Applied Intelligence (2022) 52:9861–9884
https://doi.org/10.1007/s10489-021-02912-3

NetNMSP: Nonoverlapping maximal sequential pattern mining

Yan Li[1] · Shuai Zhang[2] · Lei Guo[3] · Jing Liu[2] · Youxi Wu[2,4] · Xindong Wu[5,6]

Accepted: 7 October 2021 / Published online: 10 January 2022

Abstract
Nonoverlapping sequential pattern mining, as a kind of repetitive sequential pattern mining with gap constraints, can find more valuable patterns. Traditional algorithms focused on finding all frequent patterns and found lots of redundant short patterns. However, it not only reduces the mining efficiency, but also increases the difficulty in obtaining the demand information. To reduce the frequent patterns and retain its expression ability, this paper focuses on the Nonoverlapping Maximal Sequential Pattern (NMSP) mining which refers to finding frequent patterns whose super-patterns are infrequent. In this paper, we propose an effective mining algorithm, Nettree for NMSP mining (NetNMSP), which has three key steps: calculating the support, generating the candidate patterns, and determining NMSPs. To efficiently calculate the support, NetNMSP employs the backtracking strategy to obtain a nonoverlapping occurrence from the leftmost leaf to its root with the leftmost parent node method in a Nettree. To reduce the candidate patterns, NetNMSP generates candidate patterns by the pattern join strategy. Furthermore, to determine NMSPs, NetNMSP adopts the screening method. Experiments on biological sequence datasets verify that not only does NetNMSP outperform the state-of-the-arts algorithms, but also NMSP mining has better compression performance than closed pattern mining. On sales datasets, we validate that our algorithm guarantees the best scalability on large scale datasets. Moreover, we mine NMSPs and frequent patterns in SARS-CoV-1, SARS-CoV-2 and MERS-CoV. The results show that the three viruses are similar in the short patterns but different in the long patterns. More importantly, NMSP mining is easier to find the differences between the virus sequences.

Keywords Sequential pattern mining · Maximal pattern mining · Nonoverlapping pattern mining · Gap constraint · Backtracking strategy · COVID-19 · MERS-CoV

1 Introduction

Sequential pattern mining [1, 2], as an important part of data mining [3, 4] and knowledge discovery [5], aims to mine frequent subsequences whose support is no lower than the given threshold. Many kinds of sequential pattern mining methods were proposed, such as outlying pattern mining [6], maximal sequential pattern mining [7–9], three-way sequential pattern mining [10–13], negative sequential pattern mining [14, 15], periodic pattern mining [16, 17], co-location pattern mining [18], contrast subspace mining [19–22], closed sequential pattern mining [23], utility pattern mining [24–29], and repetitive sequential pattern mining [30, 31]. Traditional sequential pattern mining neglects the number of occurrences of a pattern in a sequence, while repetitive sequential pattern mining does not [32]. Thus, repetitive sequential pattern mining can find more patterns. However, some of them are meaningless patterns. To solve this issue, based on repetitive sequential pattern mining, sequential pattern mining with gap constraints was proposed to find the frequent subsequences (known as patterns) which do not have to be continuous. The pattern can be represented as $P = p_1[min_1, max_1]p_2[min_2, max_2] \dots p_{m-1}[min_{m-1}, max_{m-1}]p_m$, where min_i and max_i, called a group of gap constraints, are the minimal and maximal wildcards between p_i and p_{i+1}, respectively [33, 34]. In sequential pattern mining, gap constraints are often the same , such as $P = p_1[a, b]p_2[a, b] \dots p_{m-1}[a, b]p_m$ or $P = p_1p_2 \dots p_{m-1}p_m$ with $gap = [a, b]$. This mining method is also called periodic sequential pattern mining [35] and has been applied in many applications, such as time series analysis [36, 37] and feature selection [38, 39].

✉ Youxi Wu
wuc567@163.com

Extended author information available on the last page of the article.

Fig. 1 All occurrences of pattern P in sequence S_1

	1	2	3	4	5		
$S =$	A	C	A	C	A		
	A	C	A	.	.	<1, 2, 3>	The first occurrence
	A	C	.	.	A	<1, 2, 5>	The second occurrence.
	A	.	.	C	A	<1, 4, 5>	The third occurrence
	.	.	A	C	A	<3, 4, 5>	The fourth occurrence

Nonoverlapping sequential pattern mining (or sequential pattern mining under the nonoverlapping condition) [40, 41] is a kind of sequential pattern mining with gap constraints, which means that each item (s_i) in a sequence can be matched by an item (p_j) at most once [42]. Recent studies showed that the nonoverlapping sequential pattern mining is a completeness mining method with the Apriori property [30]. Importantly, compared with other the state-of-the-art methods, it is easier to find valuable frequent patterns, and solves the problem of under-expression and over-expression of patterns [30]. Example 1 illustrates the nonoverlapping support which is a key issue in nonoverlapping sequential pattern mining.

Example 1 Suppose we have a sequence $S_1 = s_1s_2s_3s_4s_5$ = ACACA and a pattern $P = p_1[0,2]p_2[0,2]p_3$ = A[0,2]C[0,2]A, (or $P = p_1p_2p_3$ = ACA with gap = [0,2]). All occurrences are shown in Fig. 1.

In this example, pattern P is a pattern with gap constraints, which has four occurrences in sequence S_1. It is easy to know that subsequence $s_1s_2s_3$ is an occurrence of pattern P which can be written as <1, 2, 3>. Similarly, the other three occurrences are <1, 2, 5>, <1, 4, 5>, and <3, 4, 5>. Among them, <1, 2, 3> and <1, 2, 5> are two overlapping occurrences, since s_1 is matched by p_1 twice in the two occurrences, while <1, 2, 3> and <3, 4, 5> are two nonoverlapping occurrences, since s_3 is matched by p_3 and p_1 in these two occurrences, respectively. Therefore, the nonoverlapping support of pattern P in sequence S_1 is 2.

Traditional nonoverlapping sequential pattern mining algorithms mainly focused on discovering all frequent patterns [30]. However, one of the disadvantages is that the mining pattern set is large and contains too many redundant short patterns. To handle the problem of reducing redundant patterns, nonoverlapping closed sequential pattern mining was proposed [23], which means that there is no super-pattern with the same support. This method can effectively compress frequent patterns in a sequence with large gap, but it is less effective in a multiple-sequence dataset or in a sequence with small gap. Nonoverlapping Maximal Sequential Pattern (NMSP) mining is another method to reduce redundant patterns, which means that all super-patterns of the maximal patterns are infrequent. Example 2 shows that maximal pattern mining has better compression performance than closed pattern mining.

Example 2 Given sequence database $SDB = \{S_1$ = ACACA, S_2 = CACAC}, gap constraint gap = [0,2], and support threshold $minsup$ = 2.

Let us consider S_1 at first. The supports of patterns ”AC”, ”CA”, and ”ACA” in sequence S_1 are all 2. Therefore, all the three patterns are frequent patterns and pattern ”ACA” is a closed pattern. Thus, patterns ”AC” and ”CA” are compressed. Pattern ”A” is a closed pattern, since the supports of all its super-patterns ”AA”, ”AC”, and ”CA” are all 2, which are less than the support of pattern ”A”. Therefore, pattern ”A” cannot be compressed in closed pattern mining. However, pattern ”A” can be compressed in NMSP mining, since its super-patterns are frequent patterns.

Table 1 Comparison of mining results

Pattern type	Pattern set	Count
Frequent patterns in S_1	{A,C,AA,AC,CA,ACA}	6
Closed patterns in S_1	{A,AA,ACA}	3
NMSPs in S_1	{AA,ACA }	2
Frequent patterns in SDB	{A,C,AA,AC,CA,CC,AAC,ACA,ACC,CAA,CAC,CCA,ACAC,CACA}	14
Closed patterns in SDB	{A,C,AA,AC,CA,CC,AAC,ACA,ACC,CAA,CAC,CCA,ACAC,CACA}	14
NMSPs in SDB	{AAC,ACC,CAA,CCA,ACAC,CACA }	6

Hence, from Table 1, NMSP mining has better compression performance in S_1.

Moreover, if we mine nonoverlapping closed sequential patterns in SDB, we notice that there is no frequent pattern whose support is the same as that of its super-patterns. Thus, all the frequent patterns are closed patterns, which means that closed pattern mining cannot compress frequent patterns in SDB. However, there are only 6 maximal patterns in SDB in Table 1. Hence, NMSP mining also has better compression performance in SDB. The comparison between frequent patterns and NMSPs is shown in Fig. 2.

In Fig. 2, we can see that NMSP mining effectively compresses the frequent patterns, since all subpatterns of NMSPs are frequent patterns. Hence, NMSPs can be seen as the boundary of frequent patterns, since all their subpatterns are frequent and their superpatterns are infrequent. Moreover, NMSPs provide boundary information for frequent and infrequent patterns.

More importantly, in bioinformatics, we know that homologous viruses have high similarity. If we mine frequent patterns in two homologous viruses, most frequent patterns are the same. Since researchers pay more attention to functional and pathogenic divergence, the common patterns are redundant. To find the difference between the two viruses easily, NMSPs can not only represent all frequent patterns, but also effectively prune the common redundant patterns. To verify this claim, we mine the NMSPs and frequent patterns in SARS-CoV-1, SARS-CoV-2 and MERS-CoV. Experimental results show that NMSP mining is easier to find the differences between the virus sequences. The main contributions are as follows.

1. To compress frequent nonoverlapping patterns in SDB and provide the boundary information between frequent and infrequent patterns, we address NMSP mining and propose an effectiveness mining algorithm, called Nettree for NMSP mining (NetNMSP).
2. To calculate the nonoverlapping support effectively, NetNMSP adopts the backtracking strategy to iteratively search the leftmost parent nodes to get the nonoverlapping occurrences by the Netback algorithm. Meanwhile, NetNMSP employs the pattern join strategy to generate candidate patterns and the screening method to find NMSPs.
3. Experiments on biological sequences verify that not only does NetNMSP have better performance than other competitive algorithms, but also the maximal pattern mining has better compression performance than the closed pattern mining.
4. We mine NMSPs and frequent patterns in SARS-CoV-1, SARS-CoV-2 and MERS-CoV. The results show that the three viruses are similar in short patterns but different in long patterns. More importantly, NMSP mining is easier to find the differences between the virus sequences.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 addresses the definitions of the problem. Section 4 proposes NetNMSP which employs the backtracking strategy to calculate the nonoverlapping support, the pattern join strategy to generate candidate pattern, and the screening method to find NMSPs. Section 5 reports experimental results and analyzes the performance of NetNMSP on biological and sales sequences. Section 6 makes the conclusion of this paper.

