Filomat 35:2 (2018), 1703–1710
https://doi.org/10.2298/FIL1805703C

Published by Faculty of Sciences and Mathematics,
University of Niš, Serbia
Available at: http://www.pmf.ni.ac.rs/filomat

Top-k Sequence Pattern Mining with Non-overlapping Condition

Xin Chai[a,c], Dan Yang[a,c], Jingyu Liu[a,c], Yan Li[b], Youxi Wu[a,b,c]

[a]*School of Computer Science and Engineering, Hebei University of Technology, Tianjin 300401, China*
[b]*School of Economics and Management, Hebei University of Technology, Tianjin 300401, China*
[c]*Hebei Province Key Laboratory of Big Data Calculation, Tianjin 300401, China*

Abstract. Pattern mining has been widely applied in many fields. Users often mine a large number of patterns. However, most of these are difficult to apply in real applications. Top-k pattern mining, which involves finding the most frequent k patterns, is an effective strategy, because the more frequently a pattern occurs, the more likely they are to be important for users. However, top-k mining can only mine short patterns in mining applications with the Apriori property. It is well-known that short patterns contain less information than long patterns. In this paper, we focus on mining top-k sequence patterns of each pattern length. We propose an effective algorithm, named NOSTOPK (non-overlapping sequence pattern mining for top-k). The algorithm calculates the support of a pattern using a Nettree data structure, which has been introduced to tackle various types of pattern matching and sequence pattern mining issues. We find the top k patterns of length len, and calculate the supports of the corresponding $k \times |\Sigma|$ super-patterns of length $len + 1$ to discover the new top k super-patterns with $len + 1$. Experimental results demonstrate that the algorithm achieves a better performance than comparable algorithms.

1. Introduction

The core idea of data mining is to mine frequent patterns [1]. To effectively mine frequent patterns have been afforded much consideration by researchers. Sequence pattern mining [2,3] has played an important role in many mining tasks, such as author characteristics [4], prediction HAS QoE [5], sequential classification [6], time series analysis [7], and multivariate temporal data [8]. In order to avoid mining a lot of unrelated patterns, gap-constrained sequence pattern mining algorithms have been proposed. For example, Zhang et al. [9] studied the problem of gap pattern mining with no-condition. Min et al. [10] redefined the problem in [9], where the Apriori property can be used. Wang et al. [11] designed the algorithm MDSP-CGC to avoid improper settings of the gap constraint, where useful patterns cannot be found. However, sequence pattern mining with gap constraints is a tough problem, since not only many strategies are involved in such as pattern matching strategy to calculate the support of a pattern [12] and

2010 *Mathematics Subject Classification*. 68T10.
Keywords. Sequence pattern mining; Gap constraint; Top-k; Non-overlapping condition; Nettree.
Received: 10 October 2017; Accepted: 27 October 2017
Communicated by Hari M. Srivastava
Research supported in part by the National Natural Science Foundation of China under Grant 61673159, in part by the Natural Science Foundation of Hebei Province under Grant F2016202145, in part by the Natural Science Foundation of Heilongjiang Province under Grant F2017019, and in part by the Science and the Technology Project of Hebei Province under Grant 15210325. *Corresponding author: Yan Li*
Email addresses: chaixin@scse.hebut.edu.cn (Xin Chai), 1694990193@qq.com (Dan Yang), liujingyu@scse.hebut.edu.cn (Jingyu Liu), lywuc@163.com (Yan Li), wuc@scse.hebut.edu.cn (Youxi Wu)

Apriori [10] or Apriori-like [9] property to reduce the space of candidate patterns, but also various the state-of-the-art mining methods were proposed including no-condition [9], the one-off condition [13], and the non-overlapping condition [14]. Previous studies [15, 16] had shown that the non-overlapping condition is a constraint that lies between the one-off condition and no- condition, which makes the result more consistent with the avoidance of redundancy. For example, the algorithm NOSEP [16], which is designed to mine frequent patterns, can effectively balance the completeness of the mining with the Apriori property. As we know, it is difficult to set the minimum support threshold, *minsup*, without a priori knowledge. The higher the *minsup* is, the fewer mined patterns there tend to be. However, some useful patterns may be missed. On the contrary, when *minsup* decreases, more patterns will be frequent patterns. However, when using all these excessive patterns it is more difficult to draw real life conclusions. Therefore, an important issue is how to effectively compress frequent patterns and mine more valuable and meaningful patterns.

