Mining Closed and Multi-Supports-Based Sequential Pattern in High-Dimensional Dataset

Meng Han[1,2], Zhihai Wang[1], and Jidong Yuan[1]
[1]School of Computer and Information Technology, Beijing Jiaotong University, China*
[2]School of Computer Science and Engineering, Beifang University of Nationalities, China

Abstract: *Previous mining algorithms on high dimensional datasets, such as biological dataset, create very large patterns sets as a result which includes small and discontinuous sequential patterns. These patterns do not bear any useful information for usage. Mining sequential patterns in such sequences need to consider different forms of patterns, such as contiguous patterns, local patterns which appear more than one time in a special sequence and so on. Mining closed pattern leads to a more compact result set but also a better efficiency. In this paper, a novel algorithm based on BI-directional extension and multi-supports is provided specifically for mining contiguous closed patterns in high dimensional dataset. Three kinds of contiguous closed sequential patterns are mined which are sequential patterns, local sequential patterns and total sequential patterns. Thorough performances on biological sequences have demonstrated that the proposed algorithm reduces memory consumption and generates compact patterns. A detailed analysis of the multi-supports-based results is provided in this paper.*

Keywords: *High-dimensional dataset, closed pattern, contiguous pattern, multi-supports, biological sequences.*

Received November 1, 2012; accepted April 29, 2013; *published online August 17, 2014*

1. Introduction

Sequential Pattern Mining (SPM) has become an essential data mining task with broad applications. Some previous studies contributed to the efficient mining of sequential patterns in high dimensional dataset. TD-Seq [11] is used for mining sequential pattern from high dimensional stock sequence database with a top-down transposition-based searching strategy. BVBUC [14] is an efficient colossal pattern mining in high dimensional datasets. SPM is becoming a promising strategy in mining biological sequences which is an important high dimensional dataset. It is used to mine structured motifs from protein or DNA sequences [1, 3, 16, 17]. Some methods are used to mine compressed patterns in biological sequences. Such as, MCSF is an efficient approach to mine maximal contiguous frequent patterns from large DNA sequence databases [9]. TOPPER is used to mine top-*k* patterns in biological sequences based on regularity measurement [18]. All these methods above are used to mine patterns to meet a support in high dimensional dataset. In recent years, some methods focus on mining some interesting patterns, such as BioPM [16] and WildSpan [8]. BioPM is based on prefix projected method to find the protein motifs in protein sequences. It discovers patterns to meet local support and distribution support [16]. WildSpan conducted experiments with two mining strategies, protein-based and family-based mining [8]. These algorithms mine complete patterns with two kinds of supports.

The complete set of patterns is huge for effective usage. We need a compact but high quality set of patterns, such as closed patterns and maximal patterns [6]. In order to mine efficient and interesting patterns on high dimensional dataset, a novel algorithm called Multi-supports-based and Contiguous Closed Pattern Mining (MCCPM) is proposed. It bases on multi-supports and discovers three kinds of contiguous closed sequential patterns.

The rest of this article is organized as follows. Section 2 reviews the closed SPM algorithm BIDE [15]. In section 3, some concepts are defined and the novel algorithm is proposed. Section 4 shows the experimental results on high dimensional dataset such as biological sequences and provides some interesting patterns. Finally, the conclusion is provided in section 5.

2. Problem Definition

In this section, we will discuss some patterns to meet different requirements. The first type is common sequential pattern which is frequent subsequence in a sequence database and the subsequence whose occurrence frequency is no less than minimum support threshold (called *min_sup*). The second one is contiguous sequential pattern. It is contiguous subsequence in each sequence. The third type is closed contiguous pattern which is compact but high quality. The last kind is multi-supports-based pattern. It finds

*Project supported by Science Foundation of State Nationalities Affairs Commission (No. 14BFZ008) and Scientific Research Funds for the Ningxia Universities (No. NGY2013094).

some interesting patterns which appear more than once in a sequence.

2.1. Problem Definition

Let S be a sequence database with a set of tuples <*sequence_id*, *s*>, where *sequence_id* is a sequence *id* and *s* is a sequence. If sequence α is contained in sequence β, α is a subsequence of β and β a supersequence of α. The support of a sequence α in S is the number of s containing α, denoted as *support*(α). Given a support threshold *min_sup*, a subsequence α is a frequent sequential pattern in S if *support*(a)≥ *min_sup***N*, where parameter N is the number of sequences in S.

For example, Table 1 shows the input biological sequence database S. Suppose the *min_sup*=50% (0.5), so a pattern occurrence frequency in the set of sequences is no less than 2 (4*0.5). The set of items in the database is {a, c, g, t} and the *sequence_id* are {s_1, s_2, s_3, s_4}. There are 7 items in sequence s_1, 14 items in sequence s_2, 11 items in sequence s_3 and 9 items in sequence s_4. Since, these 4 sequences contain sub-sequence x=(gag), x is a length-3 pattern and its support is 4, denoted as *support* (x)=4 (100%).

Table 1. Dataset S.

Sequence ID	Sequence
s_1	g a g g a g a
s_2	a g a t a t g c t t a g a g
s_3	a c t g a g g t a g a
s_4	a t t g a g c t t

2.2. Different Sequential Patterns

2.2.1. Contiguous Sequential Pattern

When minimum support=50%(0.5), 411 patterns are generated from dataset S. There are 4 length-1 patterns, 14 length-2 patterns, 47 length-3 patterns, 103 length-4 patterns, 131 length-5 patterns, 85 length-6 patterns, 24 length-7 patterns and 2 length-7 patterns. However, most of these patterns are discontinuous and useless.

For example, support of lengh-6 pattern (a g g a g a) is 3 and it appears in sequence s_1, s_2 and s_3. It is clear that its positions in sequence s_2 and s_3 are discontinuous as shown in Figure 1.

Sequence_id	Sequence
S_1	(g)(a)(g)(g)(a)(g)(a)
S_2	(a)(g)(a)(t)(a)(t)(g)(c)(t)(t)(a)(g)(a)(g)
S_3	(a)(c)(t)(g)(a)(g)(g)(t)(a)(g)(a)

Figure 1. Discontinuous pattern (aggaga).