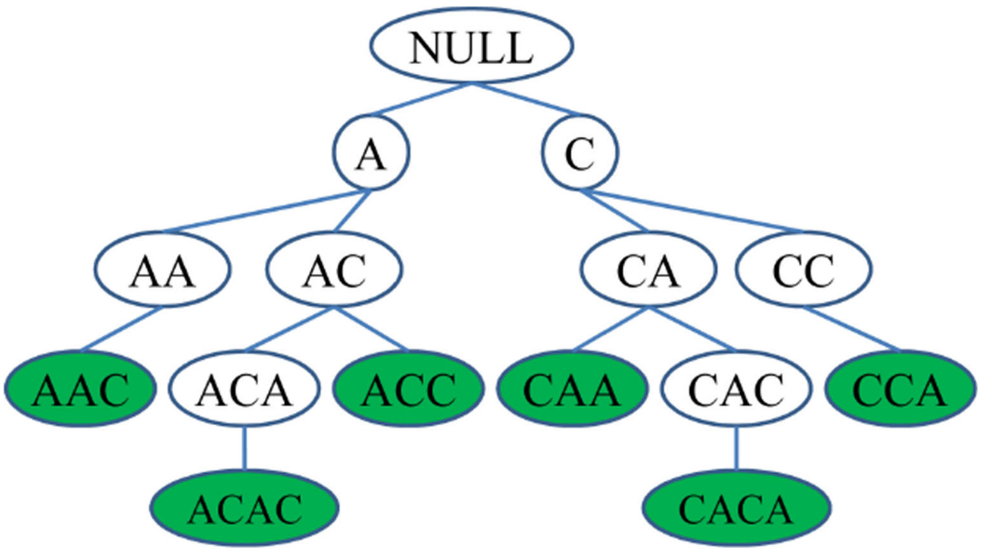


Fig. 2 Comparison between frequent patterns and NMSPs in Example 2. All nodes are frequent patterns, while green nodes are NMSPs

Table 2 Comparison of occurrences under different conditions

Condition	Support	Occurrences
No-condition	4	$<1,2,3>$, $<1,2,5>$, $<1,4,5>$, and $<3,4,5>$
One-off condition	1	$<1,2,3>$
Nonoverlapping condition	2	$<1,2,3>$ and $<3,4,5>$

2 Related work

Sequential pattern mining has been widely applied in many fields [43, 44], such as ICU patient risk prediction [45, 46] and public reactioms analysis on twitter [47]. A variety of mining methods have been investigated [48]. Some methods are derived from different data types, such as the mining of event logs [49] and transaction databases [50]. Meanwhile, some methods are derived from different tasks, such as negative sequential pattern mining [14, 15], frequent pattern discovery with tri-partition alphabets [51], utility pattern mining [52–57], and contrast pattern mining [58, 59]. Moreover, some methods are derived from different characteristics of patterns, such as frequent pattern mining [60], rare pattern mining [61], top-k pattern mining [16], closed pattern mining [62], and maximal pattern mining [63, 64].

To consider the number of occurrences for a pattern in an SDB, gap constraint sequential pattern mining was proposed, which can be divided into three types: no-condition [16, 65, 66], the one-off condition [67–69], and the nonoverlapping condition [30, 40]. The comparison of the occurrences of pattern P = ACA with gap constraint gap = [0,2] in sequence S = ACACA under different conditions is shown in Table 2.

As shown in Table 2, no-condition means that when the gap constraint is satisfied, each subsequence can be matched multiple times by different positions in a pattern. In brief, all occurrences can be acceptable. Thus, there are 4 occurrences under no-condition, i.e. {<1, 2, 3>, <1, 2, 5>, <1, 4, 5>, <3, 4, 5>}. It is easy to calculate the support under no-condition. But both its support and support rate are not monotonicity. Thus, the Apriori-like property is employed to mine all frequent patterns, which enlarges the searching space. The one-off condition means that each subsequence can be matched at most once. Thus, there is only one occurrence under the one-off condition, i.e. <1, 2, 3>. Although this method satisfies the Apriori property, calculating the support under the one-off condition is an NP-hard problem. Therefore, it is a kind of approximate mining method. As mentioned in the Introduction section, there are 2 occurrences under the nonoverlapping condition, i.e. {<1, 2, 3>, <3, 4, 5>}. Wu et al. [42] proved that calculating the support under the nonoverlapping condition can be done in polynomial time. Karim et al. [30] showed that nonoverlapping sequential pattern mining satisfies the Apriori property and proposed an effective algorithm NOSEP. Hence, nonoverlapping sequential pattern mining achieves the balance between mining completeness and mining efficiency. Table 3 shows a comparison of the related studies.

From Table 3, the work of [30] and [23] are the most related work. The differences between the above researches and this paper are as follows. First, the problems to be solved are different. Wu et al. [30] focused on mining all frequent patterns, while the work of [23] mined all closed patterns to compress the frequent patterns. Second, the methods of calculating the pattern support are different. NETGAP [30] needs to find and prune the invalid nodes after obtaining each nonoverlapping occurrence. Therefore, the time complexity of NETGAP is $O(\frac{m\times m\times n\times w}{r^2})$, where m, n, w and r are pattern length, sequence length, gap width and item number, respectively. BackTr [23] employed a backtracking strategy to find the minimal occurrence in the Nettree and did not need to find and prune the invalid nodes. Thus, the time complexity of BackTr is $O(\frac{m\times n\times w}{r^2})$. In this paper, we propose Netback to calculate supports which adopts backtracking strategy to find the maximal occurrence

Table 3 Comparison of related studies

Literature	Type of condition	Gap constraint	Support	Mining Type	Pruning strategy
Yun et al.[63]	Weighted	No	Exact	Maximal	Weight
Lee et al.[64]	Weighted	No	Approximate	Maximal	Anti-monotone
Min et al.[51]	No-Condition	Yes	Exact	All	Aprior
Li et al.[65]	No-Condition	Yes	Exact	Long	Aprior-like
Lam et al.[67]	One-off Condition	Yes	Approximate	Closed	Apriori
Wu et al.[30]	Nonoverlapping Condition	Yes	Exact	All	Apriori
Wu et al. [23]	Nonoverlapping Condition	Yes	Exact	Closed	Apriori
This paper	Nonoverlapping Condition	Yes	Exact	Maximal	Apriori

and does not need to find and prune the invalid nodes. Hence, the time complexity of Netback is also $O(\frac{m\times n\times w}{r^2})$, which is the same as that of BackTr and less than that of NETGAP. The difference between BackTr and Netback is that BackTr iteratively finds the minimal occurrences, while Netback iteratively finds the maximal occurrences.

3 Problem definition

Definition 1 (Sequence and Sequence Database) A sequence can be expressed as $S = s_1s_2\ldots s_i\ldots s_n$ $(n > 0)$, which consists of n items arranged by order, where $s_i \in \Sigma$ $(1 \leq i \leq n)$, Σ denotes the category of the items, and the size is $|\Sigma|$. A sequence database that contains T sequences can be expressed as $SDB = \{S_1, S_2, \ldots, S_T\}$.

Definition 2 (Pattern with Periodic Gap Constraint) The pattern with periodic gap constraint can be described as $P = p_1[a,b]p_2[a,b]...p_{m-1}[a,b]p_m$ $(m > 0)$ (or $P = p_1p_2\cdots p_{m-1}p_m$ with $gap = [a,b]$), where a and b are the minimal and maximal wildcards between p_i and p_{i+1} $(0 < i < m)$. A wildcard can represent any item in Σ.

Definition 3 (Occurrence) Suppose we have a sequence $S = s_1s_2s_3\ldots s_n$ and a pattern $P = p_1p_2...p_{m-1}p_m$ with $gap = [a,b]$. If $p_1 = s_{j_1}, p_2 = s_{j_2}, \ldots, p_m = s_{j_m}$, $(0 < j_1 < j_2 < \ldots < j_m \leq n)$ and $a \leq j_i - j_{i-1} - 1 \leq b$, then $occ = < j_1, j_2, \ldots, j_m >$ is an occurrence of pattern P in sequence S.

Example 3 Suppose we have a sequence $S_1 = s_1s_2s_3s_4s_5$ = ACACA and a pattern $P = p_1[0,3]p_2[0,3]p_3$ = A[0, 2]C[0, 2]A. There are four occurrences {<1, 2, 3>, <1, 2, 5>, <1, 4, 5>, <3, 4, 5>} of pattern P in sequence S_1. If gap = [0, 1], then <1, 2, 5> and <1, 4, 5> do not satisfy the gap constraint, since 5-2-1 = 2>1 and 4-1-1 = 2>1.

Definition 4 (Nonoverlapping Occurrence and Support) Suppose $occ =< j_1, j_2, \ldots, j_m >$ and $occ' =< j'_1, j'_2, \ldots, j'_m >$ are two occurrences. If and only if $j_i \neq j'_i$ $(\forall i(1 \leq i \leq m))$, then occ and occ' are two nonoverlapping occurrences. If any two occurrences in a set are nonoverlapping, then the set is called nonoverlapping occurrence set. The number of maximum nonoverlapping occurrences is the support of pattern P in sequence S under the nonoverlapping condition, which is denoted by $sup(P, S)$. The support of pattern P in $SDB = \{S_1, S_2, \cdots, S_T\}$ is $sup(P, SDB) = \sum_{t=1}^{T} sup\,(P, S_t)$.

Definition 5 (Frequent Pattern, FP) If $sup(P, SDB)$ is no less than the minimum support threshold, then pattern P is called a frequent pattern, abbreviated as FP.