Top-k pattern mining can greatly reduce the number of frequent patterns, and effectively avoid the coverage of high frequency patterns when using the frequent closed pattern [17] algorithm. Therefore, this method is a more efficient technique for compressing frequent patterns [18]. Traditional top-k algorithms are designed to obtain the top-k of all frequent patterns. Sometimes users want to know the top-k patterns of length L, but traditional algorithms cannot satisfy such requirements. Wu et al. [19] proposed the algorithm MAPBOK, which is designed to obtain the top-k patterns with a support ratio larger than a support threshold ρ for each length. However, MAPBOK deals with no-condition. The latest research shows that sequence pattern mining with the non-overlapping condition can solve the shortcomings of the one-off condition and no-condition, and that it has good prospects [16]. However, the proposed NOSEP algorithm is used to mine all frequent patterns. Therefore, the algorithm NOSTOPK for top-k sequence pattern mining with the non-overlapping condition is proposed here. In this algorithm, the minimum support threshold ρ does not need to be specified. NOSTOPK is convenient, fast, and most importantly is able to satisfy the requirements of users.

The main contributions of this paper are threefold: (1) in algorithm NOSTOPK, the minimum support threshold ρ does not need to be specified, which effectively solves the problem that the minimum support threshold ρ is difficult to set for frequent pattern mining; (2) traditional top-k algorithms are to mine the top-k patterns of all frequent patterns. We deal the issue which can mine the top-k items in each length. Moreover, the required pattern length L and the number of patterns k are determined by users; (3) we propose an effective algorithm NOSTOPK and extensive experimental results verify the feasibility and effectiveness of our algorithm.

2. Problem Definition

Definition 1. *A collection of different events (namely characters) is called an event set, denoted by* Σ. *A sequence S of length n is an ordered list of events, denoted as* $S=s_1s_2\cdots s_n$, *where* $s_i \in \Sigma$. *A sequence database is a set of multiple sequences S, expressed as* $SDB=\{S_1,S_2,\cdots,S_N\}$.

Definition 2. *The number of characters that can separate two characters in a pattern is called a gap constraint, denoted as gap=[mingap, maxgap], where mingap and maxgap are nonnegative integers and represent the minimum and maximum gap constraints, respectively.*

Definition 3. *A pattern* $P=p_1[min_1, max_1]p_2\cdots[min_{m-1}, max_{m-1}]p_m$ *is called a pattern with gap constraints, where* $p_j \in \Sigma$ *and* Σ *is the set of events. In particular, given a gap constraint gap=[mingap, maxgap]* $(0 \le mingap < maxgap)$, *if* $min_1 = \cdots = min_{m-1} = mingap$ *and* $max_1 = \cdots = max_{m-1} = maxgap$, *then the pattern P is a pattern with periodic gap constraints.*

Example 1. *Suppose that the pattern* $P=p_1[min_1, max_1]p_2[min_2, max_2]p_3$*=C[0,2]G[0,2]C is given, where the gap constraint [0,2] indicates that between each character C, G, and C there can be zero, one, or two characters. Because* $min_1 = min_2 = 0$ *and* $max_1 = max_2 = 2$, *P is a pattern with periodic gap constraints.*

Definition 4. *Given the pattern P of length m and a sequence S of length n, if there are m integers* $l_1, l_2, \cdots, l_m$ *with* $1 \le l_1 < l_2 < \cdots < l_m \le n$, $min_j \le l_{j+1} - l_j - 1 \le max_j$, *and* $p_1 = s_{l_1}, p_2 = s_{l_2}, \cdots, p_m = s_{l_m}$, *then the location index* $\langle l_1, l_2, \cdots, l_m\rangle$ *represents an occurrence of P in S.*

Example 2. *Suppose the sequence S=$s_1s_2s_3s_4s_5$=ATATA and pattern P=A[0,2]T[0,2]A are given. The occurrences of P in S are $\langle 1,2,3\rangle$, $\langle 1,2,5\rangle$, $\langle 1,4,5\rangle$, and $\langle 3,4,5\rangle$, while the occurrences of ATA with gap=[0,1] in S are $\langle 1,2,3\rangle$ and $\langle 3,4,5\rangle$.*

Definition 5. *Given length constraints len=[minlen, maxlen], where minlen and maxlen are the minimum and the maximum length constraints, respectively, if L=$\langle l_1, l_2, \cdots, l_m\rangle$ satisfy $minlen \le l_m - l_1 + 1 \le maxlen$, then L is an occurrence with length constraints.*

Example 3. *The only occurrences with length constraints len=[1,4] of P=A[0,2]T[0,2]A in S are $\langle 1,2,3\rangle$ and $\langle 3,4,5\rangle$ in Example 2. For example, for the occurrence $\langle 1,2,3\rangle$, 3 - 1 + 1 = 3 satisfies len=[1,4].*

