a detailed discussion on how the modified SPAM is used to generate protein transmembrane helix features. The three-phase archtitecture of Pex-SPAM is discussed in this section. Section 5 presents experimental evaluation of the modified SPAM algorithm. It also demonstrates the use of Pex-SPAM to gain biological insight from the protein sequences. The proposed use of Pex-SPAM in protein secondary structure query is presented in section 6. The paper concludes in section 7 with a summary and directions for future research.

2 Problem Definition

Let $I = \{i_1, i_2, ..., i_n\}$ be the set of items. We call a subset $X \subseteq I$ an itemset. A sequence $s = (s_1, s_2, ..., s_m)$ is an ordered list of itemsets, where $s_i \subseteq I$, $i \subset \{i, ..., m\}$. A sequence $s_a = (a_1, a_2, ..., a_n)$ is contained in another sequence $s_b = (b_1, b_2, ..., b_m)$ if there exist integers $1 \leq i_1 < i_2 < ... < i_n \leq m$ such that $a_1 \subseteq b_{i_1}, a_2 \subseteq b_{i_2}, ..., a_n \subseteq b_{i_n}$. If sequence s_a is contained in sequence s_b, then we call s_a a subsequence of s_b and s_b a supersequence of s_a.

A transaction database D is a set of tuple (cid, tid, X) where cid is the customer id, tid is the transaction id which represent the number of the transaction made by a particular customer and X is the itemset. The composite key (cid, tid) is a primary key in the database. A transaction database can be viewed as sequence database (cid, s) where s is a sequence of itemset purchased by a customer in sequential order. Figure 1 is an example transaction database and Figure 2 is its equivalent sequence database. This example database is used throughout section 2 and 3 to motivate our solution. The support of a sequence s_a in a database D is defined as the percentage of sequences $s \in D$ that contains s_a and is denoted by $sup_D(s_a)$. For example, $sup_D(\{a\}\{c\}\{c\}) = 0.5$ because it is contained in half of the sequences (customer 2 and 4). Given a minimum support threshold *minSup*, a sequence s_a is called a *frequent sequential pattern* on D *if* $sup_D(s_a) \geq minSup$.

A more restrictive class of sequential patterns can be specified by including gap and regular expression constraints during the mining The gap constraints are also called time constraints in [10]. These constraints restrict the number of gap between sets of transactions that a frequent sequential pattern can have. Sequence $s_a =< a_1, \ldots, a_n >$ is a subsequence of $s_b =< b_1, \ldots, b_m >$ with *maxGap (minGap)* if there exist integers $j_1 < j_2 < \ldots < j_n$ such that $a_1 \subseteq b_{j_1}, a_2 \subseteq b_{j_2}, \ldots, a_n \subseteq b_{j_n}$ and $j_k - j_{k-1} - 1 \leq (\geq)$ *maxGap*(*minGap*).

A regular expression (*regex*) constraint R is expressed in terms of its equivalent finite state machine A_R. We adapt the following definitions from [2]. A sequence s is *legal* w.r.t. R *if* every state transition in A_R is defined following the sequence of transitions for the elements of s starting from the initial state of A_R. A sequence s is *valid* w.r.t. R *if* s is legal w.r.t. R and the final state of A_R is an *accept* state. In our problem, we say a sequence s satisfies a regex constraints R *if* s is valid with respect to R.

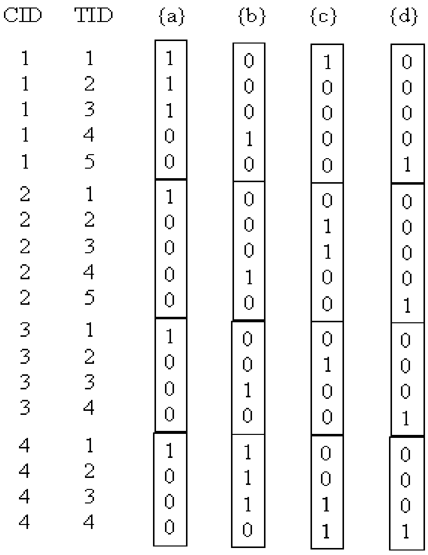
2.1 Problem Statement

SPAM with Constraints: Given a sequence database D, *minSup* and constraint set $C = \{$*minGap*, *maxGap*, *regex*$\}$, find all the sequences s which have $sup(s) \geq minSup$ and satisfy all constraints in C.

3 Applying Constraints to SPAM

3.1 Overview of SPAM

The main idea of SPAM is to combinatorially generate all candidate sequences in the manner of a depth first traversal through a lexicographic tree of sequences. Each node of the tree represents a frequent sequential pattern s_a found so far. Supersequences of s_a can be generated by the *sequence-extension step* (*S-step*) or the *itemset-extension step* (*I-step*). In S-step, a supersequence is generated by appending an itemset



CID	TID	{a}	{b}	{c}	{d}
1	1	1	0	1	0
1	2	1	0	0	0
1	3	1	0	0	0
1	4	0	1	0	0
1	5	0	0	0	1
2	1	1	0	0	0
2	2	0	0	1	0
2	3	0	0	1	0
2	4	0	1	0	0
2	5	0	0	0	1
3	1	1	0	0	0
3	2	0	0	1	0
3	3	0	1	0	0
3	4	0	0	0	1
4	1	1	1	0	0
4	2	0	1	0	0
4	3	0	1	1	0
4	4	0	0	1	1