Example 4 Suppose there is a sequence database SDB = { S_1 = ACACA, S_2 = CACAC }. Given a pattern P = ACA with gap = [0,2], we know that $sup(P, S_1)$ = 2, since the maximum nonoverlapping occurrence set of pattern P in sequence S_1 is {<1, 2, 3>, <3, 4, 5>}. Similarly, $sup(P, S_2) = 1$. Hence, $sup(P, SDB) = sup\,(P, S_1) + sup\,(P, S_2) = 3$. Suppose the minimum support threshold is $minsup = 2$. Pattern P is a frequent pattern in sequence database SDB, since $sup(P, SDB) = 3 > 2$.

Definition 6 (NMSP) If pattern P is a frequent pattern and all its super–patterns are infrequent, then pattern P is an NMSP.

When sequence database, gap constraints, and support threshold are given, our problem is to find all NMSPs. For example, when SDB = { S_1 = ACACA, S_2 = CACAC }, and $minsup = 2$ are given, we find all NMSPs which are {AAC,ACC,CAA,CCA,ACAC,CACA}.

4 NetNMSP

To deal with NMSP mining, there are three key factors affecting the mining performance: calculating pattern support, pruning candidate pattern, and determining NMSPs. Therefore, Section 4.1 proposes the Netback algorithm which employs the backtracking strategy to calculate the support of a pattern in a Nettree. Section 4.2 employs the pattern join strategy to generate candidate patterns. Section 4.3 adopts the screening method to find NMSPs. The NetNMSP algorithm is proposed in Section 4.4.

4.1 Netback

In this section, we review the related concepts of Nettree [66] at first. Then, we introduce the principles of NETGAP [30] and BackTr [23]. Finally, we propose algorithm Netback to calculate the support.

Obviously, the nonoverlapping occurrences are a subset of all occurrences which can be expressed by a Nettree.

Definition 7 (Nettree) Nettree is an extended tree structure, which has one or more root nodes. In a Nettree, a node can have more than one parent node. Some nodes may have the same node ID, which are located in different levels. n_i^j represents node j in the i^{th}level [70].

Definition 8 (Root-Leaf Path, RLP) Suppose a Nettree has m levels, a node in the first level is a root node, and a node in the m^{th} level is a leaf node. A path from a root node to a leaf node in a Nettree is a root-leaf path (RLP) [30].

Example 5 shows that all occurrences can be expressed by RLPs in a Nettree.

Example 5 Suppose we have a sequence $S = s_1s_2s_3s_4s_5s_6s_7s_8s_9s_{10}s_{11}s_{12}s_{13}$ = ATTCATCACATCA and a pattern $P = p_1[0,3]p_2[0,3]p_3[0,3]p_4$ = A[0,3]T[0,3]C[0,3]A.

We create root n_1^1, since s_1 ='A' = p_1. Using this method, all roots n_1^5, n_1^8, n_1^{10}, and n_1^{13} can be created. We create node n_2^3, since s_3 = 'T' = p_2 and there is a parent-child relationship between nodes n_1^1 and n_2^3, since s_1 and s_3 satisfy the gap constraints [0,3], i.e. 0 ≤3-1-1 = 1≤3. We create node n_2^6, since s_6 = 'T' = p_2 and there is a parent-child relationship between nodes n_1^5 and n_2^6, since s_5 and s_6 satisfy the gap constraints [0,3]. But we cannot create a parent-child relationship between nodes n_1^1 and n_2^6, since s_1 and s_6 do not satisfy the gap constraints [0,3], i.e. 6-1-1>3. Similarly, we create all nodes and parent-child relationships in a Nettree shown in Fig. 3. The following characteristics should be noticed. The Nettree has 5 roots, n_1^1, n_1^5, n_1^8, n_1^{10}, and n_1^{13}. Some nodes have the same node ID. For example, nodes n_1^5 and n_4^5 are two nodes with the same ID 5. Some nodes have more than one parent. For example, node n_3^7 has two parents, nodes n_2^3 and n_2^6. There are 12 occurrences <1, 2, 4, 5>, <1, 2, 4, 8>, <1, 3, 4, 5>, <1, 3, 4, 8>, <1, 3, 7, 8>, <1, 3, 7, 10>, <5, 6, 7, 8>, <5, 6, 7, 10>, <5, 6, 9, 10>, <5, 6, 9, 13>, <8, 11, 12, 13>, and <10, 11, 12, 13>. All of them can be expressed by RLPs in a Nettree. For example, <1, 3, 7, 8> is an occurrence and its corresponding path $< n_1^1, n_2^3, n_3^7, n_4^8 >$ is an RLP.

According to the definition of nonoverlapping occurrences, any two occurrences cannot use the same node in the Nettree. For example, <1, 2, 4, 5> and <1, 3, 7, 8> are two overlapping occurrences, since they use a common node n_1^1, while <1, 2, 4, 5> and <5, 6, 7, 8> are two nonoverlapping occurrences, since there is no common used node in the Nettree.

To calculate the support in a Nettree, we employ an iterative strategy to find the maximal occurrence.

Definition 9 (Maximal Occurrence) Suppose $occ = < j_1, j_2, \ldots, j_m >$ is an occurrence of pattern P in sequence S. For any occurrence $occ' = < j_1', j_2', \ldots, j_m' >$, if $j_i \geq j_i'$ ($\forall i(1 \leq i \leq m)$), then occurrence occ is the maximal occurrence of pattern P in sequence S.

Definition 10 (Maximal Root-Leaf Path, MRLP) An RLP that iterates the rightmost child node (i.e. the largest child node) from the rightmost root node to its leaf node in a Nettree is the maximal root-leaf path (MRLP).

Although the NETGAP algorithm [30] employs the Nettree structure to calculate the pattern support accurately, it has to find and prune the invalid nodes after obtaining each nonoverlapping occurrence. Therefore, the time complexity of NETGAP is $O(\frac{m\times m\times n\times w}{r^2})$, where m, n, w, and r are the length of pattern and sequence, the maximum gap, and the size of characters (i.e. $|\Sigma|$), respectively. To calculate the support effectively, we propose Netback which employs the backtracking strategy and iteratively finds the maximal occurrences. The time complexity of Netback is reduced to $O(\frac{m\times n\times w}{r^2})$, which is the same as that of BackTr [23] which iteratively finds the minimal occurrence.

The following two examples illustrate the principles of NETGAP, BackTr, and Netback, respectively.

Example 6 As shown in Example 5 and Fig. 3, NETGAP [30] iteratively finds the nonoverlapping occurrences from the leftmost root to its leaf. Firstly, starting from the leftmost root n_1^1, NETGAP finds the leftmost parent child (i.e. node n_2^2). Iterating this step, NETGAP finds RLP <

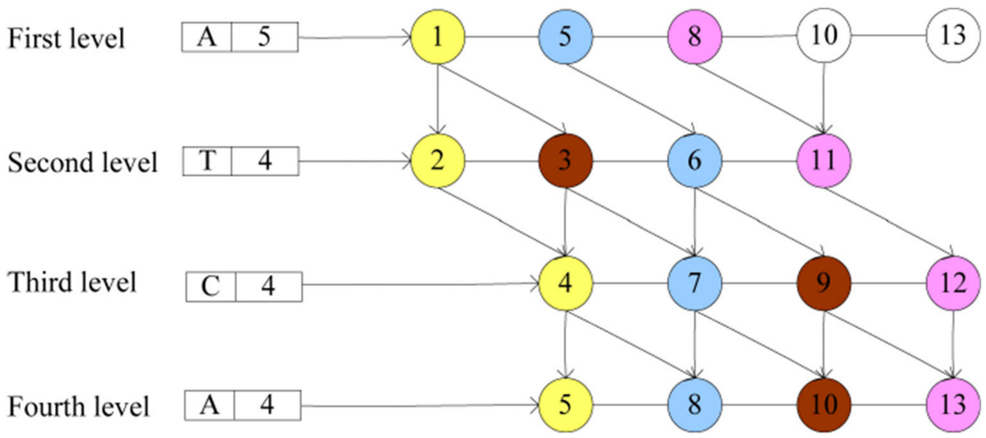


Fig. 3 The Nettree of pattern P = A[0,3]T[0,3]C[0,3]A in sequence S = ATTCATCACATCA. The Nettree has four levels, since the length of pattern P is four. Nodes n_1^1, n_1^5, n_1^8, n_1^{10}, and n_1^{13} are the roots. Nodes n_4^5, n_4^8, n_4^{10}, and n_4^{13} are the leaves. Nodes n_1^5 and n_4^5 are two nodes with the same ID 5 in the first and fourth levels, respectively. Node n_3^7 has two parents nodes n_2^3 and n_2^6. Path $< n_1^1, n_2^3, n_3^7, n_4^8 >$ is an RLP and its corresponding occurrence is <1, 3, 7, 8>

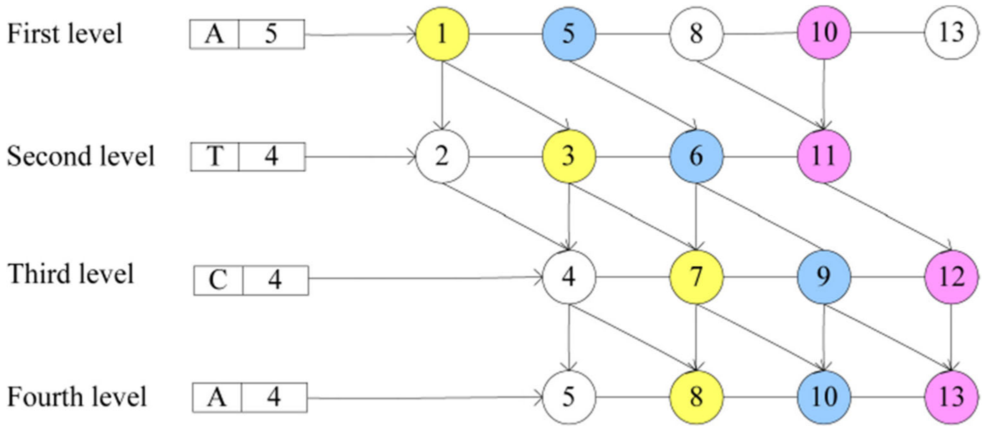


Fig. 4 Three maximal occurrences found by Netback

$n_1^1, n_2^2, n_3^4, n_4^5$ >, which is an MRLP and its corresponding occurrence is <1, 2, 4, 5>. Then nodes n_1^1, n_2^2, n_3^4, and n_4^5 are pruned. After that, to find all the nonoverlapping occurrences, NETGAP has to find and prune the invalid nodes. Therefore, node n_2^3 is pruned. Then, NETGAP gets the second occurrence <5, 6, 7, 8>. Similarly, after pruning nodes n_1^5, n_2^6, n_3^7, and n_4^8, nodes n_3^9 and n_4^{10} are found and pruned. Finally, NETGAP finds the third occurrence <8, 11, 12, 13>. Hence, NETGAP finds three nonoverlapping occurrences.