Definition 6. *For two occurrences L=$\langle l_1, l_2, \cdots, l_m\rangle$ and L′=$\langle l'_1, l'_2, \cdots, l'_m\rangle$, if $\forall 1 \le j \le m : l_j \ne l'_j$, then L and L′ are two non-overlapping occurrences. If all occurrences of an occurrence set are non-overlapping, then the occurrence set is a non-overlapping occurrence set. The support of P in S is the size of the maximum non-overlapping occurrence set, which is denoted by sup(P, S).*

Example 4. *In Example 2, under the condition of the length constraint len = [1,4], the largest non-overlapping occurrence set of P=A[0,2]T[0,2]A in sequence S is $\{\langle 1,2,3\rangle, \langle 3,4,5\rangle\}$, so sup(P, S)=2.*

Definition 7. *The support of the pattern P in the sequence database SDB is the sum of the supports of P in $S_1, S_2, \cdots, S_N$, respectively, denoted by $sup(P, SDB) = \sum_{i=1}^{n} sup(P, S_i)$.*

Definition 8. *For top-k sequence pattern mining with the non-overlapping condition, the goal of solving the problem is to mine the top-k patterns of each length with gap constraints and length constraints in a sequence S or sequence database SDB under the non-overlapping condition.*

3. NOSTOPK Algorithm

NOSTOPK, proposed in this paper, is a heuristic algorithm. Using a heuristic algorithm to solve the top-*k* pattern mining problem can significantly improve the efficiency, and provide results that have a small deviation. Because the result of the heuristic algorithm is only an approximate solution, NOSTOPK can use the Apriori property rather than an Apriori-like property to prune candidate patterns. If a sub-pattern *P* is a top-*k* pattern with length *L*, then there is a high probability that its super-pattern Q is a top-*k* pattern with length (*L* + 1). Thus, the basic idea of this algorithm is that if the pattern length is *L*, choose the top-*k* patterns to mine, then generate the corresponding $k \times |\Sigma|$ patterns with length (*L* + 1). From these patterns, select the top-*k* output patterns, and continue to mine (note that if $|\Sigma|$ is less than *k*, all characters are seen as frequent patterns with length 1). An illustrative example is provided as follows.

Example 5. *Suppose that we have a sequence S=$s_1s_2s_3s_4s_5s_6s_7s_8s_9s_{10}$ =GAATTCATCA, with length constraint len=[1,5], gap constraint gap=[0,1], L=2, and k=3. By scanning the sequence S, we can obtain the candidate C={A, C, G, T} with length 1, for which the supports are sup(A)=4, sup(T)=3, sup(C)=2, and sup(G)=1. Because k=3, {A, T, C}are put into the array TopkArr and output. Now, we generate the candidate patterns with length 2 using the three patterns with length 1. Thus, the candidate C with length 2 is {AA, AT, AC, TA, TT, TC, CA, CT, CC}. After calculating and sorting the supports, the results are sup(AT)=3, sup(TA)=2, sup(TC)=2, sup(CA)=2, sup(AA)=1, sup(AC)=1, sup(TT)=1, sup(CT)=1, and sup(CC)=0. Because k=3, {AT, TA, TC} are placed in the array TopkArr (note that there are three patterns whose sup(P, S) are 2, and any of these is correct). If L is greater than 2, we can iterate the above process.*

The NETGAP algorithm [16], which employs Nettree, a specially designed data structure, calculates the support effectively.

Definition 9. *Nettree is a tree-like data structure. It has root nodes, leaf nodes, children, parents, depth, and other concepts of the tree, but also has its own characteristics, such as that it can have multiple root nodes. In addition to the root node, other nodes can have multiple parent nodes. A node may have multiple paths to root nodes. It uses n_j^i to indicate the pattern in which layer j is labeled as i. Thus, the nodes of the same tag can appear in different levels [15, 16, 20].*

An illustrative example is presented to show the principles of NETGAP.

Example 6. *Given a sequence* $S=s_1s_2s_3s_4s_5s_6s_7s_8s_9s_{10}s_{11}s_{12}s_{13}s_{14}s_{15}$*=AATTCATCAGCCATG and a pattern* $P = p_1p_2p_3p_4$ *=A[0,3]T[0,3]C[0,3]G, the Nettree shown in Figure 1 (a) can be created according to the pattern and sequence.*