Figure 3: The bitmap representation of the database

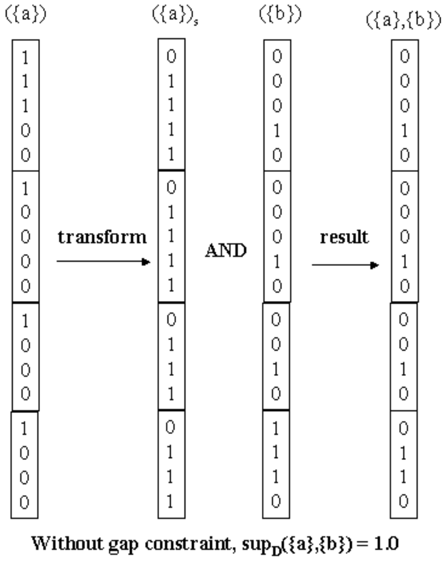


Figure 4: S-step with no gap constraints

with one item at the end of s_a. In I-step, a super sequence is generated by appending one item at the end of the last itemset of s_a. A newly generated sequence is pruned if its support is less than than minSup. In order to reduce the search space, two Apriori-based pruning techniques, called S-step pruning and I-step pruning, are used. In essence, both pruning techniques work like this: if sequence s is extended with item i which result in an infrequent sequence, SPAM will not extend item i to any supersequence of s because it must be infrequent based on the Apriori principal.

The most powerful feature of SPAM is its bitmap data structure for efficient support counting [1]. A vectical bitmap of item i represents the occurence of i in all transactions among all customers. Each vertical bitmap is partitioned into n sections where n is the numebr of customer. Each section contains m bits where m is the number of transactions made by that customer. Each bit contain a boolean value that is set to *one* if i appears in that transaction or *zero* otherwise. The support of a sequence is the number of sections in its vertical bitmap that contains at least one bit of *one*. Figure 3 shows the bitmap representation of the database in Figure 1.

3.2 Pruning Strategy

Constraints can be applied to prune out branches of the search tree which cannot produce frequent sequential patterns that satisfy the constraints. We observe that the sequences are built incrementally with each sequence in the direct parent node being the *prefix* [12] of the sequences of its child nodes. Therefore, we exploit the *prefix-antimonotonic* property of the constraints. A constraint c is prefix-antimonotonic *if* every sequence s satisfies c implies every prefix of s also satisfies c. This property is particularly important in pruning the search tree because a sequence s_a that cannot satisfy a prefix-antimonotonic property c implies none of the supersequence of s_a generated by either S-step or I-step can satisfy c because they all contain s_a as prefix. Gap contraints are enforced at the bitmap level during the S-step and regular expression constraint is applied by using its relaxed constraint which is prefix-antimonotonic.

3.3 Pushing Gap Constraints

The way in which the minGap and maxGap constraints can be pushed into SPAM is by providing a different transformation function for the S-Step at the bitmap level.

In the original SPAM algorithm, S-step for extending sequence $(\{a\})$ with sequence $(\{b\})$ involves two steps. First, let k to be the first index of *one* in every section of the vertical bitmap of $(\{a\})$, then set the bits position 1,2,...,k to be *zero* and every bit after position k to be *one*. The resulting bitmap after this step is called the *transformed bitmap* $(\{a\})_s$. In the second step, the transformed bitmap of $(\{a\})$ is ANDed with the vertical bitmap of $(\{b\})$. This operation yields the vertical bitmap of sequence $(\{a\},\{b\})$. This procedure is based on the observation that a sequential pattern can only be found if $\{b\}$ appear anywhere after the first occurrence of $\{a\}$. See Figure 4 for example.

Here we describe a way to push minGap and maxGap constraints into SPAM at the bitmap level. With minGap and maxGap constraints, the transformation step is modified to restrict the number of position that $\{b\}$ can appear after $\{a\}$. Essentially, for any position p with bit *one* in the original bitmap section of $\{a\}$, we transform only the bits between position $(p+minGap+1)$ to the bit at position $(p+maxGap+1)$ inclusively to *one* and all other bits are set to *zero*. If

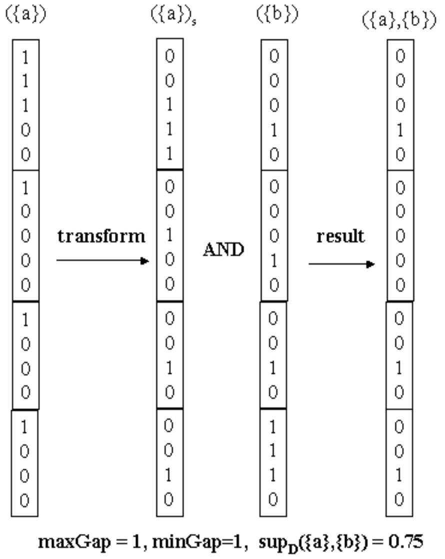


Figure 5: S-step with minGap=1 and maxGap=1

the maxGap is set ot infinity (no maxGap constraint), all bits between $(p + minGap + 1)$ till the end of the bitmap are set to *one*. If the transformation procedure encounters a bit that has been set to *one* (from the previous pass) and the current pass indicate that it should be set to *zero*, *one* is assigned to this bit because this position is valid for S-Step extension (See Fig. 5). This can be summerized in the following pseudocode.

```
Constraints-transform($s_a$)
$bm_a$ ←bitmap of $s_a$
$bm_a^{trans}$ ←empty bitmap
For each (bitmap section $bms_a \subseteq bm_a$)
    if ($maxGap = \infty$)
        $k$ ←index of the first 1
        transform bit 0 to (p+minGap) to be 0
        transform bit (p+minGap+1) till the end to be 1
    else
        $pset \leftarrow \phi$
        For ($i \leftarrow 1$ to $|bms_a|$)
            if ($bms_a[i] = 1$)
                $pset \leftarrow pset \cup i$
        For each ($p \in pset$)
            For ($i \leftarrow (p + minGap + 1)$ to
                $(p + maxGap + 1)$)
                $bms_a^{trans}[i] \leftarrow 1$
return $bm_a^{trans}$
```

As we can see, the gap constraints are enforced at the bitmap level in conjunction with support counting. Pei *et al* [11] proved that the gap constraints are antimonotomic, therefore they must also be prefix-antimonotonic. As a result, we can prune all nodes with sequences that do not satisfy the gap constraints because all its decendants must contain sequence that does not satisfy the gap constraints.