BackTr [23] also iteratively finds the nonoverlapping occurrences from the leftmost root to its leaf, but it does not need to find and prune the invalid nodes, since it employs the backtracking strategy. Therefore, BackTr also finds the same three nonoverlapping occurrences as NETGAP: <1, 2, 4, 5>, <5, 6, 7, 8> and <8, 11, 12, 13>. But the time complexity of BackTr is less than that of NETGAP.

We show that Netback can find the same number of occurrences. The time complexity of Netback is the same as that of BackTr and less than that of NETGAP.

Example 7 As shown in Example 5, Netback adopts the backtracking strategy to iteratively find the nonoverlapping occurrences from the rightmost root to its leaf. Firstly, starting from the rightmost root n_1^{13}, Netback finds its rightmost child node. Since n_1^{13} has no child, Netback find the next root n_1^{10}. It is easy to know that there is only one RLP $< n_1^{10}, n_2^{11}, n_3^{12}, n_4^{13} >$ with root n_1^{10}. Thus, Netback finds the first occurrence $< 10, 11, 12, 13 >$. Similarly, Netback finds the second occurrence $< 5, 6, 9, 10 >$. Now, Netback selects root n_1^1 which has two children: n_2^2 and n_2^3. To find the maximal occurrence, Netback iteratively selects the rightmost child node. Therefore, n_2^3 is selected, and then n_3^7 is selected. n_3^7 has two children: n_4^8 and n_4^{10}. Since n_4^{10} is used in occurrence $< 5, 6, 9, 10 >$, Netback selects n_4^8. Thus, Netback finds the third occurrence $< 1, 3, 7, 8 >$. Hence, Netback also finds three different nonoverlapping occurrences shown in Fig. 4, and the number of occurrences is the same as that of NETGAP and BackTr.

From above two examples, we know that both NETGAP and Netback can find the same number of nonoverlapping occurrences and Netback is more effective than NETGAP, since NETGAP has to find and prune the invalid nodes, while Netback employs the backtracking strategy and does not need to find and prune the invalid nodes, which reduces the time complexity. The main process of the Netback algorithm is shown as follows. First, Netback creates the Nettree according to P and S. Then, Netback iteratively finds the rightmost nonoverlapping occurrences from the rightmost root with the backtracking strategy until all roots are visited. The Netback algorithm is shown in Algorithm 1.

Algorithm 1 Netback: Calculate the support of pattern P in sequence S.

```
Require: Sequence $S$, Pattern $P$(the length of $P$ is $m$), $gap = [a, b]$
Ensure: $sup(S, P)$
1: Create a Nettree according to $P$ and $S$;
2: for each rightmost $root$ do
3:     $occ \leftarrow$ Get an MRLP by iteratively selecting the rightmost child with the backtracking strategy;
4:     if occ $\neq$ NULL then
5:         $sup(S, P)$++;
6:         Delete $occ$;
7:     end if
8: end for
9: return $sup(S, P)$
```

Theorem 1 *The space and time complexities of Netback are both $O(m \times n \times w)$ in the worst case and $O(\frac{m \times n \times w}{r^2})$ in the average case, where m, n, w, and r are pattern length, sequence length, gap width ($b - a + 1$, gap = $[a, b]$), and item number $|\Sigma|$, respectively.*

Proof A Nettree has m levels. Each level has no more than n nodes. Each node has no more than w children. Therefore, the space complexity of Netback is $O(m \times n \times w)$ in the worst case. Netback employs the backtracking strategy to find the minimal occurrences. Thus, each node could be visited no more than once. As a result, the time complexity of Netback is the same as the space complexity and is also $O(m \times n \times w)$ in the worst case. In the average case, each level has no more than $\frac{n}{r}$ nodes, and each node has no more than $\frac{w}{r}$ children. Therefore, the space and time complexities of Netback are $O(\frac{m \times n \times w}{r^2})$ in the average case. □

4.2 Pattern join strategy

Since the nonoverlapping sequential pattern mining satisfies the Apriori property, we employ the pattern join strategy [31] to generate candidate patterns.

Definition 11 (Prefix and Suffix Pattern and Pattern join) Given event items α, β (α, β $\in \Sigma$), if there is a pattern Q = $P\alpha$, then P is called the prefix pattern of Q, denoted by *Prefix* (Q). If pattern R = βP, then P is called the suffix pattern of R, denoted by *Suffix* (R). Patterns Q and R are the super–patterns of pattern P, and pattern P is the sub-pattern of patterns Q and R. Suppose there are two patterns Q = $P\alpha$, and R = βP. We get a new super-pattern $P' = R \oplus Q =$ β$P\alpha$ by pattern join since *Prefix* (Q) = *Suffix* (R) = P.

Although the set enumeration tree strategy can be used to generate the candidate patterns, the following example shows that the pattern join strategy outperforms the set enumeration tree strategy.

Example 8 Suppose we have a sequence S = $s_1s_2s_3s_4s_5s_6s_7s_8s_9s_{10}s_{11}s_{12}s_{13}$ = ATTCATCACATCA, Σ = {A, T, C}, gap = [0,3], and the minimum support threshold $minsup$ = 3. Set F_3 including all 11 frequent patterns with length 3 is {AAA , AAC, ACA, ACC, ATA, ATC, CAA, CAC, CCA, TCA, TCC}. The principle of the set enumeration tree strategy is adding a character in Σ at the end of each pattern to generate a new pattern. Thus, each pattern can generate $|\Sigma|$ candidate patterns. Taking frequent pattern AAA as an example, the set enumeration tree strategy generates three candidate patterns, AAAA, AAAC, and AAAT since Σ = {A, T, C}. Therefore, this strategy generates 11*3=33 candidate patterns. However, the pattern join strategy generates 18 candidate patterns shown in set C_4 = { AAAA, AAAC, AACA, AACC, ACAA, ACAC, ACCA, ATCA, ATCC, CAAA, CAAC, CACA, CACC, CCAA, CCAC, TCAA, TCAC, TCCA }. The number of candidate patterns generated by the set enumeration tree strategy and pattern join strategy is compared in Table 4. We can see that with increasing pattern length, more candidate patterns can be pruned by the pattern join strategy. Hence, the pattern join strategy is more effective than the set enumeration tree strategy.

4.3 Screening method

Suppose pattern P is a frequent pattern. According to Definition 6, if pattern P is an NMSP, we should generate all its super–patterns and determine that all these super–patterns are infrequent. Obviously, this method is very complex. In this section, we propose the screening method to find NMSPs. The principle is shown as follows.

Suppose all frequent patterns with length m are stored in frequent pattern set F_m. We generate all candidate patterns with length $m + 1$ using F_m and store them in candidate pattern set C. If pattern P in set C is a frequent pattern, then its prefix and suffix patterns are not NMSPs. Thus, we delete *Prefix* (P) and *Suffix* (P) from set F_m. According to this method, after checking all patterns in set C, the remaining patterns in set F_m are NMSPs. It is worth emphasizing that if pattern P is infrequent, we cannot say that its prefix and suffix patterns are NMSPs. The following example illustrates the principle of the screening method.

Example 9 In Example 8, pattern "AACA" is a frequent pattern since sup ("AACA", S) = 3. Thus, its prefix pattern "AAC" and suffix pattern "ACA" are not NMSPs. Pattern "AACC" is an infrequent pattern since sup("AACC", S) = 2. Its prefix pattern "AAC" and suffix pattern "ACC" are not NMSPs since their super-patterns "AACA" and "ACCA" are frequent patterns whose supports are both 3.

Example 10 In Example 8, we use 11 frequent patterns F_3 = {AAA , AAC, ACA, ACC, ATA, ATC, CAA, CAC, CCA, TCA, TCC} to generate 18 candidate patterns, C_4 =

Table 4 Number of different length patterns

Length of pattern	*Length* = 1	*Length* = 2	*Length* = 3	*Length* = 4	*Length* = 5	*Length* = 6	Total
Number of frequent patterns	3	7	11	9	2	0	32
Number of candidate patterns by set enumeration tree	3	9	21	33	27	6	99
Number of candidate patterns by pattern join	3	9	17	18	8	0	55

{AAAA, AAAC, AACA, AACC, ACAA, ACAC, ACCA, ATCA, ATCC, CAAA, CAAC, CACA, CACC, CCAA, CCAC, TCAA, TCAC, TCCA}. Pattern "AACA" is a frequent pattern. Thus, patterns "AAC" and "ACA" in F_3 are not NMSPs and deleted. Similarly, patterns "ACAA", "ACAC", "ACCA", "ATCA", "CACA", "TCAA", "TCAC", and "TCCA" are frequent patterns stored in F_4. Therefore, patterns "ACC", "ATC", "CAA", "CAC", "CCA", "TCA", and "TCC" are not NMSPs and deleted. Hence, the remaining patterns in F_3 are "AAA" and "ATA" which are NMSPs. We iterate the above process to find all NMSPs.

4.4 NetNMSP

In this section, we propose the NetNMSP algorithm and analyze the space and time complexities of it.

The main process of the NetNMSP algorithm is shown as follows. First, NetNMSP generates candidate pattern set C with pattern length $m+1$ using frequent pattern set F_m. Then, NetNMSP calculates the support of pattern P in set C. If pattern P is frequent, then store it in frequent pattern set F_{m+1}, and its prefix and suffix patterns in F_m are not NMSPs and deleted. NetNMSP iterates the above process until all patterns in set C have been checked. All patterns remaining in F_m are NMSPs and are stored in F_{max}. Finally, NetNMSP iterates the above process until pattern set C is empty. The NetNMSP algorithm is shown in Algorithm 2.