In this paper, we focus on mining contiguous frequent patterns from high dimensional dataset, mainly on biological sequences. A sequence α=<$sequence_\alpha$_id, a_1, a_2, ..., a_n> is called a contiguous subsequence [9] of another sequence β=<$sequence_\beta$_id, b_1, b_2, ..., b_m> and β is a contiguous supersequence of α, if there exists integers $1 \le j_1 \le j_2 \le \ldots \le j_n \le m$ and $j_i = j_{i-1}+1$ for $1 \le i \le n-1$, while $a_1=b_{j1}$, $a_2=b_{j2}$, …, $a_n=b_{jn}$. Given a support threshold *min_sup*, a contiguous subsequence α is a contiguous sequential pattern in S if *support*(α)≥ *min_sup***N*.

The horizontal search tree [14] of generating contiguous sequential pattern in S is shown in Figure 2. There are 4 items a, c, g, t. Suppose *min_sup*= 50%(0.5), there are 24 contiguous patterns of dataset S. There are 4 length-1 patterns, 9 length-2 patterns, 7 length-3 patterns and 4 length-4 patterns.

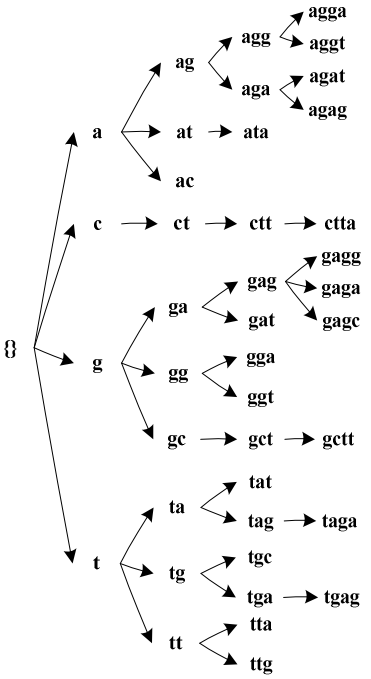


Figure 2. Horizontal search tree.

Figure 3 shows discontinuous and contiguous pattern (tgag). The first Figure 3-a shows the discontinuous pattern. Support of discontinuous pattern (tgag) is 3 and it appears in sequence s_2, s_3 and s_4. Figure 3-b shows the contiguous pattern (tgag). It is clear that the *support* (tgag)=2 and it appears in sequence s_3 and s_4. In this paper, we focus on contiguous patterns.

Sequence_id	Sequence
S_2	(a)(g)(a)(t)(a)(t)(g)(c)(t)(t)(a)(g)(a)(g)
S_3	(a)(c)(t)(g)(a)(g)(g)(t)(a)(g)(a)
S_4	(a)(t)(t)(g)(a)(g)(c)(t)(t)

a) Discontinuous pattern.

Sequence_id	Sequence
S_3	(a) (c) (t)(g)(a)(g) (g) (t) (a) (g) (a)
S_4	(a) (t) (t)(g)(a)(g) (c) (t) (t)

b) Contiguous pattern (tgag).

Figure 3. Discontinuous and contiguous pattern (tgag).

2.2.2. Closed Contiguous Sequential Pattern

The number of complete patterns is huge and we need a compact with high quality set of patterns. In general, compact pattern can be divided into two categories: lossless and lossy compression [6]. Closed patterns are lossless compression of frequent patterns. It contains the complete support information regarding to its corresponding frequent patterns.

When minimum support is 50%(0.5), the horizontal search tree of dataset *S* is shown in Figure 2. There are 4 items and 24 patterns. However, some of these patterns can be compressed, such as 24 patterns can be compressed into 9 closed patterns. Figure 4 shows the search tree of frequent subsequence (g). In this Figure, the value 'ga (4)' means that the support of frequent subsequence (ga) is 4. The subsequence with gray background means that it doesn't meet the minimum support and it needs to be pruned out. Closed sequential patterns are in black boxes. It is clear that there are 4 closed patterns of prefix (g). Dotted arrow points from *A* to *B* means that *A* is the supersequence of *B* and they have the same support. Therefore, *B* can be pruned. Such as *support*(gag)= *support*(ga) and *support*(ga)= *support*(g), then subsequences (ga) and (g) are deserted.

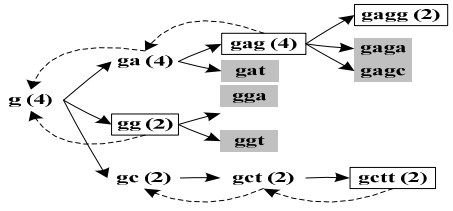


Figure 4. Search tree of prefix (g).

2.2.3. Multi-Supports-Based and Contiguous Closed Pattern

There are some interesting patterns in biological datasets. In this paper, we focus on three types of patterns:

1. Pattern appears in some sequences meeting the *min_sup*.
2. Pattern appears many times in a special sequence.
3. Pattern appears in some sequences but appears different times in each sequence. Positions and times of the last two kinds of patterns are recorded.

For example, pattern (gag) appears 5 times in 4 sequences, as shown in Figure 5. It appears 2 times in sequence s_1 and 1 time in other three sequences. Therefore, from this perspective, the relation of sequence s_2 to s_3 and s_4 is closer than them with sequence s_1.

Sequence	Sequence_id
S_1	(g)(a)(g)(g)(a)(g)(a)
S_2	(a)(g)(a)(t)(a)(t)(g)(c)(t)(t)(a)(g)(a)(g)
S_3	(a)(c)(t)(g)(a)(g)(g)(t)(a)(g)(a)
S_4	(a)(t)(t)(g)(a)(g)(c)(t)(t)

Figure 5. Patten (gag).

Meanwhile, we find that there are 9 patterns appear in sequence s_1. They are: Pattern (g) appears 4 times; (a) and (ga) appear 3 times; (ag) and (gag) appear 2 times; (gg), (aga), (agg) and (gagg) appear 1 time. Suppose the pattern which appears in sequence s_1 more than 2 times is needed, 3 local patterns: (g), (a) and (ga) are mined. Furthermore, there are only 3 closed patterns: (gag), (aga) and (gagg) in sequence s_1. If the local minimum support is 2, the local pattern (gag) is generated. There are 7 closed patterns in sequence s_2. There are 2 patterns: (at), (aga) appear 2 times and 5 patterns: (ct), (tg), (gag), (gctt), (taga) appear 1 time. We can also get 7 closed patterns in sequence s_3 and 6 closed patterns in sequence s_4.

The focus of this paper is on mining continuous and closed sequential patterns based on multi-supports in high dimensional dataset.

3. The BIDE Algorithm

BIDE [15] is a fast algorithm for mining discontinuous closed sequential patterns. It uses a sequence closure checking scheme to avoid the maintenance of closed candidate sequence. The proposed back scan pruning method can prune the search space aggressively [15].