3.4 Pushing Regular Expression Constraints

In our discussion, we denote the regular expression (regex) constraint as R. It is easy to see that regex constraint are not prefix-antimonotonic. For example, the pattern ({a},{c},{b},{d}) is a valid sequence with respect to regular expression <a+[b|c]2d>. However, its prefix ({a},{c},{b}) is not valid w.r.t. the regular expression because the accept state in A_R cannot be reached by this sequence. However, we observe that the prefix ({a},{c},{b}) is legal w.r.t. R. Therefore, we propose the use of a *relaxed* constraint of R to prune the search tree when dealing with the regex constraint. Lets first suppose we have a constraint R' that is defined as:

Definition 1: Let R' be a constraint such that sequence s satisfies R' *if* s is legal w.r.t. R.

Next, we will show that R' is a relaxed constraint [2] of R. In general, consider constraint C' that generates a set of output sequences P', and constraint C that generates a set of output sequences P. We say C' is a *relaxed* constraint of C *iff* $P \subseteq P'$.

Lemma 1 *R' is a relaxed constraint of R.*

Proof: By definition, all sequences that is valid w.r.t. R must be legal w.r.t. R'. Therefore, all sequences generated under the constraint R must also be generated under constraint R'.

Now, lets examine the prefix-antimonocity of constraint R'.

Lemma 2 *R' is a prefix-antimonotonic constraint.*

Proof: If there exists a prefex α of a legal sequence s that is not legal w.r.t. R, there must be at least one transition in α that is not defined which implies the transition path of s must not be complete. This contradict the premise that s is legal. Therefore, sequence s that satisfies the constraint R', all prefix of s must also satisfies R'.

Using the nice prefix-antimonotonicity of R', we propose the following pruning strategy: at each s-step and i-step, prune all nodes containing sequence that violates the relaxed constraint R'. At the end of the tree building process, enforce the full regex constraint by traversing the search tree once more to retrieve the sequences that is valid w.r.t. R.

symbol	meaning
a,b,c	item
[a|b|c]	either one of a, b or c
a+	one or more occurence of a
a2	exactly two occurences of a

Table 1: Example of regular expression

During the tree building step in SPAM, the initial node is assigned with the initial state of the finite

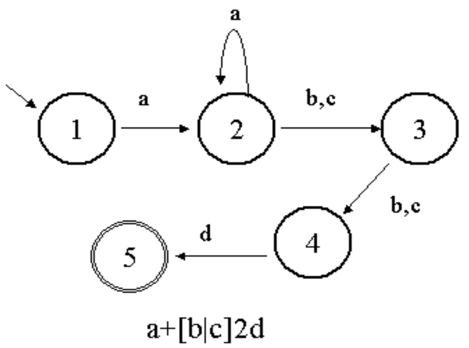


Figure 6: Finite State Machine for regular expression <a+[b|c]2d>

state machine A_R. For every possible S-step on sequence s_a with current state m the algorithm checks if the newly appended item $\{i\}$ can cause a transition $m \xrightarrow{i} n$ where n is a state in the A_R. If it can, the new node $(s_a, \{i\})$ is created and is assigned with the state n. Otherwise, the node $(s_a, \{i\})$ is pruned. An example search tree using the database in Figure 1 with minSup=50%, no gap constraints and applied regex constraint <a+[b|c]2d> is shown in Figure 7.

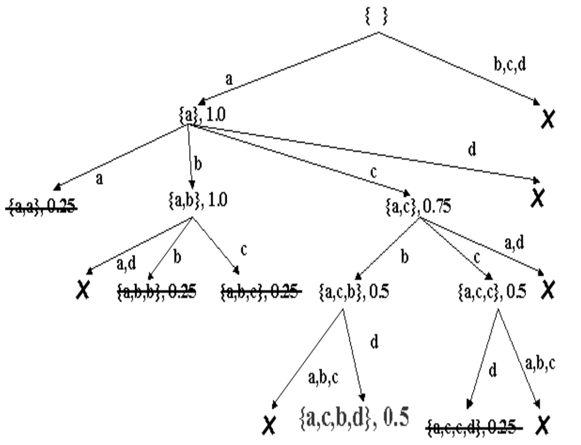


Figure 7: Running SPAM with no gap constraint and regex constraint of <a+[b|c]2d>

3.5 Overall Algorithm

With the addition of the gaps and regular expression constraints, a sequence can be pruned even if it has support greater than or equal to minSup if it does not satisfy any constraint. This raises a problem for S-step and I-step pruning in which the pruning techniques are based on only minSup. Our approach to preserve the S-step and I-step is to do support counting twice. The first round of support counting do not include any constraint and thus prune the search tree with only minSup. The information of this step is used for S-step and I-step pruning. The second round of support counting incoporates the contraints and prune all child nodes that contain sequence that does not satisfy the constraints. The pseudocode for the modified SPAM algorithm is as follows:

```
SPAM-Constraints(Database $D$, Regex $R$)
$st_{init} \leftarrow$initial state of $R$
$I \leftarrow$set of all items in $D$
$F \leftarrow \phi$
For each ($i \in I$)
    DFS-Constraints(($\{i\}$),$I$,$I$,$st_{init}$)
For each ($n$ in the tree)
    if(n is *valid* w.r.t $R$)
        $F \leftarrow F \cup n$
Return $F$
```

```
DFS-Constraints(node n=$(s_1, ..., s_k)$,$S_n$,$I_n$, $st_{curr}$)
$S_{temp} \leftarrow \phi$
$I_{temp} \leftarrow \phi$
For each ($i \in S_n$)
    if(($s_1, ..., s_k, \{i\}$) is frequent)
        $S_{temp} \leftarrow S_{temp} \cup \{i\}$
For each ($i \in S_{temp}$)
    if($st_{curr} \xrightarrow{i} st_{next}$ is *legal* AND ($s_1, ..., s_k, \{i\}$)
        with minGap, maxGap is frequent)
            DFS-Constraints(($s_1, ..., s_k, \{i\}$), $S_{temp}$,
            all elements in $S_{temp}$ greater than $i$,$st_{next}$ )
For each ($i \in I_n$)
    if(($s_1, ..., s_k \cup \{i\}$) is frequent)
        $I_{temp} \leftarrow I_{temp} \cup \{i\}$
For each ($i \in I_{temp}$)
    if($st_{curr} \xrightarrow{i} st_{next}$ is *legal*)
        DFS-Constraints(($s_1, ..., s_k \cup \{i\}$), $S_{temp}$,
        all elements in $I_{temp}$ greater than $i$,$st_{next}$)
```