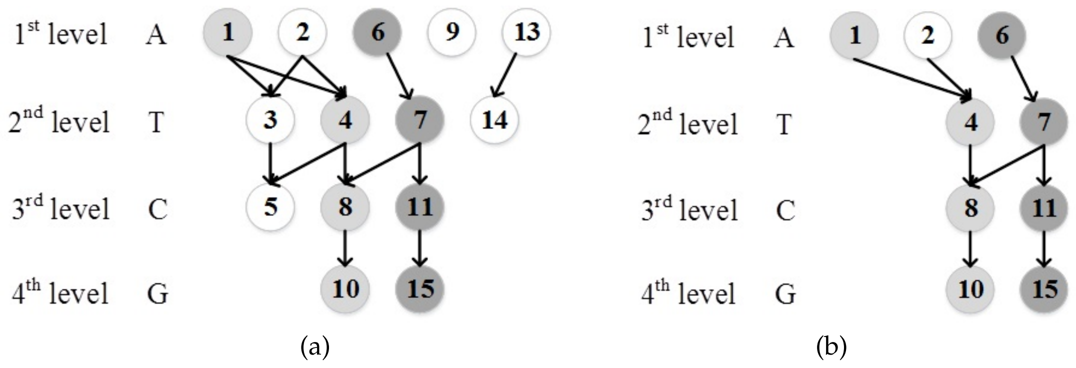


Figure 1: The Nettree for *P* in *S*

(1) In Figure 1 (a), n_3^5 *, which is a third level leaf, can be pruned according to NETGAP, because there is no path to reach a fourth level leaf via node* n_3^5*. Then,* n_2^3 *can also be deleted after pruning node* n_3^5*. Similarly, nodes* n_2^{14}*,* n_1^{13} *, and* n_1^9 *are pruned. The final Nettree is shown in Figure 1 (b).*

(2) It is easy to obtain the first occurrence $\langle 1,4,8,10\rangle$ *in the updated Nettree, marked in light grey in Figure 1 (b). Then, all four light grey nodes are pruned according to the non-overlapping condition. After pruning the four nodes,* n_1^2 *is also pruned, because there is no path to reach a fourth level leaf via node* n_1^2*. By iterating this process, the second occurrence* $\langle 6,7,11,15\rangle$*, marked in dark grey, can be found. Therefore, we know that there are two non-overlapping occurrences,* $\langle 1,4,8,10\rangle$ *and* $\langle 6,7,11,15\rangle$*, for P in S. Hence,* $sup(P,S)$*=2.*

The algorithm NOSTOPK is presented as follows.

Algorithm 1. *NOSTOPK*

```
Require: Sequence S or sequence database SDB, len = [minlen, maxlen], gap = [a, b], L, k
Ensure: The top-k patterns of each length in array TopkArr
1:  Scan sequence S or sequence database SDB, calculate the support of each character and sort them, get the top-k
    characters, store them in array TopkArr;
2:  level ← 1;
3:  while level ≤ L − 1 do
4:      C = gen_candidate ( level − 1); // generate candidate set and store the result in C
5:      for each cand in C do
6:          pro.value ← NETGAP(cand);
7:          if (pro.value <> 0) then
8:              queue.push(pro);
9:          end if
10:         while !queue.empty() && i ≤ k do
11:             Store the top-k patterns in array TopArr;
12:         end while
13:     end for
14:     level++;
15: end while
16: return TopkArr;
```

Table 1: Protein sequence datasets

Sequence Database	Number of Sequences	Length
SDB1	507	91875
SDB2	338	62985
SDB3	169	32503
SDB4	590	109424
SDB5	400	73425
SDB6	200	37327

4. Experimental Results and Analysis

As a benchmark dataset, the data used in this paper has been investigated in previous studies, such as [16] and [19], and consists of six protein sequences. The first three protein sequences SDB1, SDB2, and SDB3 can be obtained from ASTRAL95_1_161, and the last three SDB4, SDB5, and SDB6 can be obtained from ASTRAL95_1_171. The features of the protein sequences are shown in Table 1. All experiments are run on a computer with the Intel Core i3-2350M 2.30GHz CPU, 6 GB RAM, and Windows 7 operating system. Dev-C++ is used to develop all algorithms.

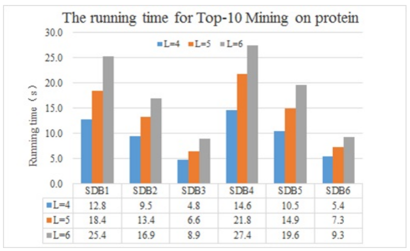
For the top-k mining problem with the non-overlapping condition, we use the ratio calculation formula with a weight to measure the precision of the top-k mining results. The equation is given in the text as Eq. (1).

$$Precision = (\sum_{i=1}^{d}(i \times a_i))/(\sum_{i=1}^{d}(i \times b_i)) \quad (1)$$

where a_i and b_i are the numbers of correct top-k frequent patterns and total top-k frequent patterns with length i, respectively, and d is equal to the length L. In general, b_i is equal to k, but when c is less than k, b_i becomes c, where c is the number of length i frequent patterns. According to Definition 8, suppose we want to find top k patterns with length L that means we will find $L \times k$ patterns and NOSTOPK mines the same amount of patterns. Therefore, the number of false positives is the same as that of false negatives. Hence, the precision and the recall are the same.