- ***Definition* 1** [6]. *Closed Sequential Pattern*: A pattern *X* is a closed sequential pattern in a data set *S* if *X* is frequent in *S* and there exists no super pattern *Y* such that *Y* has the same support as *X* in *S*.

The BIDE algorithm [15] is shown as Algorithm 1. It contains two methods, one for length-1 frequent subsequence and one for other length frequent subsequence. BIDE algorithm first scans the database *S* once to find the length-1 frequent subsequences. Treat each subsequence α as a prefix, build a pseudoprojected database $S|\alpha$ and compute the number of forward-extension items which are the frequent items appear before prefix with the same support of prefix. If there is no forward-extension item, then calls subroutine *bide*(α, $S|\alpha$). If there is no backward-extension item and no super sequence of α, then output α as a closed pattern. This is the first method.

Algorithm 1: BIDE

```
Method 1:BIDE(S)
{
  scan the database S once to find the length-1 frequent
  subsequences α;
  for each α do
     bide(bide(α, S|α));
  end for
}
Method 2: bide(α, S|α))
{
  for each α do
     if there is no forward-extension item;
       then
         scan its projected database S|α, find each
         frequent item b, append b to α to form a
         subsequence α';
         bide(α', S|α'));
     end if
     if there is no backward-extension item;
       then
         output α as a frequent closed pattern;
     end if
  end for
}
```

Subroutine *bide*() is the second method. It recursively calls itself and works in some steps:

1. For each prefix α, scan $S|\alpha$ once to find each frequent item b. For each frequent item b, append it to α to form a new prefix α' and compute its forward-extension items number. If there is no forward-extension item, then calls subroutine *bide*(α', $S|\alpha'$).
2. If there is no backward-extension item and no super sequence of α', then output α' as a closed pattern.
3. Return the maximal *support*(α').

For example, when *min_sup* is 0.5, prefixes and the corresponding projected databases and patterns of database S are shown in Table 2. It is clear that, there are 24 complete sequential patterns. In Table 2, 24 complete patterns are compressed into 5 maximal sequential patterns or 9 closed sequential patterns. Therefore, 9 closed patterns are more compressed than 24 and they contain the complete information regarding to its corresponding frequent patterns. Though 5 maximal patterns are more compact, it does not contain the complete information.

Table 2. Prefixes and sequential patterns.

Prefix	Complete Atterns	Maximal Patterns	Closed Patterns
g	g, gc, ga, gg, gct, gag,gctt, gagg	gctt, gagg	gag, gctt, gagg
t	t, ta, tt, tg,tag, tga,taga, tgag	taga, tgag	tg, taga, tgag
c	c, ct, ctt,		ct
a	a, at, ag, aga, agg	at	at, aga

4. Mining Contiguous Closed Patterns

In this paper, two novel algorithms which are used to discover contiguous sequential patterns based on multi-supports are proposed. The first algorithm is PrefixSpan* which is an improvement of PrefixSpan [13] and is used to mine contiguous pattern based on multi-supports. The second one is used to mine contiguous closed sequential patterns based on multi-supports, called MCCPM. Some methods are similar in these two algorithms. Therefore, we just provide the information of MCCPM. At first, we propose some definitions.

- ***Definition* 2** [4]. *Support*: The support of a subsequence X in a dataset S is the number of tuples in the dataset containing X, denoted as *support*(X)= $|\{<sequence\_id, s>|(<sequence\_id, s>\in S) \wedge (X\subseteq s)\}|$.
- ***Definition* 3** [18]. *Local Support*: The local support of a subsequence X in a dataset S is the number of tuples in a specific sequence Y containing X, denoted as *local_support*(X, Y)=$|\{ <location\_id, Y> |(Y\in S)\wedge (X\subseteq Y)\}\}|$.
- ***Definition* 4**. *Total Support*: The total support of a subsequence X in a dataset S is the total number of tuples in S, denoted as *total_support*(X)=$\sum_Y$*local_support*(X, Y).
- ***Definition* 5**. *Local Sequential Pattern*: Local sequential pattern is a subsequence whose occurrence frequency in one specific sequence is no less than local minimum support (*local_min_sup* (*sequence_id*)).
- ***Definition* 6**. *Total Sequential Pattern*: Total sequential pattern is a subsequence whose occurrence frequency in dataset S is no less than total minimum support (*total_min_sup*).

There are three types of patterns in this paper. The first one is sequential pattern, which is a frequent subsequence whose occurrence frequency in the set of sequences is no less than *min_sup* [4]. The second one is local sequential pattern, which is a frequent subsequence whose occurrence frequency in one specific sequence is no less than *local_min_sup* (*sequence_id*). The third one is total sequential pattern, which is a frequent subsequence whose occurrence frequency in dataset S is no less *total_min_sup*.

For example, the database S is as shown in Table 1. Table 3 shows some information of closed contiguous patterns on S. The first column is closed sequential pattern, denoted as X, the second column is the *support*(X) and the third column is the *total_support*(X). The tuples <*sequence_id*, *location_ id*> in last column is the location where the frequent pattern appears. The variable *sequence_id* is the sequence *id* and *location_id* is the item *id* started with number 0 in that sequence. Suppose the *min_sup* is 0.75, we get 4 closed sequential patterns as shown in Table 3. The *support*(aga)=3 means that pattern (aga) appears in 3 sequences: s_1, s_2 and s_3. While *total_support*(aga)=4 means that (aga) appears 4 times in dataset S. It appears 1 time in *location_id*=4 of sequence s_1 (denoted as <*sequence_id*, *location_id*>=<s_1, {4}>). It appears 2 times in *location_id*={0, 10} of sequence s_2(<s_2, {0, 10}>). It appears 1 time in *location_id*=8 of sequence s_3(<s_3, {8}>). If *total_min_sup*=4, 2 total sequential patterns: (gag) and (aga) are discovered.

Table 3. Closed sequential patterns and supports.

Closed Pattern	Support	Total Support	Location <sequence_id, location_id>
tg	3	3	<s_4, {2}>, <s_3, {2}>, <s_2, {5}>
ct	3	3	<s_4, {6}>, <s_3, {1}>, <s_2, {7}>
gag	4	5	<s_4, {3}>, <s_3, {3}>, <s_2, {11}, <s_1, {0, 3}>
aga	3	4	<s_3, {8}>, <s_2, {0, 10}>, <s_1, {4}>

Table 4 shows the local frequent subsequences in sequence s_1. The first column is local sequential pattern and the second column is the local support of pattern. The tuples <*sequence_id*, *location_id*> in the last column has the same meaning as in Table 3. Subsequences(gag) appears 2 times in *location_id*={0, 3}. Therefore, its local support is 2, denoted as *local_support*(gag, s_1)=2 and (aga) appears 1 time in *location_id*={4}, denoted as *local_support*(aga, s_1)=1. If *local_min_sup* (s_1) is 1, two local sequential patterns in sequence s_1 are discovered.