4 Features Extraction using SPAM

One of the goals of this paper is to describe the way in which SPAM is used to extract features for accurate transmembrane helix prediction for IMPs. In our framework, the input is an annotated protein sequence database D that consists of a collection of protein sequences and their corresponding labels. For example, one instance of the database may be:

```
MDLLYMAAAVMMGLAAIGIGILGGKFLEGAIPLLRTQFFIKLL
0000011111111111111111000000000000000111111
```

The first line is the protein sequence. Each character represents an amino acid residue. Each amino acid residue is labeled 1 if it appears in the transmembrane helix region, otherwise it is labeled 0. Therefore, transmembrane regions are the set of contiguous subsequences of protein that are labeled 1. There are two transmembrane helix regions in the sequence shown above.

There are twenty amino acids in nature and each amino acid possesses a number of biochemical properties (Hydrophobic, Charged, Small, Aliphatic, Aromatic, Polar, Tiny, Positive and Negative). For example, amino acid residue valine (V) is Small, Aliphatic and Hydrophobic. Usually, an amino acid is associated with a number of these properties. Since the amino acid properties are the ultimate determining factor of the protein's structure and function, ability to extract sequential patterns of amino acid property is also important. Using a mix of amino acid properties and amino acid residues in the process of sequential pattern mining allow the process to be more generic since we are able to consider patterns like ({hydrophobic}{L}{I}{polar}{charge}{L}) which may be far more interesting than just amino acid sequences.

We have identified three phases in protein transmembrane helix features extraction:(1)pre-processing, (2)features extraction and (3)features verification. This framework is implemented as a software called Pex-SPAM. The overview of the Pex-SPAM system is summarized in Figure 8.

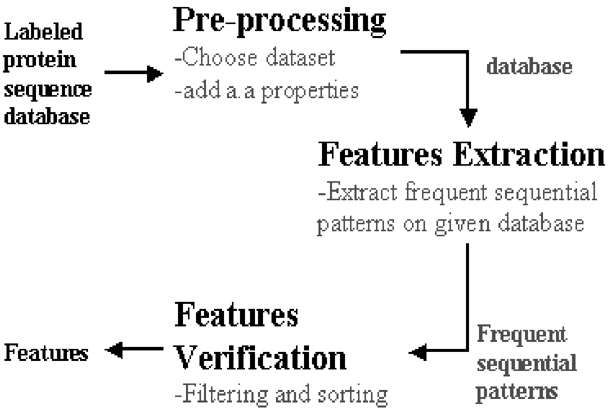


Figure 8: The three-phase architecture of Pex-SPAM

4.1 Pre-processing Phase

In this phase, the users have to select the regions that they are interested in and the pre-processor would select those regions and regard them as one input sequence for SPAM. If we are interested in looking at frequently occuring patterns within the transmembrane region (i.e. transmembrane feature), the pre-processor will traverse through all the protein sequences and make each transmembrane region a customer, with each amino acid converted to a transaction of one itemset. For example The transmembrane region FFIKLL will be converted to a customer with associated sequence ({F}{F}{I}{K}{L}{L}). A sequence database formed by considering a particular region r is denoted D_r. Its complementary database D_{non-r} contains all the regions other than r. The transaction database of the transmembrane regions is denoted as D_{trans} and its reciprocal database $D_{non-trans}$ contains all the sequences in the non-transmembrane regions.

Amino acid properties can be incorporated into the sequence database after choosing the type of feature to be extracted. Each itemset $i = \{aa\}$ is substituted by $\{aa, prop_1, prop_2, ..., prop_n\}$ where aa is an animo acid residue with n properties $prop_1$ to $prop_n$. For example, The itemset {V} is replaced with itemset {V, small, aliphatic, hydrophobic}. This enables the extraction of patterns that contain a combination of amino acid residues and their properties. If substitution is disabled, only sequences of amino acid resides are mined. If only the amino acid properties $\{prop_1, prop_2, ..., prop_n\}$ are substituted, only sequences of amino acid properties are mined. This give rise to an extra level of output selectivity.

4.2 Feature Extraction Phase

This is the core component of Pex-SPAM where the features are actually extracted. We use our modified version of SPAM. The details of this modified SPAM were described in section 3.

4.3 Post-processing Phase

Although the number of output features can be significantly reduced using appropriate constraints (e.g. minSup, minGap, maxGap and regex), the number of output features may still be quite large. Also, users may want to view the features in order of how promising the features are. The most promising features are likely to satisfy the following criteria: (1) Long sequence length, (2) high amino-acid/property ratio, (3) high support in region r and (4) low support in region *non-r*. Pex-SPAM allows the users to filter and sort the output features in order to identify the most promising features.

4.3.1 Filtering

The set of output sequences are filtered according to their length $|s|$ and support sup(s). User can define the minimum sequence length and minimum support that a feature should have. Also, regular expression constraints can also be applied in this step to further restrict the output space and increase the features' specificity. After sorting the features according to the score (see below), features can be filtered by their order. For example, a user can filter out most of the sequences except the top ten sequences.