Algorithm 2 NetNMSP: Mine all NMSPs.

```
Require: Sequence database $SDB$, $minsup$, $gap = [a, b]$.
Ensure: $F_{max}$: NMSP set
 1: Scan sequence database SDB once, calculate the
    support of each event item, and store the frequent
    patterns with length 1 in $F_1$;
 2: $m \leftarrow 1$;
 3: $C \leftarrow$ PatternJoin ($F_m$) ;
 4: while $C \neq$ NULL do
 5:     for each $P$ in $C$ do
 6:         $support \leftarrow 0$;
 7:         for each $S$ in $SDB$ do
 8:             $support \leftarrow$ support + Netback($S$, $P$, $gap$);
 9:         end for
10:         if $support \geq minsup$ then
11:             $F_{m+1} \leftarrow F_{m+1} \bigcup P$;
12:             Delete Prefix ($P$) and Suffix ($P$) in $F_m$;
13:         end if
14:     end for
15:     $F_{max} \leftarrow F_{max} \bigcup F_m$;
16:     $C \leftarrow$ PatternJoin ( $F_{m+1}$);
17:     $m \leftarrow m + 1$;
18: end while
19: return $F_{max}$;
```

Theorem 2 *The space complexity of the NetNMSP algorithm is $O(m \times (\frac{n \times w}{r^2} + L))$, where m, n, L, w, and r are the length of the longest pattern, the length of the longest sequence in database, the number of the candidate patterns, gap width ($b - a + 1$, $gap = [a, b]$), and item number (i.e.$|\Sigma|$), respectively.*

Proof The space of the NetNMSP algorithm consists of two parts, the space of frequent patterns and candidate patterns and the space of Netback. It is easy to know that the space complexity of the first part is $O(m \times L)$. Meanwhile, the space complexity of the Netback algorithm is $O(\frac{m \times n \times w}{r^2})$. Hence, the space complexity of the NetNMSP algorithm is $O(m \times (\frac{n \times w}{r^2} + L))$. □

Theorem 3 *The time complexity of the NetNMSP algorithm is $O(\frac{m \times N \times w \times L}{r^2})$, where N is the total length of all sequences.*

Proof According to Theorem 1, the time complexity of Netback is $O(\frac{m \times n \times w}{r^2})$ for a sequence. The time complexity of Netback for all sequences is therefore $O(\frac{m \times N \times w}{r^2})$. Thus, for all candidate patterns, the time complexity of NetNMSP is $O(\frac{m \times N \times w \times L}{r^2})$. Since the binary search is used in PatternJoin to generate candidate patterns, the time complexity of generating all candidate patterns is $O(L \times log(L))$. Therefore, the time complexity of the NetNMSP algorithm is $O(\frac{m \times N \times w \times L}{r^2} + L \times log(L)) = O(\frac{m \times N \times w \times L}{r^2})$. □

5 Experimental results and analysis

Section 5.1 explains the benchmark datasets and the baseline methods. Section 5.2 verifies the correctness of the NetNMSP algorithm. Section 5.3 reports the time efficiency of the NetNMSP algorithm. Section 5.4 compares the compression performance of maximal sequential pattern mining and closed sequential pattern mining. Section 5.5 reports the NMSPs in a biological application of COVID-19.

5.1 Benchmark datasets and baseline methods

The experimental running environment is: Intel(R) Core(TM) i5-3120M, 2.50GHZ CPU, 8GB RAM, Windows 7, and the 64-bit operating system computer. The program development environment is VC++6.0. To verify the performance of the NetNMSP algorithm, this paper adopts DNA, protein, and virus sequences as experimental data. All algorithms and datasets can be downloaded from https://github.com/wuc567/Pattern-Mining/tree/master/NetNMSP. The datasets are shown in Table 5.

Table 5 Description of datasets

Dataset	Type	Source	Number of sequences	Description	Total length
DNA1[1]	DNA	Homo Sapiens AL158070	1	Single	6,000
DNA2	DNA	Homo Sapiens AL158070	1	Single sequence	8,000
DNA3	DNA	Homo Sapiens AL158070	1	Single sequence	10,000
DNA4	DNA	Homo Sapiens AL158070	1	Single sequence	12,000
DNA5	DNA	Homo Sapiens AL158070	1	Single sequence	14,000
DNA6	DNA	Homo Sapiens AL158070	1	Single sequence	16,000
SDB1[2]	Protein	ASTRAL95_1_161	507	Multiple/unequal length	91,875
SDB2	Protein	ASTRAL95_1_161	338	Multiple/unequal length	62,985
SDB3	Protein	ASTRAL95_1_161	169	Multiple/unequal length	32,503
SDB4	Protein	ASTRAL95_1_171	590	Multiple/unequal length	109,424
SDB5	Protein	ASTRAL95_1_171	400	Multiple/unequal length	73,425
SDB6	Protein	ASTRAL95_1_171	200	Multiple/unequal length	37,327
Baby1[3]	Babysale	Sales of baby products	1,636	Multiple/unequal length	73,272
Baby2	Babysale	Sales of baby products	2,077	Multiple/unequal length	94,152
Baby3	Babysale	Sales of baby products	2,544	Multiple/unequal length	115,088
Baby4	Babysale	Sales of baby products	3,057	Multiple/unequal length	137,941
Super1[4]	Superstore	Superstore time series	1	Single sequence	100,001
Super2	Superstore	Superstore time series	1	Single sequence	120,001
Super3	Superstore	Superstore time series	1	Single sequence	140,001
Super4	Superstore	Superstore time series	1	Single sequence	161,048
TSS[5]	Human genes	Transcriptional Start Sites	200	Multiple / equal length	20,000
SARS-CoV-1[6]	DNA of the virus	Severe Acute Respiratory Syndrome Coronavirus	1	Single sequence	29,751
SARS-CoV-2[7]	DNA of the virus	Severe Acute Respiratory Syndrome Coronavirus 2	1	Single sequence	29,903
MERS-CoV[8]	DNA of the virus	Middle East Respiratory Syndrome Coronavirus	1	Single sequence	30,119

[1] Homo Sapiens AL158070 is a DNA sequence, which can be downloaded from https://www.ncbi.nlm.nih.gov/nuccore/AL158070.11

[2] ASTRAL is a database of protein sequences based on the SCOP database, which can be downloaded from https://scop.berkeley.edu/astral/subsets/ver=1.61 and https://scop.berkeley.edu/astral/ver=1.71

[3] Babysale is a sales dataset for infant products, which can be downloaded from https://tianchi.aliyun.com/dataset/dataDetail?dataId=45

[4] Superstore is a sales dataset from SuperStore, which can be downloaded from https://tianchi.aliyun.com/dataset/dataDetail?dataId=93285

[5] TSS (Transcriptional Start Sites) contains 200 human genes of positive and negative classes, which comes from http://dbtss.hgc.jp/

[6] SARS-CoV-1 (Severe Acute Respiratory Syndrome Coronavirus 1) is the gene sequence of virus causing in 2003, which was reported by Ref [71] and can be downloaded from https://www.ncbi.nlm.nih.gov/nuccore/30271926?report=fasta

[7] SARS-CoV-2 (Severe Acute Respiratory Syndrome Coronavirus 2) is the gene sequence of virus causing COVID-19, which was reported by Ref [72] and can be downloaded from https://www.ncbi.nlm.nih.gov/nuccore/MN908947.3?report=fasta

[8] MERS-CoV (Middle East Respiratory Syndrome Coronavirus) was reported by Ref [73] and can be downloaded from https://www.ncbi.nlm.nih.gov/nuccore/NC_019843.3?report=fasta

It should be noted that, SDB1-6 are protein sequences composed of 20 amino acids, and DNA1-6, SARS-CoV-1, SARS-CoV-2, MERS-CoV, and TSS are DNA sequences composed of four deoxynucleotides, A, T, C, and G.

In this paper, GSgrow-MAX, NOSEP-MAX, MAXB, MAXD, NetNMSP-S, and NOSEP are employed as competitive algorithms whose principles are shown as follows.

1. *GSgrow-MAX*: GSgrow [40] algorithm is an efficient mining algorithm, which employs INSgrow algorithm to approximately calculate the support. Based on GSgrow, GSgrow-MAX further finds the maximal patterns in the frequent patterns.

2. *NOSEP-MAX*: To verify the effectiveness of Netback, NOSEP-MAX adopts NETGAP to calculate the support which was employed in NOSEP [30].
3. *MAXB* and *MAXD*: To verify the pruning efficiency of pattern join, we propose the MAXB and MAXD algorithms, which employ the breadth-first and depth-first searching for the set enumeration tree to generate the candidate patterns, respectively.
4. *NetNMSP-S*: To verify the efficiency of the screening method, NetNMSP-S is proposed and mines NMSPs according to Definition 6.
5. *NOSEP* [30] and *NetNCSP* [23]: To verify the compression effect of NMSPs, we employ the NOSEP and NetNCSP algorithms to mine all frequent patterns and closed patterns.