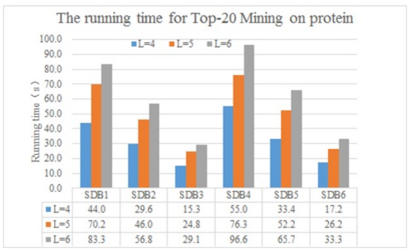
4.1. Running Time Evaluation

In order to illustrate how L, k, and the length of a sequence affect the running time of NOSTOPK, the results of protein sequences mined with different values for L, k, and the lengths of sequences are shown in Figure 2. (Note: L is the layer number and the maximum length of generated patterns.)



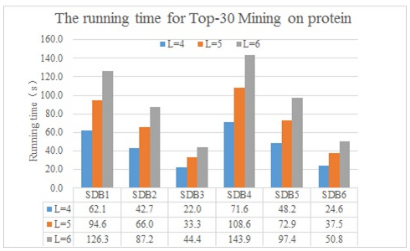
	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
L=4	12.8	9.5	4.8	14.6	10.5	5.4
L=5	18.4	13.4	6.6	21.8	14.9	7.3
L=6	25.4	16.9	8.9	27.4	19.6	9.3

(a) For top-10



	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
L=4	44.0	29.6	15.3	55.0	33.4	17.2
L=5	70.2	46.0	24.8	76.3	52.2	26.2
L=6	83.3	56.8	29.1	96.6	65.7	33.3

(b) For top-20



	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
L=4	62.1	42.7	22.0	71.6	48.2	24.6
L=5	94.6	66.0	33.3	108.6	72.9	37.5
L=6	126.3	87.2	44.4	143.9	97.4	50.8

(c) For top-30

Figure 2: The running time on protein sequences

As we can see from Figure 2, the longer the sequence is, the greater L and k are, and the longer the running time is.

(1) When L and k are the same, the longer the length of the sequence is, the longer the running time is.

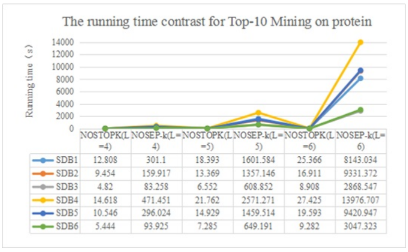
(2) When the lengths of sequences and k remain the same, the larger the L is, the greater the number of patterns at each layer is, and the number of candidate patterns increases. Thus, the running time increases. For example, for the top-30 patterns on each layer mined on SDB1, shown in Figure 2 (c), the running time is 62.1s when $L = 4$, while the running time is 126.3s when $L = 6$, which obtains more than two layers of the patterns, but the running time is more than twice as long.

(3) Clearly k also has a significant effect on the running time. The running time will be longer as k increases, because the number of candidate patterns also increases. We can clearly see that SDB4 is the longest protein sequence. When mining the top-10 on it, the running time for L=6 in Figure 2 (a) is 27.4s, the running time for the top-20 mining under the same condition in Figure 2 (b) is 96.6s, and the running time for the top-30 under the same condition in Figure 2 (c) is 143.9s.

4.2. Running Time Contrast Evaluation

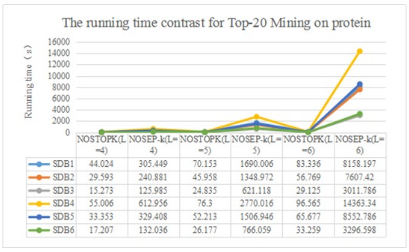
The algorithm NOSTOPK is compared with NOSEP-k (NOSEP for top-k), based on the algorithm NOSEP [16], to verify its efficiency. The algorithm NOSEP-k mines the top-k patterns in all frequent patterns with support greater than or equal to ρ at each length. The main difference between NOSTOPK and NOSEP-k is that the candidate set is generated differently. NOSEP-k generates the candidate set by the pattern growth, generating candidate patterns using frequent patterns with support greater than or equal to ρ. However, NOSTOPK generates a candidate set using the top-k patterns, which improves efficiency considerably.

A comparison of the running times of the two algorithms for the top-10, top-20, and top-30 on the protein sequences is given in Figure 3.