4.3.2 Sorting

In order to measure how promising a potential feature is compared to other features, a score is assigned to each sequential pattern outputed from the features extraction phase. The score for any sequence s is calculated as follow:

$$score(s) = A \times |s| + B \times apr(s) + C \times sup(s) \quad (2)$$

All A, B and C are non-negative constant coefficients. The function apr(s) returns the ratio of amino acid over amino acid properties. For example, suppose we have a sequence s =({A,L},{I},{V},{hydrophobic,L,I}). There are three itemsets that contain only amino acid residue and one itemset that contain at least one property therefore apr(s)=3/1=3. This means sequence receive a higher score when they consist of a large number of animo acid residues and a small number of amino acid property since having the amino acid residue is more specific. If the denominator is zero, apr(s) is defined to be the number of itemsets. Therefore, long specific sequences with high support receive the highest scores. The sequences are then sorted in descending order of score. The sequences with the highest scores are candidates for promising features.

In reality we are not only interested in sequential patterns that appears very frequently in the regions r, but also want to ensure that these sequences do not appear often in regions *non-r* Therefore, the last step of features verification is to calculate the support of sequences in D_{non-r} for each pattern extracted with D_r using the same set of constraints and minSup=1%. Then, each of the patterns is assigned with a *Uniqueness* score which represent how unique this pattern is in relation to the other part of the protein regions. It is formulated as follows:

$$U(s) = \frac{Sup_{D_r}(s)}{Sup_{D_{non-r}}(s)} \quad (3)$$

For example, sequence s_a was extracted in the feature extraction phase and received a high score according to the scoring function (2). If its support in the transmembrane database $Sup_{D_{trans}}(s_a)$ is 0.4 and its support in the non-transmembrane database is $Sup_{D_{non-trans}}(s_a)$ is 0.02. Then, $U(s_a)$ is 0.4/0.02=20 which means the chance of finding pattern s_a in the transmembrane regions is 20 times higher than non-transmembrane regions and is thus a good candidate for a transmembrane feature. As a result, the sequences with high $U(s)$ are the most important features.

5 Experimental Evaluation

Pex-SPAM was implemented as a Java standalone application. We evaluated Pex-SPAM by running it on a real-world protein dataset. All experiments were performed on a 1.8GHz Intel Pentium 4 machine with 256MB main memory running Microsoft XP and J2SE Runtime Environment 1.5.

The dataset we used was taken from the same dataset as described in [4]. It consists of 124 sequences of significantly non-similar IMPs for which the transmembrane regions were experimentally confirmed. On average, there are four transmembrane helix regions per sequence and each region is twenty five amino acids long.

In the rest of this section, we present some experimental result on how runtime performance and output size is affected with different levels of constraints. Then we discuss how biologically relevant sequential features can be extracted by Pex-SPAM, particularly emphasizing the use of gap and regular expression constraints. Furthermore, a complete space complexity analysis of SPAM is presented to show that the memory requirements for SPAM are acceptable for real biological applications.

5.1 Runtime and Output Size

The runtime efficiency of SPAM has been extensively evaluated by [1] and was shown to outperform other similar algorithms, like SPADE [15] and PrefixSpan [12] on large dataset for long patterns. Since our main focus of this paper was the incorporation of gap and regular expression constraints into SPAM, the experiments we performed concentrate on the runtime performance gain and reduction of output size by running SPAM with different level of constraints.

Figure 9 shows the runtime performance of different level of maximum gap constraints with varying minimum support on a dataset that contains 269 transmembrane regions. From our biological knowledge, we know that transmembrane regions are on average 25-30 amino acid long and the longest sequential feature is about six amino acid long. Therefore a maximum gap of (30/6) = 5 is a reasonable limit for any biologically relavent sequential feature. Therefore, we only consider maxGap from 0 to 5 and compared it with the case where there is no gap constraints (maxGap=∞,the original SPAM algorithm). At high minimum support ($\geq$ 30%) there is not much difference in runtime performance. At low minimum support, a runtime benefit is clear by using stringent maximum gap constraints.

The second experiment looks at the effect of varying size of the input dataset with minimum support of 20% (Fig. 10). The result shows that runtime scales linearly with respect to the number of input transmembrane regions no matter how strong the maximum gap constraint is. However, a strong gap constraint can reduce the constant coefficient and hence improve the runtime performance for large input dataset.

Reducing the output space to gain features specificity was the main motivation to introduce gap constraints into SPAM. Experiments were performed to examine the relationship between the strength of the gap constraints and the resulting number of output patterns extracted. In this set of experiments, a dataset with 269 transmembrane regions was mined

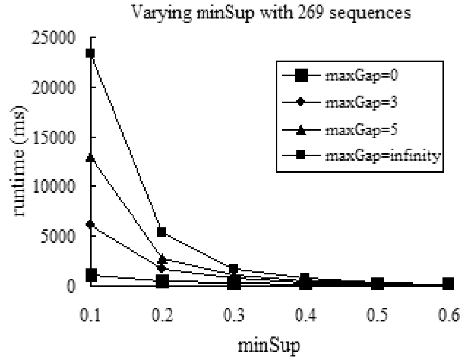


Figure 9: Runtime performance with varying minGap with 269 transmembrane regions

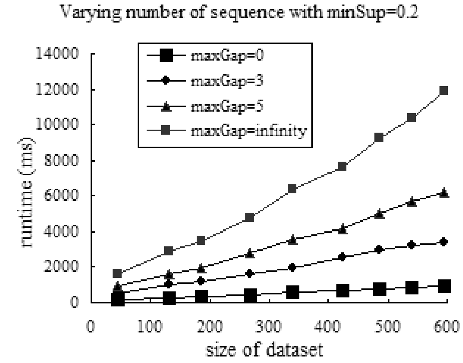


Figure 10: Runtime performance with varying number of input transmembrane regions