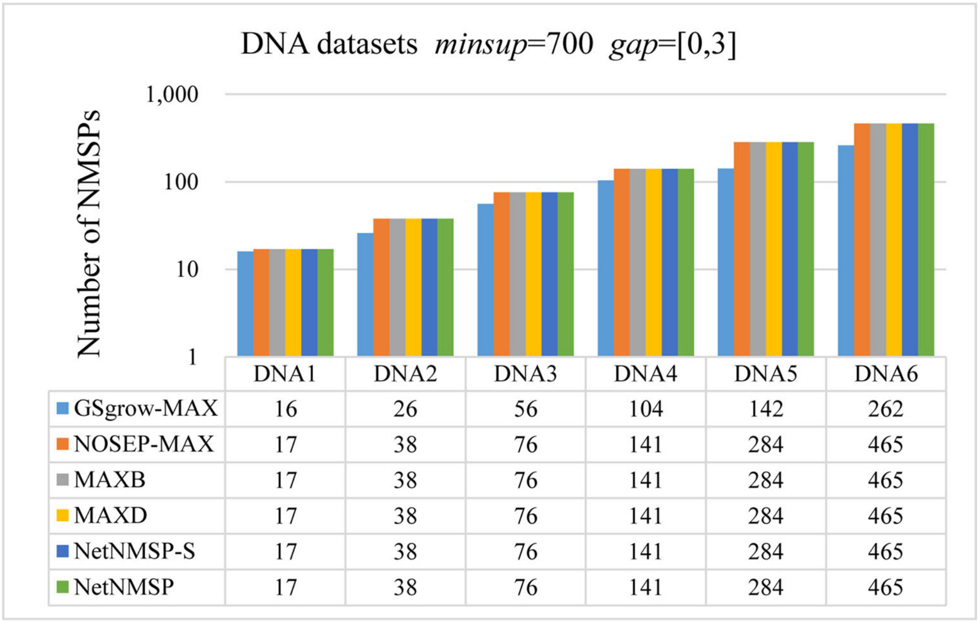
5.2 Mining efficiency

To further illustrate the Efficiency of the NetNMSP algorithm, we perform experiments on single-sequence DNA datasets (DNA1 to DNA6) and multi-sequence protein datasets (SDB1 to SDB6). We select GSgrow-Max, NOSEP-MAX, MAXB, MAXD, NetNMSP-S, and NetNMSP as the competitive algorithms. Considering the difference between DNA and protein datasets, we set $minsup$ = 700, gap = [0,3] on the DNA datasets, and $minsup$ = 1800, gap = [0,10] on the protein datasets. The comparisons of the number of NMSPs, running time and the number of candidate patterns on DNA and protein datasets are shown in Figs. 5, 6, 7, 8, 9 and 10, respectively.

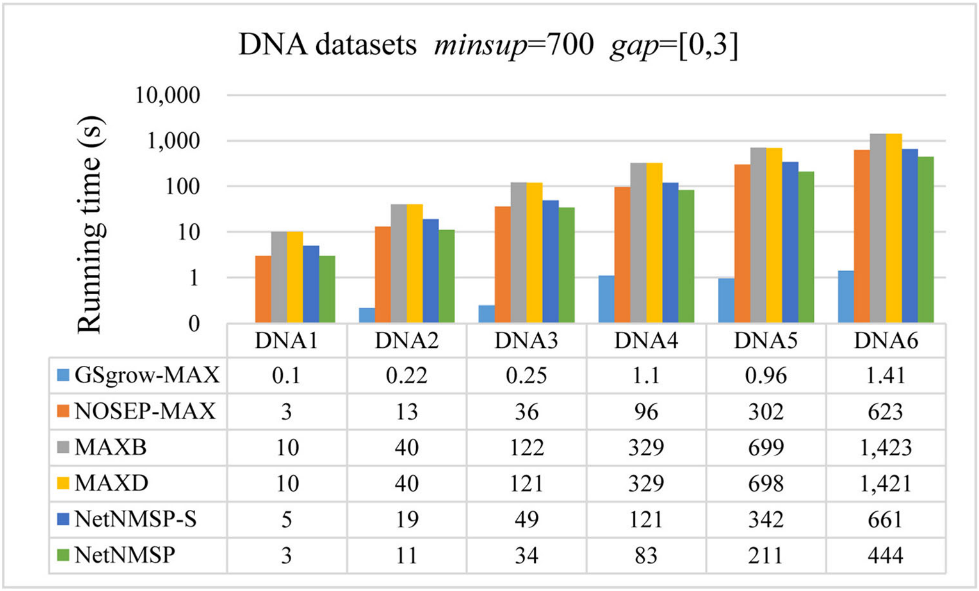
The results indicate the following observations.

1. Although GSgrow-MAX is faster than NetNMSP, NetNMSP has better performance than GSgrow-MAX. From Figs. 6 and 9, we know that GSgrow-MAX is faster than all other algorithms. However, from Figs. 5 and 8, we know that GSgrow-MAX discovers less NMSPs. For example, on DNA6, GSgrow-MAX runs for 1.41 s, while NetNMSP runs for 444 s. However, GSgrow-MAX only finds 262 NMSPs, while NetNMSP finds 465. The reasons are as follows. GSgrow-MAX employs INSgrow to calculate the support. INSgrow is an approximate algorithm and the time complexity of INSgrow is less than that of Netback. Therefore, GSgrow-MAX runs faster than NetNMSP. However, GSgrow-MAX is also an approximate algorithm. Thus, some frequent patterns cannot be discovered by GSgrow. Moreover, with the increase of the sequence, more NMSPs will be lost. For instance, on DNA1, GSgrow-MAX discovers 16 out of 17 NMSPs, while on DNA6, GSgrow-MAX discovers 262 out of 465 NMSPs, since the length of DNA1 is shorter than that of DNA6. The protein datasets are composed of multiple sequences, and each sequence is short. Thus, GSgrow-MAX can find out most NMSPs on protein datasets. Hence, NetNMSP outperforms GSgrow-MAX.
2. NetNMSP has better performance than NOSEP-MAX. NetNMSP checks the same number of candidate patterns as NOSEP-MAX and finds the same NMSPs as NOSEP-MAX, and NetNMSP runs faster than NOSEP-MAX. For example, on SDB1, both NetNMSP and NOSEP-MAX check 2,045 candidate patterns and discover 179 NMSPs, and NetNMSP runs for 239 s,

Fig. 5 Comparison of number of NMSPs on DNA1 to DNA6



	DNA1	DNA2	DNA3	DNA4	DNA5	DNA6
🟦 GSgrow-MAX	16	26	56	104	142	262
🟧 NOSEP-MAX	17	38	76	141	284	465
⬜ MAXB	17	38	76	141	284	465
🟨 MAXD	17	38	76	141	284	465
🟦 NetNMSP-S	17	38	76	141	284	465
🟩 NetNMSP	17	38	76	141	284	465

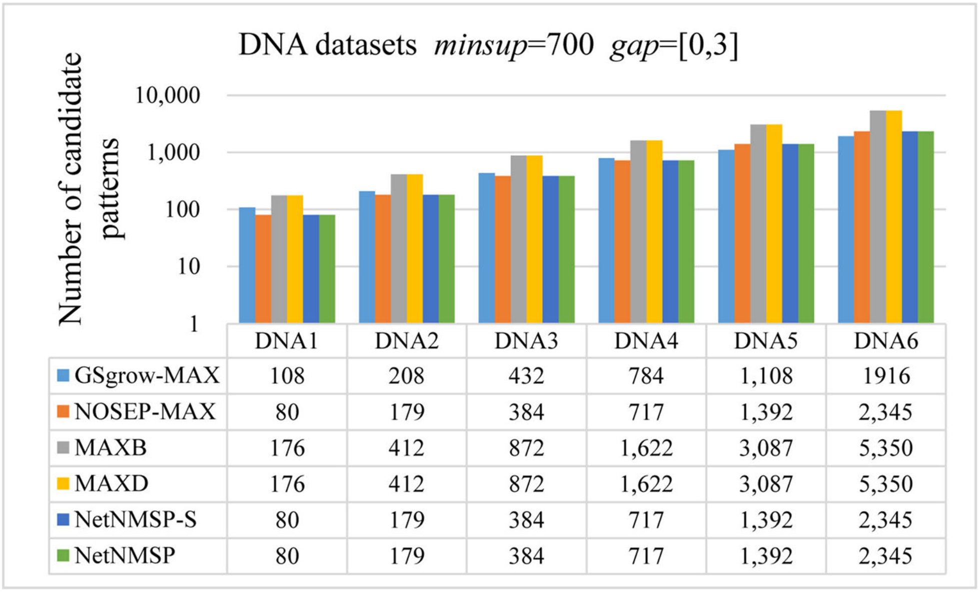


	DNA1	DNA2	DNA3	DNA4	DNA5	DNA6
GSgrow-MAX	0.1	0.22	0.25	1.1	0.96	1.41
NOSEP-MAX	3	13	36	96	302	623
MAXB	10	40	122	329	699	1,423
MAXD	10	40	121	329	698	1,421
NetNMSP-S	5	19	49	121	342	661
NetNMSP	3	11	34	83	211	444

Fig. 6 Comparison of running time on DNA1 to DNA6

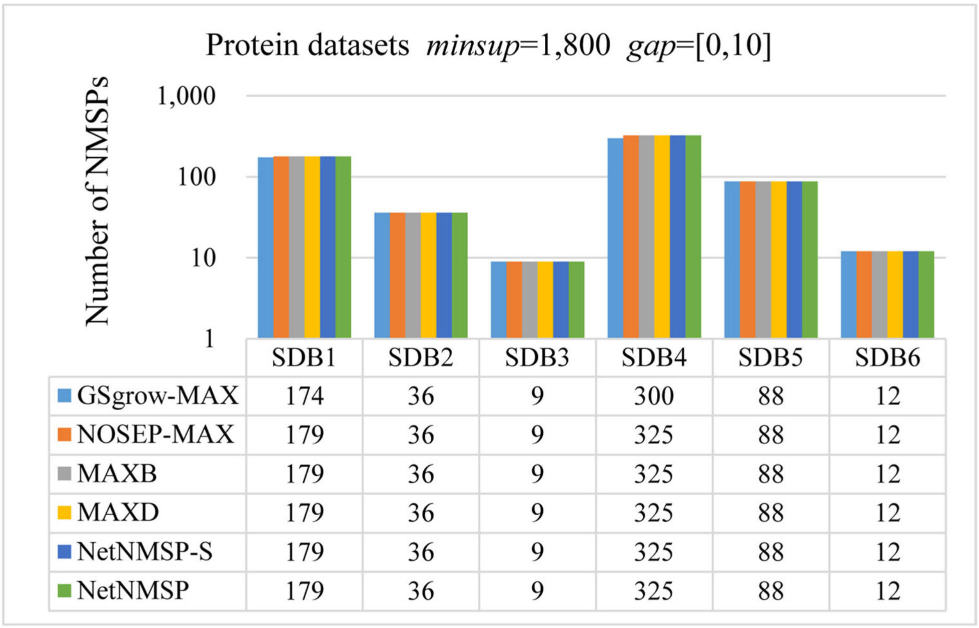
while NOSEP-MAX runs for 371 s. The reasons are as follows. NetNMSP and NOSEP-MAX employ the same candidate pattern generation strategy, and Netback and NETGAP are exact algorithms. Therefore, both NetNMSP and NOSEP-MAX check the same number of candidate patterns and discover the same number of NMSPs. However, NETGAP has to find and prune the invalid nodes, while Netback does not. Thus, the time complexity of Netback is less than that of NETGAP. Hence, NetNMSP runs faster than NOSEP-MAX.