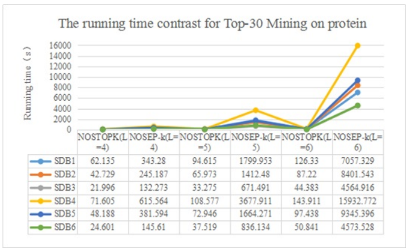
	NOSTOPK(L=4)	NOSEP-k(L=4)	NOSTOPK(L=5)	NOSEP-k(L=5)	NOSTOPK(L=6)	NOSEP-k(L=6)
SDB1	12.808	301.1	18.393	1601.584	25.366	8143.034
SDB2	9.454	159.917	13.369	1357.146	16.911	9331.372
SDB3	4.82	83.258	6.552	608.852	8.908	2868.547
SDB4	14.618	471.451	21.762	2571.271	27.425	13976.707
SDB5	10.546	296.024	14.929	1459.514	19.593	9420.947
SDB6	5.444	93.925	7.285	649.191	9.282	3047.323

(a) For top-10



	NOSTOPK(L=4)	NOSEP-k(L=4)	NOSTOPK(L=5)	NOSEP-k(L=5)	NOSTOPK(L=6)	NOSEP-k(L=6)
SDB1	44.024	305.449	70.153	1690.006	83.336	8158.197
SDB2	29.593	240.881	45.958	1348.972	56.769	7607.42
SDB3	15.273	125.985	24.835	621.118	29.125	3011.786
SDB4	55.006	612.956	76.3	2770.016	96.565	14363.34
SDB5	33.353	329.408	52.213	1506.946	65.677	8552.786
SDB6	17.207	132.036	26.177	766.059	33.259	3296.598

(b) For top-20



	NOSTOPK(L=4)	NOSEP-k(L=4)	NOSTOPK(L=5)	NOSEP-k(L=5)	NOSTOPK(L=6)	NOSEP-k(L=6)
SDB1	62.135	343.28	94.615	1799.953	126.33	7057.329
SDB2	42.729	245.187	65.973	1412.48	87.22	8401.543
SDB3	21.996	132.273	33.275	671.491	44.383	4564.916
SDB4	71.605	615.564	108.577	3677.911	143.911	15932.772
SDB5	48.188	381.594	72.946	1664.271	97.438	9345.396
SDB6	24.601	145.61	37.519	836.134	50.841	4573.528

(c) For top-30

Figure 3: Comparison of the running time under different L on protein sequences

(1) As the sequence length and k increase, the running times of NOSTOPK and NOSEP-k steadily increase with a little fluctuation.

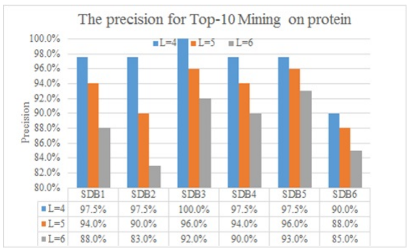
(2) As L increases, the running time of NOSEP-k increases more than that of NOSTOPK, which remains short with little fluctuation. When $L = 4$, the running time of NOSEP-k is over hundred seconds than that of NOSTOPK. The gap increases when $L = 5$, where the running time of NOSEP-k can be over 3000s greater than that of NOSTOPK. The longest running time for NOSTOPK was for the top-30 mining on SDB4 in Figure 3 (c), requiring about 144s, while NOSEP-k required almost 16,000s, which is 100 times greater than the running time of NOSTOPK. The difference in running time between the two algorithms is greatest when $L = 6$. The shortest time for NOSEP-k is between 2000s and 3000s, and the longest is between 15000s and 16000s, which is more than four hours, while NOSTOPK is completed within 144s.

Thus, the NOSTOPK algorithm has a high efficiency.

4.3. Precision evaluation

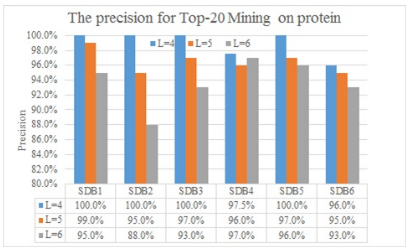
In order to illustrate how L, k, and the length of a sequence affect the precision of NOSTOPK concisely, the results of protein sequences mined with different L and k values and lengths of sequences are shown in Figure 4.

(1) We can see from Figure 4 that the precision is higher when k is larger, while the precision is lower when L is larger.