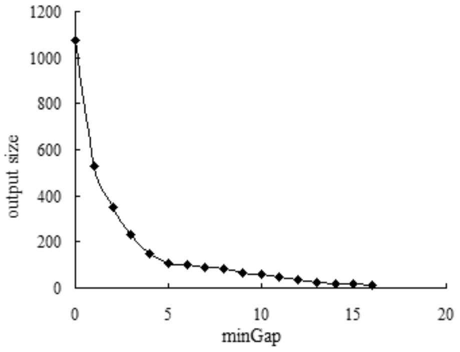


Figure 11: minGap vs output size

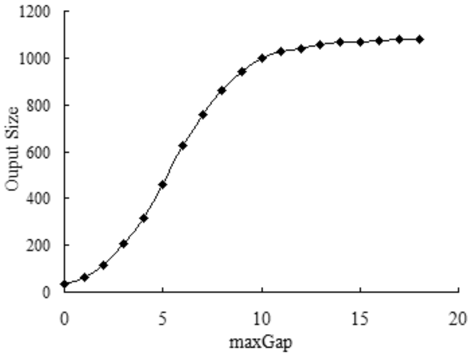


Figure 12: maxGap vs output size

with minimum support of 20% and varying number of minimum gap (Fig. 11) and maximum gap (Fig. 12).

Figure 11 clearly shows that the output size is significantly reduced by increasing minGap from 0 to 5. In Figure 12 the output size increases exponentially when maxGap is increased from 0 to 5. At maxGap=5 the curve has a stationary point beyond which the rate of growth decreases till the size of the output flattens out beyound maxGap=10.

Both experiments confirm that most transmembrane features, with minimum support of 20%, have a gap of five or below.. This is a very important finding in extracting transmembrane features as it justifies our motivation of incorporating the gap constraints into SPAM. Without the gap constraints, in particular the maxGap constraint, more than half of the output sequential features extracted by SPAM may not be biologically relevant.

5.2 Exploring biochemical properties of protein regions

Transmembrane and nontransmembrane regions are known to be dominated with hydrophobic and polar residues respectively[16]. Therefore one would expect to find long patterns of contiguous hydrophobic (polar) residues in such regions. We confirm this experimental finding with Pex-SPAM. In the first experiment, we ran Pex-SPAM on a dataset of 126 sequences with 526 transmembrane regions and 628 non-transmembrane regions. Using the regex constraint </hyd*> and minGap=maxGap=0 and minSup=1%, we compare the support level of different number contiguous hydrophobic residues on the transmembrane and the non-transmembrane regions. We only used amino acid property in this experiment as our interest is on the general pattern. The result is summarized in Figure 13. Essentially the uniqueness of the pattern grows exponentially as the number of contigous hydrophobic residues increases.

In general features that have a uniqueness score greater than five are likely to increase the accuracy of the transmembrane helix prediction. The best methods for transmembrane helix prediction can achieve a per-residue accuracy of 83% which means it can correctly predict an amino acid residue for being in the transmembrane region or in the non-transmembrane region 83% of the time [4]. In order to improve the accuracy of prediction, we need to use features that appear in the transmembrane region about 80% more often than in the non-transmembrane region.

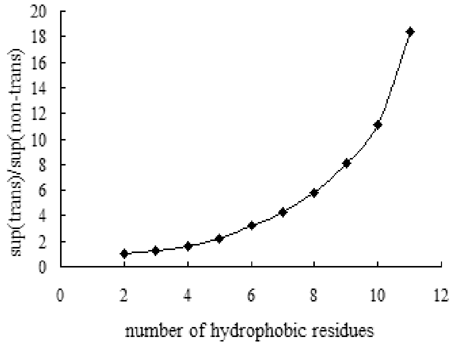


Figure 13: $\frac{sup_{trans}}{sup_{non-trans}}$ with varying number of contiguous hydrophobic residues. It is clear that long hydrophobic residue sequences dominate the transmembrane regions

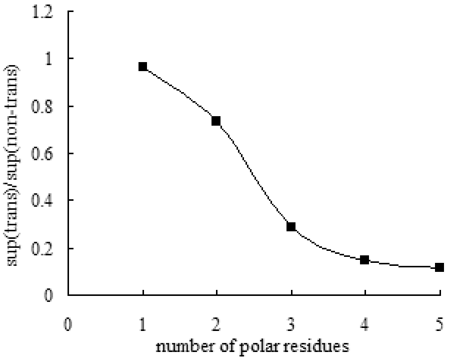


Figure 14: $\frac{sup_{trans}}{sup_{non-trans}}$ with varying number of contiguous polar residues

In order to achieve uniqueness score greater than 5, we need to have at least eight contiguous hydrophobic residues (see Fig. 13). A similar experiment was performed using regular expression constraint of </pol*> and the result is shown in Figure 14. We observe that the longest number of contiguous polar residues is half of that of the hydrophobic residues. Both computational experiments confirm that long contiguous hydrophobic residues and long contiguous polar residues are good features to distinguish transmembrane and nontransmembrane regions.

5.3 Extracting Features

Extending our previous result, we are now interested in finding some general transmembrane helix features based on amino acid properties. We now know that long contiguous hydrophobic protein segments are needed to obtain high uniqueness score but long patterns tend to have low support. We decided to extract contiguous amino acid property patterns of length 9, which were found as long enough together with satisfing high uniqueness score.. Pex-SPAM was executed with regex contraint </prp9>, minGap=maxGap=0 (i.e. any 9 contiguous amino acid properties) and minSup=1%. The list of features extracted with the highest uniqueness score are shown in Table 2.

feature	uniqueness
(hyd7 sml hyd)	6.72
(hyd5 sml hyd3)	6.63
(hyd8 sml)	6.61
(hyd6 sml hyd2)	6.58

Table 2: Example of some generic transmembrane features. hyd=hydrophobic, sml=small