Table 4. Local closed sequential patterns in sequence s_1.

Local Pattern	Support	Local Support	Location <*sequence_id, location_id*>
gag	4	2	<s_1, {0, 3}>
aga	3	1	<s_1, {4}>

From Tables 3 and 4 we can get three types of sequential patterns and their positions:

1. If *min_sup*=0.75, four closed patterns are generated: (tg), (ct), (gag) and (aga).
   <*pattern*, <*seuqnece_id*, *location_id*>> ={ <tg, {< s_4,{2}>, < s_3, {2}>, < s_2, {5}>}>, <ct, {<s_4,{6}>, < s_3, {1}>, <s_2, {7}>}> <gag, {<s_4, {3}>, <s_3, {3}>, <s_2, {11}, <s_1, {0, 3}>}>, <aga, {<s_3, {8}>, <s_2, {0, 10}>, <s_1, {4}>}>}.
2. If *min_sup*=0.75 and *local_min_sup*=2, two local patterns are generated: (gag) and (aga).
   <*local pattern*, *sequence_id*> = {< gag, s_1 >, < aga, s_2 >}.
3. If *min_sup*=0.75 and *total_min_sup*=4, two total patterns are generated: (gag) and (aga).
   <*total pattern*, <*seuqnece_id*, *location_id*>>={<gag,{<s_4, {3}>, <s_3, {3}>, <s_2, {11}, <s_1, {0, 3}>}>, <aga,{<s_3, {8}>, <s_2, {0, 10}>, <s_1, {4}>}>}.

Algorithm MCCPM is suitable to discover patterns on biological sequences and different interesting patterns based on multi-supports. MCCPM is shown as follows. The input parameters are sequence database and three kinds of minimum supports.

Algorithm 2: MCCPM

Input:

1. *Database S,*
2. *minimum support thresholds: support_thresholds= {min_sup,local_min_sup, total_ min_sup}.*

Output: Sets of three kinds of contiguous closed sequential patterns.

```
Method 1:   MCCPM(S)
{
  scan S, find length-1 frequent patterns α;
  for each α do
      scan S again, find the location information of α: <α,
      sequence_id, location_id> and store them into
      PrefixLocation|α.
      generate S|α = pseudo projected database of prefix α.
      if (!ForwardScan(α))
         then
            call bide(α, S|α, PrefixLocation|α, support_
            thresholds);
      end if
  end for
}
Method 2: bide(α, S|α, PrefixLocation|α, support_   thresholds)
{
  scan S|α once, find each frequent item b;
  for each b do
      append b to α to form a new prefix α';
      according to PrefixLocation|α, find the location
      information <α', sequence_id, location_ id> of α'
      and store into PrefixLocation|α';
      let S|α'=pskjeudo projected database of α';
      if ( !ForwardScan(α') )
          then
              call bide(α', S|α', PrefixLocation|α',
              support_thresholds);
      end if
      if(!ForwardScan(α')&&!BackwardExtension(α'))
          then
              if support(α')>min_sup
                 then
                     output α' as sequential pattern;
              end if
      if local_support(α', s)>local_min_sup(s)
          then
              output <α', s> as local sequential pattern;
      end if
      if total_support(α')>total_min_sup
          then
              output α' as total sequential pattern.
      end if
  end for
}
```

In algorithm MCCPM, the method *ForwardScan*() is used to check whether it exists items before prefix to meet the same support. Return true if we should stop to explore this prefix. Method *Backward Extension*() return true if there is a backward-extension [15] as it is used in BIDE.

We provide examples of discovering contiguous closed sequence patterns. Dataset *S* is as shown in Table 1. There are four items: (a), (c), (g), (t) in *S*. Suppose the *min_sup*=0.5, take item (g) as prefix to generate patterns. There are 4 closed sequential patterns of prefix(g) and it works in 5 steps.

- *Step* 1: Generate patterns (gagg) and (gag) as shown in Figure 6. Value 'gagg (2)' means frequent *subsequence*(gagg) and *support* (gagg)=2. Because of no more frequent subsequence of *prefix*(gagg) and *support*(gagg)≥ 2, it is a frequent pattern. *Support*(gaga) and *support*(gagc)) are less than minimum support, therefore they are pruned. After all frequent subsequence of *prefix*(gag) are generated, it gets the return value '2' as shown in Figure 6-b. It is clear that 2 is the maximal support of *support*(gagg), *support*(gaga) and *support*(gagc). Because *support*(gag)=4 is not equal to 2, (gag) is a frequent pattern.

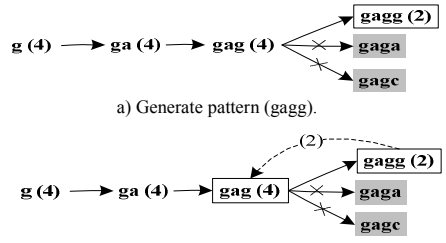
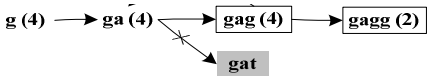


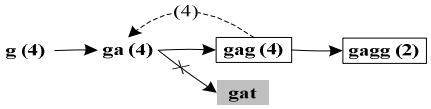
Figure 6. Generate patterns of prefix (gag).

- *Step* 2: Figure 7 shows the subsequences of prefix (gat). Because of no more frequent subsequence of *prefix*(gat) and its support is less than minimum support, it is pruned. After generating all frequent subsequence of *prefix*(ga), it get the return value '4'

as shown in Figure 7-b. Value 4 is the maximal support of *support*(gag) and *support*(gat). Because *support*(ga)=4 is equal to the return value 4, (ga) is not a frequent pattern.



a) Generate subsequence (gat).



b) Return To prefix (ga).

Figure 7. Generate subsequence of prefix (ga).

- *Step* 3: Generate sequential pattern (gg) as shown in Figure 8. The supports of subsequences (gag) and (ggt) are less than minimum support, therefore they are pruned. The *support*(gg)=2 is more than minimum support, it is a frequent pattern.