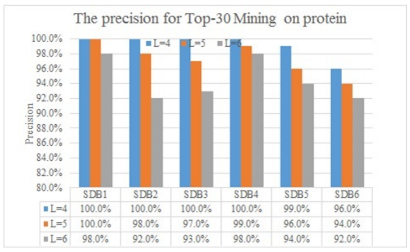
	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
🟦 L=4	97.5%	97.5%	100.0%	97.5%	97.5%	90.0%
🟧 L=5	94.0%	90.0%	96.0%	94.0%	96.0%	88.0%
⬜ L=6	88.0%	83.0%	92.0%	90.0%	93.0%	85.0%

(a) For top-10



	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
🟦 L=4	100.0%	100.0%	100.0%	97.5%	100.0%	96.0%
🟧 L=5	99.0%	95.0%	97.0%	96.0%	97.0%	95.0%
⬜ L=6	95.0%	88.0%	93.0%	97.0%	96.0%	93.0%

(b) For top-20



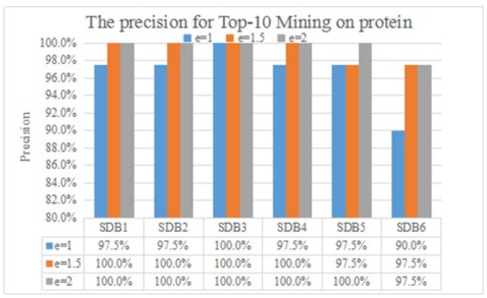
	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
🟦 L=4	100.0%	100.0%	100.0%	100.0%	99.0%	96.0%
🟧 L=5	100.0%	98.0%	97.0%	99.0%	96.0%	94.0%
⬜ L=6	98.0%	92.0%	93.0%	98.0%	94.0%	92.0%

(c) For top-30

Figure 4: Comparison of the precision on protein sequences

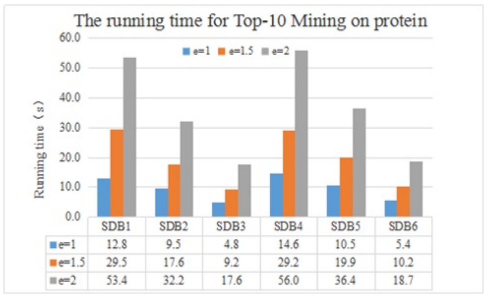
(2) Regardless of the lengths of sequences and how large k is, the precision is almost the lowest when $L = 6$ in Figure 4. This is because when $L = 6$, there are significantly more candidate patterns than when $L = 5$. As protein sequences contain more characters, while NOSTOPK generates patterns of length 6, only the top-k patterns of length 5 are used. Thus, there will be significantly fewer generating patterns, and there is a certain deviation from the correct patterns. We use the average precision to reflect this effect. Taking the mining from SDB1 to SDB6, for example, when $L = 4$ the average precision of NOSTOPK is 98%, when $L = 5$ the average precision is 95.6%, and when $L = 6$ the average precision is 92%. Since when $L = 4$, we consider the instances for Top-10, Top-20, and Top-30 on Datasets from SDB1 to SDB6, there are 18 instances and 1380 patterns in total will be discovered according to Definition 8. Similarly, we know that we will discover 1740 patterns and 2100 patterns in total for $L = 5$ and $L = 6$, respectively. The experimental results show that 18, 64, and 140 patterns are lost in NOSTOPK when $L = 4$, $L = 5$, and $L = 6$, respectively. Therefore, the smaller L is, the higher the precision is. However, as k increases, the overall precision also increases. For example, when $k = 10$ the average precision of NOSTOPK is 93%, when $k = 20$ it is 96%, and when $k = 30$ it reaches 97%. We know that we will find 900 patterns, 1800 patterns, and 2520 patterns, while the lost patterns are 70, 71, and 81 for $k = 10$, $k = 20$, and $k = 30$, respectively. Thus, the larger k is, the higher the precision is. Therefore, the effect of k on the precision is very important. In summary, the algorithm NOSTOPK achieves a higher precision within a short period of time.

To improve the precision, an effective method is enlarging the searching space. We mine top $e \times k$ patterns with length j and output top k patterns in them at first. Then we use $e \times k$ patterns to generate the candidate patterns with length $j + 1$ and find top $e \times k$ patterns. We iterate this process to tackle this issue. We can see that NOSTOPK is the special case with e=1. To report how e affects the precision and the running time, the results with L=4 and Top 10 are shown in Figure 5.



	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
🟦 e=1	97.5%	97.5%	100.0%	97.5%	97.5%	90.0%
🟧 e=1.5	100.0%	100.0%	100.0%	100.0%	97.5%	97.5%
⬜ e=2	100.0%	100.0%	100.0%	100.0%	100.0%	97.5%

(a) The precision for top-10



	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
🟦 e=1	12.8	9.5	4.8	14.6	10.5	5.4
🟧 e=1.5	29.5	17.6	9.2	29.2	19.9	10.2
⬜ e=2	53.4	32.2	17.6	56.0	36.4	18.7

(b) The running time for top-10

Figure 5: The precision and running time for Top-10 and L=4 mining on protein sequences

From Figure 5, the large the e is, the higher the precision is, but the longer the running time is. The reason lies that the larger the e is, the more the number of candidate patterns will be and the higher the precision of the results is. For example, when e=1.5, the precision on SDB1 to SDB4 all are 100%, but the running time is about twice greater than that when e=1.