This gives rise to an interesting finding that hyd*sml*hyd* is possibly an important pattern in the transmembrane regions. Although there are amino acid residues that are both hydrophobic and small, but the small amino acid in the patterns in Table 2 is probably not hydrophobic as the pattern (hyd9) has low uniqueness score. Therefore, another round of features mining was performed. This time we looked for specific amino acid patterns. We defined a regular expression </hyd*/sml*/hyd*> and tested it without using any amino acid property. The features with the best uniqueness score is listed in Table 3. Among those patterns, amino acid T and V are small and V is also hydrophobic.

feature	uniqueness
(IITL)	11.1
(ITVL)	5.7
(ITLV)	5.4
(LITL)	5.1

Table 3: Example of some specific transmembrane features

The pattern (IITL) is a good feature for predicting protein transmembrane helix region as the chance of finding this pattern in the transmembrane region is 11 times higher than in the non-transmembrane region. Therefore, using this feature should improve the prediction accuracy.

5.4 Space Complexity Analysis

The biggest disadvantage of SPAM is that it is quite space inefficient. The use of SPAM is thus a space-time tradeoff. Ayres *et al* [1] gave an analysis of the memory consumed by the input database of SPAM. Let D be the total number of customer, C be the average number of transactions per customer and N be the total number of items across all transactions. The amount of memory required to store the input database in the bitmap data structure is $(D \times C \times N)/8$ bytes. Extending this analysis, we discuss the total amount of memory needed for the entire data mining process. For each S-step and I-step, a new vertical bitmap for the extended sequence is created and stored. If the sequence is deemed infrequent, the node is pruned and the bitmap is destroyed to free up the memory. Therefore, the maximum amount of memory needed for the algorithm is the amount of memory needed to store the complete search tree after mining process. The total amount of memory occupied by the tree is $(D \times C \times P)/8$ bytes where P is the total number of output patterns. Thus, the estimated amount of memory (in byte) occupied by SPAM is:

$$memory \approx \frac{D \times C \times (N + P)}{8} \qquad (4)$$

For example, suppose we want to extract transmembrane features from a large protein database of 10000 sequences of IMPs with each containing 10 transmembrane regions. On average, there are 30 amino acid per transmembrane region. We know there are 20 amino acid and we consider 10 different amino acid properties, therefore 30 possible items are present. Furthermore, suppose we have 1000 output sequences. The estimated amount of memory taken up is $[100000 \times 30 \times (30 + 1000)]/8 = 386250000$ bytes which is aproximately 370 MB, which is very reasonable for a database nowadays. The main point is to show that the whole data mining process of SPAM can be be done within main memory on a modern machine which can avoid I/O delay. Therefore, althought SPAM is space insefficient, it is still suitable for protein transmembrane helix features extraction

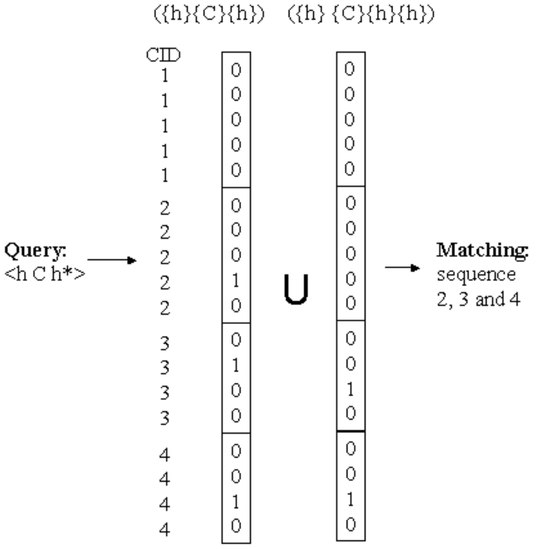
6 Pex-SPAM as Query Engine for Protein Secondary Structure

Another application of Pex-SPAM is its use as an underlying engine for protein secondary structure queries. Our sequential pattern mining approach is able to handle more complex queries that contain both primary and secondary structures. For example, on top of the class of queries as shown in [7], We are able to pose query like <h 3 8><C 5 7><? 0 *><G 3 4>. In the query, C stands for amino acid cystein and G for glycine. This section describes how this can be achieved.

We extend Pex-SPAM to handle secondary structure query in the following minner: In the pre-processing phase, each protein sequence is regarded as one customer and the secondary structure label of an amino acid is regarded as the property of that amino acid. Therefore, after the pre-processing phase, the input sequence database will contain sequences that looks like: ({G,h}{Q,l}{S,l}{D,l}{S,l}...).

In the features extraction phase, the input query is first converted to a regular expression constraint for SPAM. Gaps specified in the input query is converted to minGap and maxGap constraints. They are applied locally only at the location specified. For example, query <h 2 3><? 0 *><G 5 5><e 3 4> is converted to regular expression <[h2|h3][G5][e3|e4|e5]>. Globally, minGap=0 and maxGap=0, except the position after [h2|h3], which should have minGap=0 and maxGap=∞. After SPAM is run, each leaf node of the sequence tree contains a bitmap of a particular sequence that match the query. Since gap constraints, can be handled efficiently using the bitmap data structure, we believe that our approach can handle queries with gaps more efficiently than the previous approach.

In the third phase, instead of performing features verification, we perform a linear search on the bitmap of matching sequences. A sequence matches the query if it contains at least one 1 in its corresponding bitmap section. The complete set of output sequences is the union of all sequences retrived from each bitmap (Fig. 15). This union operation can be efficiently implemented using hash table.



CID	({h}{C}{h})	({h} {C}{h}{h})
1	0	0
1	0	0
1	0	0
1	0	0
1	0	0
2	0	0
2	0	0
2	0	0
2	1	0
2	0	0
3	0	0
3	1	0
3	0	1
3	0	0
4	0	0
4	0	0
4	1	1
4	0	0

Figure 15: Proposed framework for answering secondary structure query using Pex-SPAM

Althought this query system has not been implemented yet, the framework for using Pex-SPAM as a secondary structure query engine is established. The theoretical runtime of this proposed query engine is the runtime of SPAM plus $O(D \times C \times P)$ where D is the number of input protein sequences, C is the average number of amino acid per protein sequence and P is the number of matching patterns.