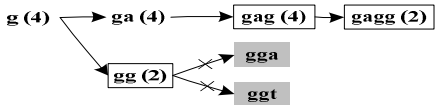
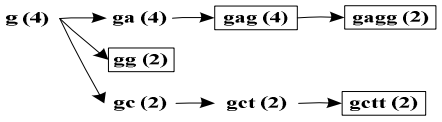
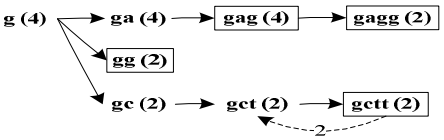


Figure 8. Generate pattern (gg).

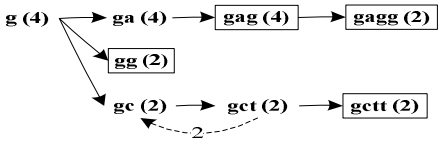
- *Step* 4: Generate sequential pattern (gatt) as shown in Figure 9. The support of frequent subsequence (gctt) is more than minimum support as shown in Figure 9-a, therefore it is a sequential pattern. After generating all frequent subsequence of prefix (gct), it gets the return value 2 as shown in Figure 9-b. The *support* (gct)=2 is equal to the return value 2, so (gct) is not a frequent pattern. The frequent subsequence (gc) is not a pattern too as shown in Figure 9-c.
- *Step* 5: Maximal support value 4 is returned to frequent subsequence (g) as shown in Figure 10. The value 4 is the maximal support value of *support*(ga), *support*(gg) and *support*(gc). Because the *support*(g)=4 is equal to value 4, it is not a frequent pattern.



a) Generate pattern (gctt).



b) Return to prefix (gct).



c) Return to prefix (gc).

Figure 9. Generate sunsquences of prefix (gc).

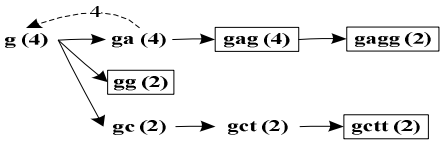


Figure 10. Return maximal support to subsequence (g).

From step 1 to step 5, four closed patterns are mined of prefix (g). Methods of discovering local closed patterns and total closed patterns are the same as discovering closed patterns.

5. Performance Evaluation

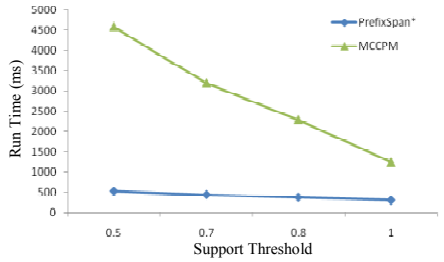
In this chapter, we provide experimental results to compare the performance of two algorithms. PrefixSpan* mines complete and contiguous patterns based on multi-supports. MCCPM mines closed and contiguous patterns based on multi-supports. All experiments were conducted on a 3.0 GHz AMD PC with 2.0 GB main memory, running Microsoft Windows 7.

In our performance study, we use 7 DNA sequences, which meet the condition of "Homo sapiens" and "cancer" from NCBI. There are four kinds of cancers as shown in Table 5, the first 4 lines are sequences about colon cancer, line 5 is about stomach cancer, line 6 is about colorectal cancer and the last line is about ovarian cancer. The sequence lengths are shown in the last column in Table 4 and the average length is 2384.

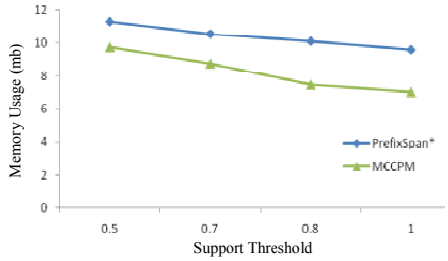
Table 5. DNA sequences.

ID	Sequence Title	#items
iu	Mutation in the DNA mismatch repair gene homologue hPMS2 is associated with hereditary nonpolyposis colon cancer	2697
s_1 (U03911)	The human mutator gene homolog MSH2 and its association with hereditary nonpolyposis colon cancer	3080
s_2 (U07418)	Mutation of a mutL homolog in hereditary colon cancer	2503
s_3 (U07343)	Mutation in the DNA mismatch repair gene homologue hMLH1 is associated with hereditary non-polyposis colon cancer	2484
s_4 (U27467)	A novel Bcl-2 related gene, Bfl-1, is overexpressed in stomach cancer and preferentially expressed in bone marrow	737
s_5 (U04045)	Mutations of a mutS homolog in hereditary nonpolyposis colorectal cancer	2947
s_6 (U34880)	A cDNA from the ovarian cancer critical region of deletion on chromosome 17p13.3	2234

Figure 11-a shows the processing time of the two algorithms: PrefixSpan* and MCCPM, at different support thresholds. The *min_sup* is from 0.5 to 1. It is clear that the runtime of MCCPM is higher than PrefixSpan*. The reason is that the time consumptions of backward and forward extension checks on every prefix. The memory usages of the two algorithms at different support thresholds are shown in Figure 11-b. It is clear that the memory usage of MCCPM is lower than PrefixSpan* because of the pruning strategy. Figure 12 shows the numbers of complete patterns and closed patterns. From this figure, we can get that closed sequential pattern compress the result of complete sequential patterns, about 43% patterns are be compressed.



a) The runtime of algorithms.



b) The memory usage of algorithms.

Figure 11. The performance of algorithms PrefixSpan* and MCCPM on biological dataset.

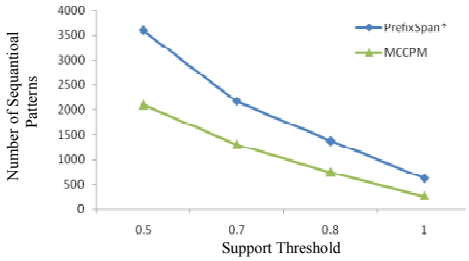


Figure 12. Numbers of frequent patterns on biological dataset.

When *min_sup*=100%, the comparison between complete and closed patterns is shown in Figure 13. There are 623 complete frequent patters. The number of closed frequent patterns is 267. The number of closed patterns is 42% of number of complete patterns. Therefore, more than half of complete patterns are deleted and the result set becomes more concise to use.

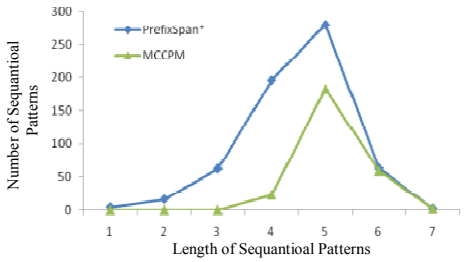


Figure 13. Length distribution of patterns on biological dataset.