3. NetNMSP has better performance than MAXB and MAXD. NetNMSP finds the same NMSPs as MAXB and MAXD, checks less candidate patterns than MAXB and MAXD, and runs faster than MAXB and MAXD. For example, on DNA2, NetNMSP, MAXB and MAXD all find 38 NMSPs, and check 179, 412 and 412 candidate patterns, respectively, and cost 11, 40 and 40 s, respectively. The reasons are as follows. The three algorithms all adopt Netback to calculate the support, but employ different candidate pattern generation strategies: the pattern join strategy, breadth-first and depth-first searching for the set enumeration tree. In Section 4.2, we show that the pattern join strategy outperforms the set enumeration tree strategy. Hence, NetNMSP checks less candidate patterns than MAXB and MAXD, and runs faster than MAXB and MAXD.



	DNA1	DNA2	DNA3	DNA4	DNA5	DNA6
GSgrow-MAX	108	208	432	784	1,108	1916
NOSEP-MAX	80	179	384	717	1,392	2,345
MAXB	176	412	872	1,622	3,087	5,350
MAXD	176	412	872	1,622	3,087	5,350
NetNMSP-S	80	179	384	717	1,392	2,345
NetNMSP	80	179	384	717	1,392	2,345

Fig. 7 Comparison of number of candidate patterns on DNA1 to DNA6



	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
🟦 GSgrow-MAX	174	36	9	300	88	12
🟧 NOSEP-MAX	179	36	9	325	88	12
⬜ MAXB	179	36	9	325	88	12
🟨 MAXD	179	36	9	325	88	12
🟦 NetNMSP-S	179	36	9	325	88	12
🟩 NetNMSP	179	36	9	325	88	12

Fig. 8 Comparison of number of NMSPs on SDB1 to SDB6

4. NetNMSP has better performance than NetNMSP-S. NetNMSP checks the same number of candidate patterns as NetNMSP-S and finds the same NMSPs as NetNMSP-S, and runs faster than NetNMSP-S. For example, on SDB4, both NetNMSP and NetNMSP-S check 3,882 candidate patterns and find 325 NMSPs, and run for 574 and 806 s, respectively. The reason is that NetNMSP employs the screening method to find NMSPs, while NetNMSP-S uses the definition. Hence, NetNMSP runs faster than NetNMSP-S.

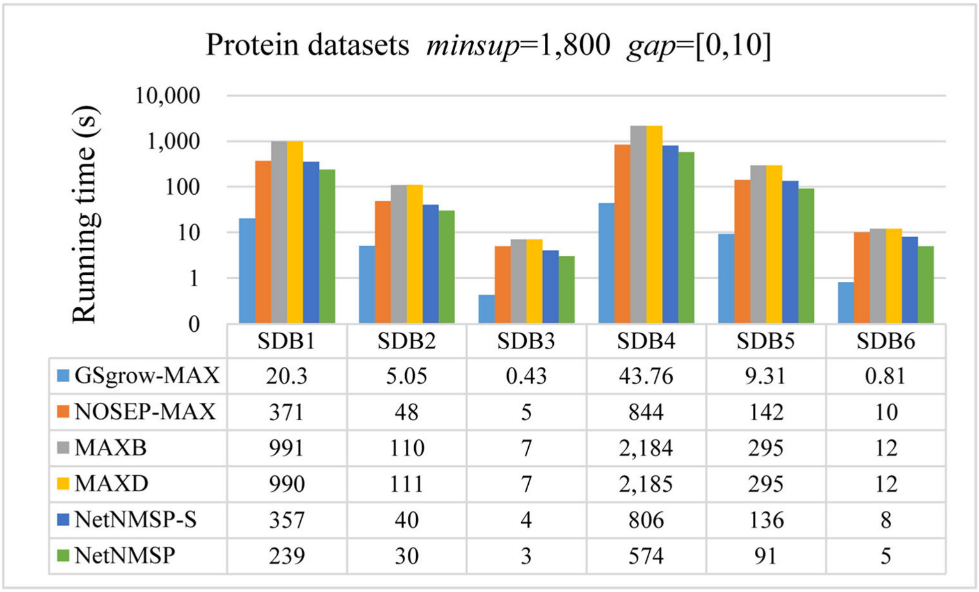
In summary, NetNMSP outperforms all competitive algorithms.

5.3 Scalability analysis

To analyze the scalability of the NetNMSP algorithm, we select two datasets in Table 5: multi-sequence dataset BABYSALE and single-sequence dataset Superstore. We set gap = [0,3] and $minpau$ = 6,000 for Babysale and gap = [0,7] and $minpau$ = 6,000 for Superstore. The number of NMSPs, running time and number of candidate patterns are shown in Figs. 11, 12 and 13, respectively.

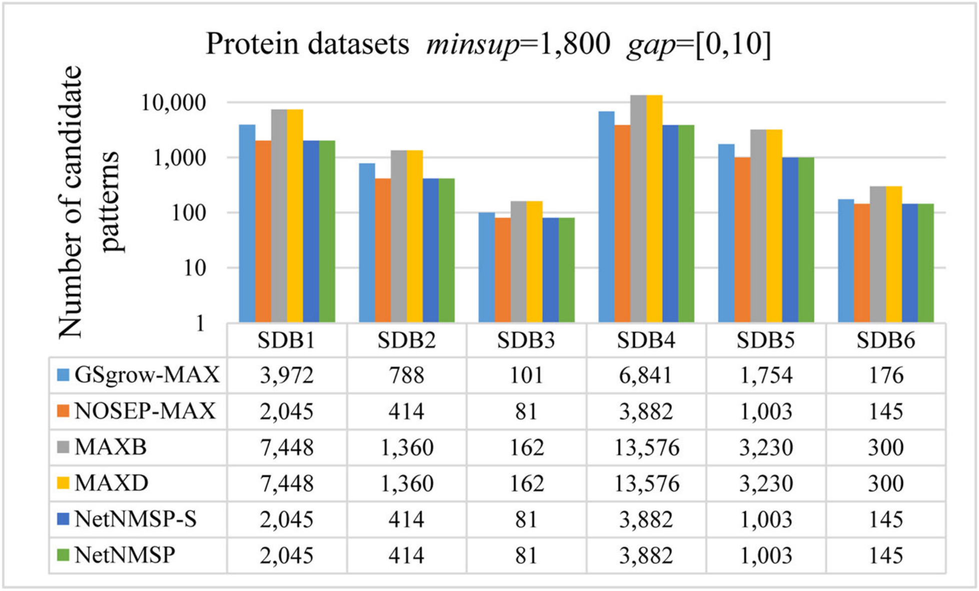
The results show the following observations.

With the increase of sequence length, the number of NMSPs, running time and the number of candidate patterns



	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
🟦 GSgrow-MAX	20.3	5.05	0.43	43.76	9.31	0.81
🟧 NOSEP-MAX	371	48	5	844	142	10
⬜ MAXB	991	110	7	2,184	295	12
🟨 MAXD	990	111	7	2,185	295	12
🟦 NetNMSP-S	357	40	4	806	136	8
🟩 NetNMSP	239	30	3	574	91	5

Fig. 9 Comparison of running time on SDB1 to SDB6



	SDB1	SDB2	SDB3	SDB4	SDB5	SDB6
🟦 GSgrow-MAX	3,972	788	101	6,841	1,754	176
🟧 NOSEP-MAX	2,045	414	81	3,882	1,003	145
⬜ MAXB	7,448	1,360	162	13,576	3,230	300
🟨 MAXD	7,448	1,360	162	13,576	3,230	300
🟦 NetNMSP-S	2,045	414	81	3,882	1,003	145
🟩 NetNMSP	2,045	414	81	3,882	1,003	145

Fig. 10 Comparison of number of candidate patterns on SDB1 to SDB6

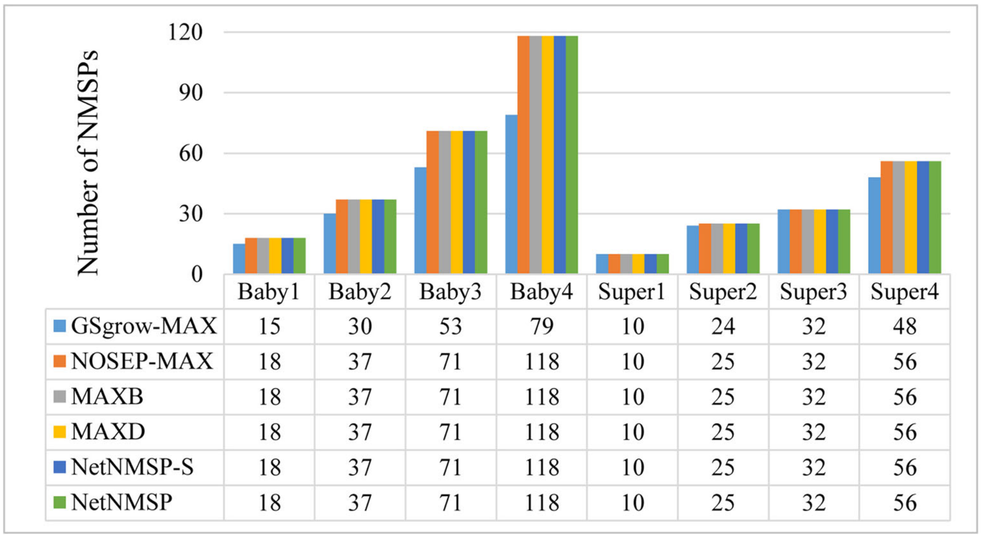
increase. For example, the lengths of Baby1 and Baby4 are 73,272 and 137,941, respectively. There are 18 and 118 NMSPs, the running time is 46 and 892 s, and the number of candidate patterns is 88 and 637 on Baby1 and Baby4 datasets, respectively. This phenomenon can be found on SUPERSTORE dataset. The reasons are as follows. With the increase of sequence length, more frequent patterns can be found. Therefore, there are more candidate patterns. As a result, the running time increases. More importantly, more patterns are NMSPs.