The theoretical amount of space taken up in this case can also be estimated by equation (4). Lets consider running a query on a real database like swiss-prot, which consists of about 200000 protein sequences and the average sequence length is 400 (http://au.expasy.org/sprot/relnotes/relstat.html). The number of possible item in this case is 23 (20 amino acid and 3 secondary structure labels). The expected number of output sequences in this case is very low as the query is used as the regular expression constraints for Pex-SPAM. Therefore, let's assume that we have maximum of 100 matching patterns. The total amount of memory used up in this query is approximately $[200000 \times 400 \times (23 + 100)]/8$ bytes which is 1.23 GB. This memory requirement is very reasonable for real database query engine. Since SPAM is time efficient and the memory trade off is acceptable in a real biological database the application of Pex-SPAM in secondary structure queries should be further studied.

7 Conclusion

Motivated by the need of mining protein transmembrane helix features for protein sequence classification and secondary structure queries, we modified a time-efficient sequential pattern mining algorithm, SPAM, to consider gaps and regular expression constraints.

A three phase framework of protein transmembrane helix features extraction was described and implemented into a software called Pex-SPAM. The effect of different level of contraints on runtime performance and output size is evaluated. Our experimental results show that introducing gap constraints in extracting transmembrane helix feature is essential for mining biologically relavent patterns. We also show that the use of combination of gap and regular expression constraints is important in the process of extracting high quality transmembrane helix features.

We also suggest a way to use Pex-SPAM as an underlying query engine for answering queries based on protein secondary structure. Our proposed approach would harness the efficient bitmap data structure, and would be able to consider both protein primary and secondary structure together.

Despite the space-inefficiency of SPAM, we have shown that the amount of memory taken up by both applications is modest. Given the excellent runtime of SPAM for large databases, Pex-SPAM system is ideal for our bioinformatics problem.

References

[1] Ayres, J., Flannick, J., Gehrke, J., Yiu, T.: Sequential pattern mining using a bitmap representation. In: KDD. (2002) 429–435

[2] Garofalakis, M.N., Rastogi, R., Shim, K.: Spirit: Sequential pattern mining with regular expression constraints. In Atkinson, M.P., Orlowska, M.E., Valduriez, P., Zdonik, S.B., Brodie, M.L., eds.: VLDB'99, Proceedings of 25th International Conference on Very Large Data Bases, September 7-10, 1999, Edinburgh, Scotland, UK, Morgan Kaufmann (1999) 223–234

[3] Leite, J., Amoscato, A., Cascio, M.: Coupled proteolytic and mass spectrometry studies indicate a novel topology for the glycine receptor. J Biol Chem **275** (2000) 13683–13689

[4] Lukov, L., Chawla, S., Church, W.B.: Conditional random fields for transmembrane proteins prediction. In: Proceedings of the 19th Pacific-Asia Conference on Knowledge Discovery and Data Mining (PAKDD). (2005) 151–161

[5] McCallum, A.: Efficiently inducing features of conditional random fields. In: Proceedings of 19th Conference on Uncertainty in Artificial Intelligence. (2003)

[6] Branden, C., Tooze, J.: Introduction to Protein Structure. Garland Publishing, USA (1999)

[7] Hammel, L., Patel, J.M.: Searching on the secondary structure of protein sequences. In: Proceedings of the 28th VLDB Conference, Hong Kong, China. (2002) 634–645

[8] Agrawal, R., Srikant, R.: Mining sequential patterns. In Yu, P.S., Chen, A.L.P., eds.: Proceedings of the Eleventh International Conference on Data Engineering, March 6-10, 1995, Taipei, Taiwan, IEEE Computer Society (1995) 3–14

[9] Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In Bocca, J.B., Jarke, M., Zaniolo, C., eds.: VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile, Morgan Kaufmann (1994) 487–499

[10] Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In Apers, P.M.G., Bouzeghoub, M., Gardarin, G., eds.: Advances in Database Technology - EDBT'96, 5th International Conference on Extending Database Technology, Avignon, France, March 25-29, 1996, Proceedings. Volume 1057 of Lecture Notes in Computer Science., Springer (1996) 3–17

[11] Pei, J., Han, J., Wang, W.: Mining sequential patterns with constraints in large databases. In: CIKM '02: Proceedings of the eleventh international conference on Information and knowledge management, New York, NY, USA, ACM Press (2002) 18–25

[12] Pei, J., Han, J., Mortazavi-Asl, B., Wang, J., Pinto, H., Chen, Q., Dayal, U., Hsu, M.C.: Mining sequential patterns by pattern-growth: The prefixspan approach. IEEE Transactions on Knowledge and Data Engineering **16** (2004) 1424–1440

[13] Ng, R.T., Lakshmanan, L.V.S., Han, J., Pang, A.: Exploratory mining and pruning optimizations of constrained association rules. In Haas, L.M., Tiwary, A., eds.: SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA, ACM Press (1998) 13–24

[14] Wang, K., Xu, Y., Yu, J.X.: Scalable sequential pattern mining for biological sequences. In: CIKM '04: Proceedings of the thirteenth ACM conference on Information and knowledge management, New York, NY, USA, ACM Press (2004) 178–187

[15] Zaki, M.J.: Spade: An efficient algorithm for mining frequent sequences. Machine Learning **42** (2001) 31–60

[16] Wallin, E., Tsukihara, T., Yoshikawa, S., von Heinjne, G., Elofsson, A.: Architecture of helix bundle membrane proteins: An analysis of cytochrome c oxidase from bovine mitochondria. Protein Science **6** (1997) 808–815