What is the relationship between *min_sup*, *local_ min_sup* and *total_min_sup*? From the description of MCCPM algorithm we get that the sequential pattern should meet the *min_sup*. While local and total pattern should first meet the *min_sup* and then meet *local_min_sup* and *total_min_sup*. Let us discuss the relationship in three steps.

In the first step, we analyze the result set of complete contiguous patterns. When *min_sup* is 100% and 50%, the length distribution of patterns is shown in Figure 14. There are 623 patterns and the length of pattern is from 1 to 7 when *min_sup* is 100%. When *min_sup* is 50% there are 3593 patterns and the length of pattern is from 1 to 11.

Figure 15-a shows the numbers of total patterns when *min_sup* is 100% and 50%. The total minimum support threshold is from 50 to 90. In this figure, the first value '255' is the number of total frequent pattern when *min_sup*=100% (the minimum support number is 7) and *total_min_sup*=50. Specifically, 255 patterns appear more than 50 times in dataset *S* and appear in all sequences in *S*. The second value '271' is the number of total pattern when *min_sup*= 50% and *total_min_ sup*= 50. It means that 271 patterns appear more than 50 times in dataset *S* and appear in 4 different sequences in *S*.

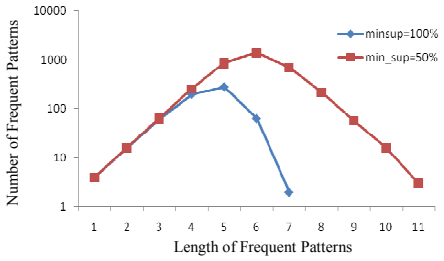
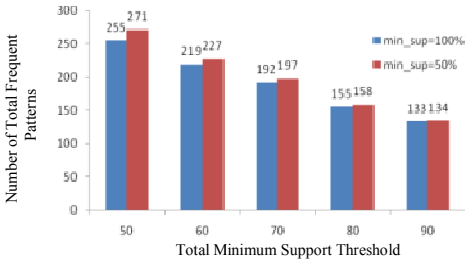
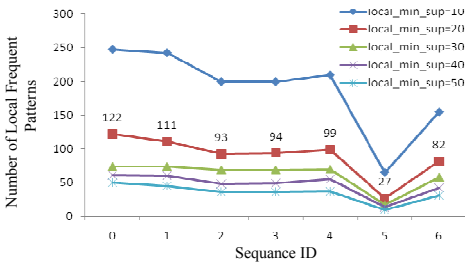


Figure 14. Length distribution of complete patterns.



a) The number of total patterns with different min_sup.



b) The number of local patterns in each sequence with different local_min_sup.

Figure 15. Number of total and local sequential patterns.

When *min_sup* is 100%, the numbers of local patterns are shown in Figure 15-b. The value of the abscissa is the sequence id. There are 5 broken lines in the figure which are the values of different local minimum supports. The *local_min_sup* is from 10 to 50. The value '122' on the second line means 122 patterns appear more than 20 times in sequence s_0 and appear in all sequences in *S*. The number of local pattern in sequence s_0 is shown in Figure 16. The first value '248' in this figure means 248 local patterns which appear more than 50 times in sequence s_0 when *min_sup*=100%. The second value '266' means 266 local patterns which appear more than 50 times in sequence s_0 and all these patterns appear in different 4 sequences in *S*.

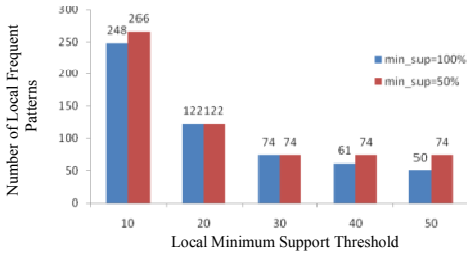


Figure 16. Number of local pattern of sequence s_0.

It is concluded that there is a little difference existed in numbers of total complete patterns with various *min_sup*. But, disparity in numbers of local complete patterns with various *min_sup* is increasing when *local_min_sup* is increasing.

In the second step, we analyze the result sets of closed contiguous patterns. When *min_sup* is 100% or 50%, the distribution of different length closed patterns is shown in Figure 17. There are 267 patterns and length of them is from 1 to 7 when *min_sup* is 100%. When *min_sup* is 50% there are 2103 patterns and length of that is from 1 to 11. It is clear that value 2103 (number of closed pattern) is much less than 3593 (number of complete pattern).

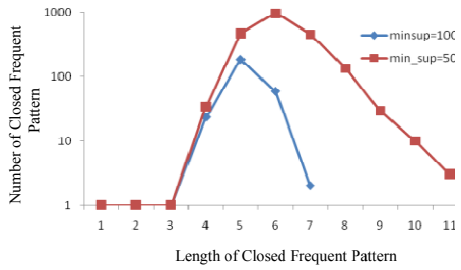
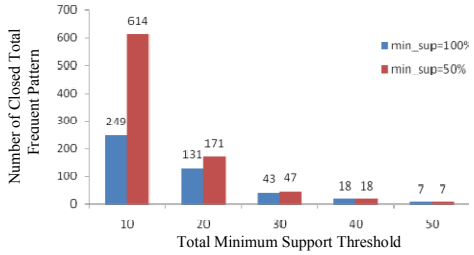


Figure 17. Number of different length closed sequential patterns.

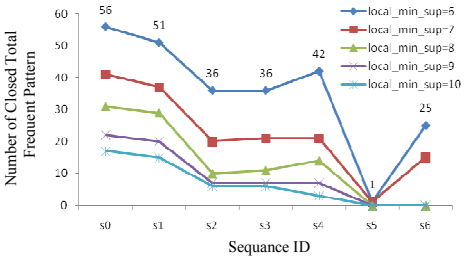
The numbers of total closed patterns are shown in Figure 18-a. Total minimum support threshold is from 10 to 50 which are lower than 50 to 90 (*total_min_sup* in mining complete patterns). The reason is that when *total_min_sup* number is 50, there are 255 complete patterns and only 7 closed patterns. Therefore, we choose the threshold from 10 to 50. In this figure, the first value '249' is the number of total closed pattern when *min_sup*=100% and *total_min_sup*=10. It means 249 patterns appear more than 10 times in dataset *S* and appear in all sequences in *S*. The second value '614' is the number of total patterns when *min_sup*=50% and *total_min_sup*=10. It is clear that, there is a large difference in the number of total pattern with two *min_sup* values. But, when *total_min_sup* number is 20 to 50, it is a little difference in these two *min_sup* values.