5. Conclusion

In this paper, we analyze the shortcomings of existing algorithms, and propose a top-k sequence pattern mining algorithm with the non-overlapping condition, named NOSTOPK. This algorithm does not produce a large number of superfluous candidate patterns, and will not lose the patterns that the user is interested in. Furthermore, this algorithm is more effective in satisfying the needs of users than traditional top-k algorithms. The efficiency of NOSTOPK is verified by a large number of experiments on protein sequences.

References

[1] O. K. Alkan, P. Karagoz , CRoM and HuspExt: Improving efficiency of high utility sequential pattern extraction , IEEE Transactions on Knowledge and Data Engineering 27 (2015) 2645–2657.
[2] C. Li, Q. Yang, J. Wang, M. Li, Efficient mining of gap constrained subsequences and its various applications, ACM Transactions on Knowledge Discovery from Data 6 (2012) 2.
[3] L. Zhang, P. Luo, L. Tang, et al. Occupancy-based frequent pattern mining, ACM Transactions on Knowledge Discovery from Data 10 (2015) 14.
[4] H. Jiang, J. X. Zhang, H. J. Ma, et al. Mining authorship characteristics in bug repositories, Science China Information Sciences 60 (2017) 12107.
[5] F. Wang, Z. S. Fei, J. Wang, et al. HAS QoE prediction based on dynamic video features with data mining in LTE network, Science China Information Sciences 60 (2017) 042404.
[6] E. Egho, D. Gay, M. Boull, et al. A user parameter-free approach for mining robust sequential classification rules, Knowledge and Information Systems 52 (2017) 53–81.
[7] C. Tan, F. Min, M. Wang, et al. Discovering patterns with weak-wildcard gaps, IEEE Access 4 (2016) 4922–4932.
[8] I. Batal, G. F. Cooper, D. Fradkin, et al. An efficient pattern mining approach for event detection in multivariate temporal data, Knowledge and information systems 46 (2016) 115–150.
[9] M. Zhang, B. Kao, D. W. Cheung, et al. Mining periodic patterns with gap requirement from sequences, ACM Transactions on Knowledge Discovery from Data 1 (2007) 7.
[10] F. Min, Y. Wu, X. Wu, The Apriori property of sequence pattern mining with wildcard gaps, International Journal of Functional Informatics and Personalised Medicine 4 (2012) 15–31.
[11] H. F. Wang, L. Duan, J. Zuo, et al. Efficient mining of distinguishing sequential patterns without a predefined gap constraint, Chinese Journal of Computers 39 (2016) 1979–1991.
[12] Y. Wu, S. Fu, H. Jiang, et al. Strict approximate pattern matching with general gaps, Applied Intelligence 42 (2015) 566–580.
[13] H. Lam, F. Morchen, D. Fradkin, et al. Mining compressing sequential patterns, Statistical Analysis and Data Mining 7 (2013) 34–52.
[14] B. Ding, D. Lo, J. Han, et al. Efficient mining of closed repetitive gapped subsequences from a sequence database, In: Proceedings of IEEE International Conference on Data Engineering (2009) 1024–1035.
[15] Y. Wu, C. Shen, H. Jiang, et al. Strict pattern matching under non-overlapping condition, Science China Information Sciences 60 (2017) 012101.
[16] Y. Wu, Y. Tong, X. Zhu, et al. NOSEP: Non-overlapping sequence pattern mining with gap constraints, IEEE Transactions on Cybernetics 42017 DOI: 10.1109/TCYB.2017.2750691
[17] H. Yang, L. Duan, B. Hu, et al. Mining top-k distinguishing sequential patterns with gap constraint, Journal of Software 26 (2015) 2994–3009.
[18] Y. M. Chai, Z. Zhang, L. M. Wang, An algorithm for mining global closed frequent itemsets based on distributed frequent concept direct product, Chinese Journal of Computers 35 (2012) 990–1001.
[19] Y. Wu, L. Wang, J. Ren, et al. Mining sequential patterns with periodic wildcard gaps, Applied Intelligence 41 (2014) 99–116.
[20] Y. Wu, Z. Tang, H. Jiang, et al. Approximate pattern matching with gap constraints, Journal of Information Science 42 (2016) 639–658.