Figure 18-b shows the numbers of closed local pattern in each sequence when *min_sup*= 100%. The value of the abscissa is the sequence id. There are 5 broken lines in the figure which are the values of different local minimum supports. The *local_min_sup* is from 6 to 10 which are lower than 10 to 50 as shown in Figure 15-b. Because when *local_min_sup* is 10, there are 8 closed local patterns much lower than 198 complete local patterns on average of all sequences. In Figure 18-b, the last value '25' means that 25 closed local patterns appear more than 6 times in sequence s_6 and appear in all sequences in *S*.

The comparison between number of local pattern with *min_sup* 50% and with 100% in sequence s_0 is shown in Figure 19. The first value '56' means 56 closed local patterns appear more than 6 times in sequence s_0 and appear in 7 sequences in *S*. The second value '75' means 75 closed local patterns appear more than 6 times in sequence s_0 and all these patterns appear in different 4 sequences in *S*. From Figure 19, it is clear that the difference in two *min_sup* values is small when *local_min_sup* is higher than 8.



a) The number of total closed patterns with different min_sup.



b) The number of local closed patterns in each sequences with different local_min_sup.

Figure 18. Number of total and local closed sequential patterns.

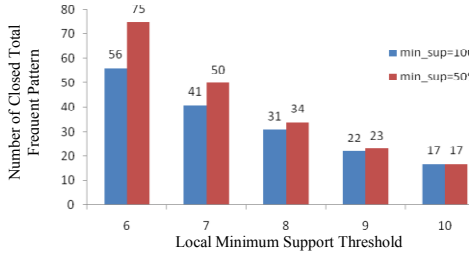
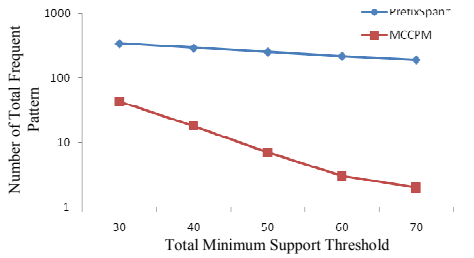
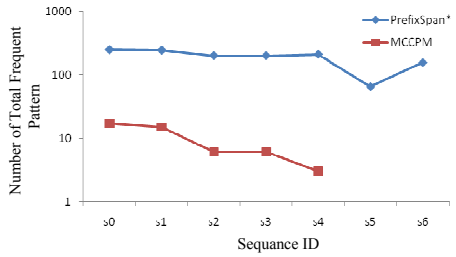


Figure 19. The closed local frequent pattern numbers of sequence s_0.

The last step compares the result sets of complete and closed patterns. Figure 20-a shows the numbers of total patterns and Figure 20-b shows the numbers of local pattern in different sequences. It is clear that the closed total pattern number and closed local number are much lower than complete pattern numbers.



a) The comparison of complete total patterns and closed total patterns.



b) The comparison of comlete and closed local patters in each sequence.

Figure 20. The comparison of complete and closed patterns in the same dataset.

From Figures 14 to 20, we concluded that:

1. Number of closed frequent patterns is lower than number of complete frequent patterns. About 43% complete patterns, 90% complete total patterns and 70% complete local patterns (with *local_min_sup*=10) are compressed on average.
2. Each total pattern and local pattern should meet two supports: *min_sup* and *total_min_sup* (or *local_min_sup*). They should first satisfy minimum support and then satisfy total or local minimum support.
3. Numbers of total or local patterns are decreasing with increasing *min_sup*. The difference in numbers of complete total patterns is little with various *min_sup*. But, the difference in numbers of complete local patterns with various *min_sup* is increasing with the growth of *local_min_sup*.
4. There are only 7 sequences in dataset *S*, but many patterns appear more than 100 times in *S* and appear more than 60 times in one sequence. Therefore, these patterns should be mined out and analyzed.

Now, we analyze the results about multi-supports sequential patterns. When *min_sup*=100% there are 623 complete contiguous patterns and 267 closed contiguous patterns. It is clear that closed patterns are much compressed than complete patterns.

Supposed *min_sup* number is 7 (100%) and *total_min_sup* is 50 seven total sequential patterns are mined as shown in Table 6. There are 7 patterns: 5 lentgh-4 patterns and 2 length-5 patterns. Pattern (tatt) appears 76 times in 7 DNA sequences. In details, it appears 19 times in sequence U14658, 15 times in sequence U03911, 14 times in sequence U07418, 14 times in sequence U07343, 6 times in sequence U27467, 7 times in sequence U04045 and 1 time in U34880. Because (tatt) appears 14 times in U07343 and U07418, the relationship between these two sequences may be closer than that with others.

From the results of MCCPM, we can also get the local patterns in one sequence. For example, when *min_sup*=100% and *local_min_sup* (U14658)=13, there are 8 local patterns in U 14658 as shown in Table 7. Pattern (tatt) appears 19 times and *location_id*={2036, 2651, 401, 2946, 2978, 911, 772, 1294, 1900, 2835, 1952, 1563, 354, 2866, 1751, 2936, 2876, 2846, 2570}.

Table 6. Patterns of total supports higher than 50.

Closed Pattern	Support	Total support
gcct	7	72
tact	7	52
tatt	7	76
ttat	7	55
aaaaa	7	54
aatc	7	62
agatg	7	50

Table 7. Part of local patterns in sequence U 14658.

Closed Pattern	Support	Local Support	Location < Sequence_Id, Location_Id>
tatt	7	19	<U14658, {2036, 2651, 401, 2946, 2978, 911, 772, 1294, 1900, 2835, 1952, 1563, 354, 2866, 1751, 2936, 2876, 2846, 2570}>
ttac	7	13	<U14658, {2002, 1128, 2950, 1565, 1203, 1321, 2797, 603, 1140, 2362, 2689, 1214, 1654}>
ttat	7	13	<U14658, {713, 1252, 3053, 308, 2444, 1951, 2863, 2650, 2977, 1897, 1293, 840, 2873}>
aaaaa	7	13	<U14658, {685, 744, 1975, 684, 1611, 2985, 683, 2645, 2644, 2807, 1267, 1591, 1695}>
aatc	7	16	<U14658, {821, 2810, 2777, 789, 1431, 2237, 2578, 381, 558, 2376, 1236, 259, 222, 2397, 2026, 3063}>
aaga	7	13	<U14658, {1135, 1936, 955, 2224, 2319, 1978, 2970, 2932, 1207, 1473, 397, 1105, 271}>
atga	7	13	<U14658, {1939, 1711, 1399, 2773, 2825, 1681, 2356, 249, 2245, 900, 759, 1455, 1273}>
agtt	7	13	<U14658, {305, 2475, 1304, 826, 2859, 865, 800, 2560, 3071, 1348, 1180, 391, 1555}>

6. Conclusions

Most of the previous work focuses on mining discontinuous closed sequential patterns. Some dataset, such as biological sequence analysis needs contiguous patterns. In order to mine contiguous and efficient sequential patterns, a novel algorithm, called Multi-supports-based and Contiguous Closed Pattern Mining (MCCPM) is proposed. It is based on multi-supports and used to mine contiguous closed pattern in high dimensional datasets.

There are three kinds of contiguous closed patterns: sequential patterns, local sequential patterns and total sequential patterns. These patterns correspond to three subsequence supports: *support*, *local support* and *total support*. The *support*(X) (X is a subsequence) is the number of sequences in the dataset containing X. If *support*(X)≥ *min_sup*, then X is a sequential pattern. The *local_support*(X, Y) (Y is a sequence) is the number of tuples in sequence Y containing X. If *local_support*(X, Y)≥*local_min_sup*(Y), then X is a local sequential pattern in sequence Y. The *total_support*(X) is the sum number of *local_ support* (X, Y). If *total_support*(X)≥ *total_min_sup*, then X is a total sequential pattern.

There are many interesting issues that need to be studied in the future, such as mining sequential patterns with constraints [4, 7, 12], mining closed gapped sub-sequences [5, 10] and so on.

References

[1] Alves R., Baena D., and Ruiz J., "Gene Association Analysis: A Survey of Frequent Pattern Mining from Gene Expression Data," *Briefings in Bioinformatics*, vol. 11, no. 2, pp. 210-224, 2009.

[2] Chen Y., Peng W., and Lee S., "CEMiner-an Efficient Algorithm for mining Closed Patterns from Time Interval-Based Data," *in Proceedings of the 11th IEEE International Conference on*

Data Mining, Vancouver, Canada, pp. 121-130, 2011.

[3] Exarchos T., Papaloukas C., Lampros C., and Fotiadis D., "Mining Sequential Patterns for Protein Fold Recognition," *the Journal of Biomedical Informatics*, vol. 41, no. 1, pp. 165-179, 2008.

[4] Ferreira P. and Azevedo P., "Protein Sequence Pattern Mining with Constraints," *in Proceedings of the 9th European Conference on Principles and Practice of Knowledge Discovery in Databases*, Porto, Portugal, pp. 96-107, 2005.

[5] Fournier-Viger P., Nkambou R., and Nguifo E., "A Knowledge Discovery Framework for Learning Task Models from User Interactions in intelligent Tutoring Systems," *in Proceedings of the 7th Mexican International Conference on Artificial Intelligence*, Zaragoza, Mexico, pp. 765-778, 2008.

[6] Han J., Cheng H., Xin D., and Yan X., "Frequent Pattern Mining: Current Status and Future Directions," *Data Mining and Knowledge Discovery*, vol. 15, no. 1, pp. 55-86, 2007.

[7] He D., Zhu X., and Wu X., "Mining Approximate Repeating Patterns from Sequence Data with Gap Constraints," *Computational Intelligence*, vol. 27, no. 3, pp. 336-362, 2011.

[8] Hsu C., Chen C., and Liu B., "WildSpan: Mining Structured Motifs from Protein Sequences," *Algorithms for Molecular Biology*, vol. 6, no. 1, pp. 1-16, 2011.

[9] Karim M., Rashid M., Jeong B., and Choi H., "An Efficient Approach to Mining Maximal Contiguous Frequent Patterns from large DNA Sequence Databases," *Genomics and Informatics*, vol. 10, no. 1, pp. 51-57, 2012.

[10] Lavanya B. and Murugan A., "A DNA Based Approach to Find Closed Repetitive Gapped Subsequences from A Sequence Database," *International Journal of Computer Applications*, vol. 29, no. 5, pp. 45-49, 2011.

[11] Liu H., Lin F., He J., and Cai Y., "New Approach for the Sequential Pattern Mining of High-Dimensional Sequence Databases," *Decision Support Systems*, vol. 50, no. 1, pp. 270-280, 2010.

[12] Mallick B., Garg D., and Grover P., "Constraint-Based Sequential Pattern Mining: A Pattern Growth Algorithm Incorporating Compactness, Length and monetary," *the International Arab Journal of Information Technology*, vol. 11, no. 1, pp. 1-11, 2014.

[13] Pei J., Jiawei H., Han J., Mortazavi-Asl B, Pinto H., Chen Q., Dayal U., and Hsu M., "Mining Sequential Patterns by Pattern-Growth: The prefixspan Approach," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1-17, 2004.

[14] Sohrabi M. and Barforoush A., "Efficient Colossal Pattern Mining in High Dimensional Datasets," available at: http://www.sciencedirect.com/science/article/pii/S0950705112000597, last visited 2012.

[15] Wang J., Han J., and Chun Li., "Frequent Closed Sequence Mining without Candidate Maintenance," *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, no. 8, pp. 1042-1056, 2007.

[16] Wang K., Xu Y., and Yu J., "Scalable Sequential Pattern Mining For Biological Sequences," *in Proceedings of the 13th ACM International Conference on Information and Knowledge Management*, New York, USA, pp. 178-187, 2004.

[17] Xiong Y. and Zhu Y., "BioPM: An Efficient Algorithm for Protein Motif Mining," *in Proceedings of the 1st International Conference on Bioinformatics and Bio-Medical Engineering*, Wuhan, China, pp. 394-397, 2007.

[18] Xiong Y., "TOPPER: An Algorithm for Mining Top K Patterns In Biological Sequences Based On Regularity Measurement," *in Proceedings of IEEE International Conference on Bioinformatics and Biomedicine Workshops*, HongKong, China, pp. 283-288, 2010.

Meng Han is received the MS degree in computer science from Beijing Jiaotong University, in 2008. She is now an Associate Professor at Beifang University of Nationalities in China and studies her PhD at Beijing JiaoTong Univeristy. Her research interests include data mining and machine learning.

Zhihai Wang is received the Doctor's degree in computer application from HeFei University of Technology in 1998. He is now a Professor in School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China. He has published dozens of papers in international conferences and journals. His research interest includes data mining and artificial intelligence.

Jidong Yuan is currently a candidate of PhD student in Beijing Jiaotong University (Beijing, China). His research interests include machine learning and data mining.