Meanwhile, from Fig. 12, we can see that NOSEP-MAX, MAXB, MAXD, NetNMSP-S and NetNMSP have the same tendency in running time. However, the tendency of NetNMSP is more gentle than other algorithms which means that our algorithm guarantees the best scalability on large scale datasets.

5.4 Compression Performance

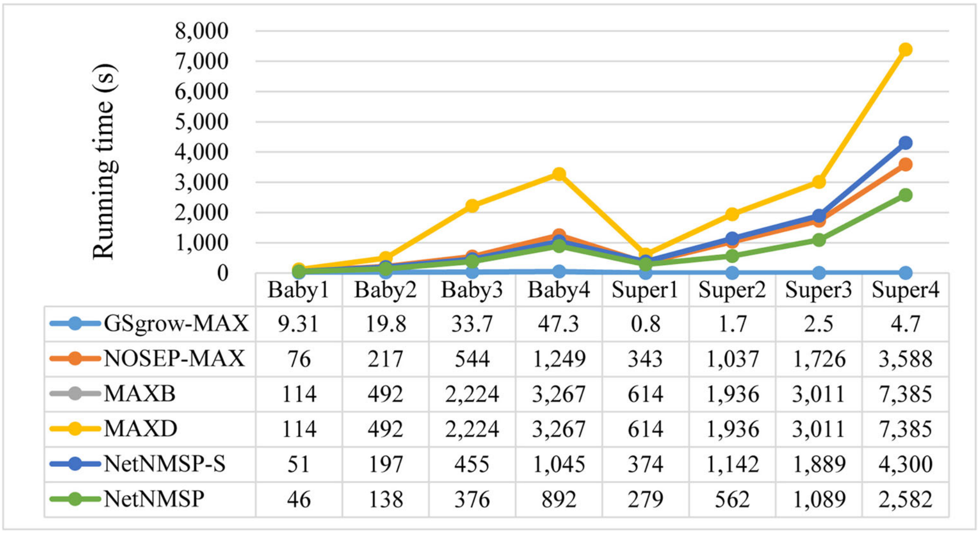
To show the compression performance of NMSP mining, we report the number of nonoverlapping frequent sequential patterns mined by NOSEP [30], the nonoverlapping closed sequential patterns mined by NetNCSP [23], and NMSPs mined by NetNMSP. In the experiments, we also report the proportion of closed patterns and maximal patterns in frequent patterns, expressed by rate_close (rate_close $=\frac{cp}{fp}$) and rate_max (rate_max $=\frac{mp}{fp}$), where cp, mp, and fp are the number of nonoverlapping closed patterns, maximal patterns, and frequent patterns, respectively.

To evaluate the compression performance of NMSPs, we report the number of nonoverlapping frequent patterns, closed patterns, and NMSPs on single-sequence DNA2 and



	Baby1	Baby2	Baby3	Baby4	Super1	Super2	Super3	Super4
🟦 GSgrow-MAX	15	30	53	79	10	24	32	48
🟧 NOSEP-MAX	18	37	71	118	10	25	32	56
⬜ MAXB	18	37	71	118	10	25	32	56
🟨 MAXD	18	37	71	118	10	25	32	56
🟦 NetNMSP-S	18	37	71	118	10	25	32	56
🟩 NetNMSP	18	37	71	118	10	25	32	56

Fig. 11 Comparison of number of NMSPs



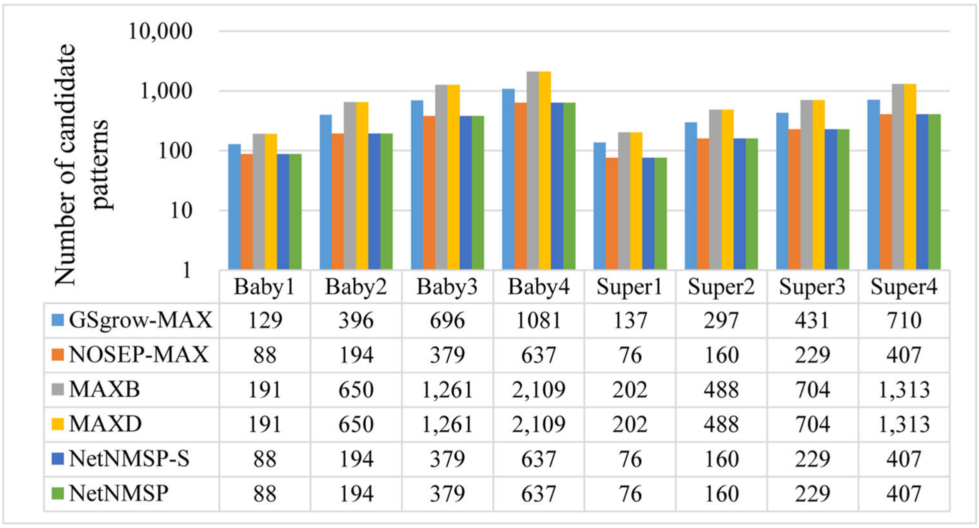
	Baby1	Baby2	Baby3	Baby4	Super1	Super2	Super3	Super4
GSgrow-MAX	9.31	19.8	33.7	47.3	0.8	1.7	2.5	4.7
NOSEP-MAX	76	217	544	1,249	343	1,037	1,726	3,588
MAXB	114	492	2,224	3,267	614	1,936	3,011	7,385
MAXD	114	492	2,224	3,267	614	1,936	3,011	7,385
NetNMSP-S	51	197	455	1,045	374	1,142	1,889	4,300
NetNMSP	46	138	376	892	279	562	1,089	2,582

Fig. 12 Comparison of running time

multiple-sequence datasets SDB5. On DNA2, we increase *gap* from [0,1] to [0,6] with *minsup* = 800. On SDB5, we increase *gap* from [0,5] to [0,15] with *minsup* = 1800. To further show the consistence of the compression performance of NMSPs, we also use different support thresholds. Single-sequence DNA dataset DNA4, multiple-sequence DNA dataset TSS, and multiple-sequence protein dataset SDB2 are selected. On DNA4 and TSS, we decrease *minsup* from 950 to 450 with *gap* = [0,3]. On SDB2, we decrease *minsup* from 2000 to 1250 with *gap* = [0,15]. The experimental results are shown in Figs. 14, 15, 16, 17 and 18.

The results indicate the following observations.

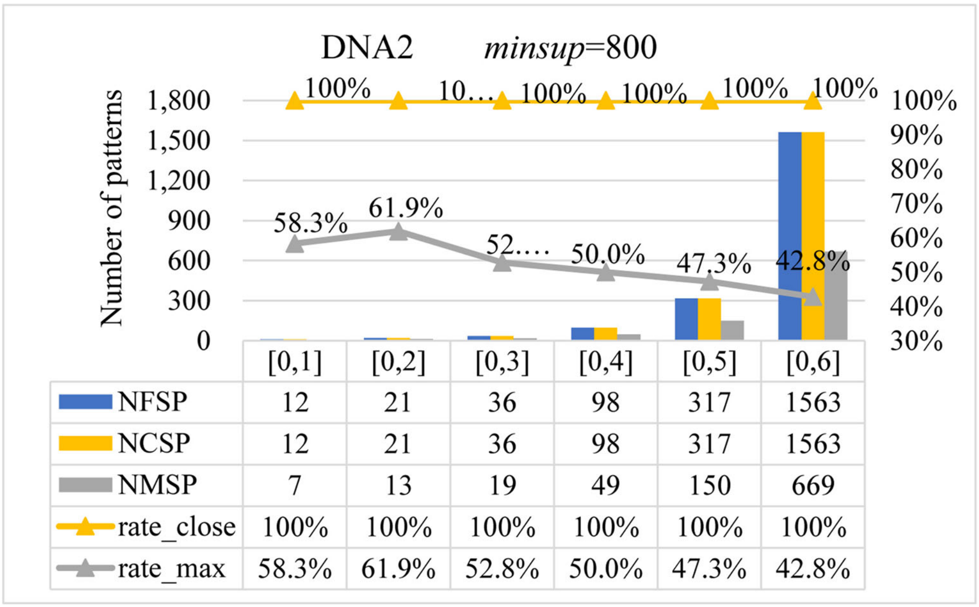
1. From Figs. 14–18, all experiment results show that NMSP mining has better compression performance than nonoverlapping closed pattern mining in all cases, such as single sequence, multiple-sequences, different gap constraints, and different support thresholds. In all experiments the number of closed patterns is the same as that of frequent patterns, while the number of NMSPs is less than that of frequent patterns. For example, in Fig. 14, the number of frequent patterns, closed patterns, and NMSPs are 317, 317, and 150, respectively. The same phenomena can be found in Figs. 15–18. The reasons are as follows. For nonoverlapping closed pattern mining, if pattern P and its supper-pattern Q are frequent patterns and have the same support, then pattern P can be compressed. However, for NMSP mining, if pattern P and its supper-pattern Q are frequent patterns, then pattern P can be compressed no matter whether its support is the same as that of pattern Q. Therefore, NMSP mining is easy to obtain better compressing performance than nonoverlapping closed pattern mining.



	Baby1	Baby2	Baby3	Baby4	Super1	Super2	Super3	Super4
GSgrow-MAX	129	396	696	1081	137	297	431	710
NOSEP-MAX	88	194	379	637	76	160	229	407
MAXB	191	650	1,261	2,109	202	488	704	1,313
MAXD	191	650	1,261	2,109	202	488	704	1,313
NetNMSP-S	88	194	379	637	76	160	229	407
NetNMSP	88	194	379	637	76	160	229	407

Fig. 13 Comparison of number of candidate patterns

Fig. 14 Comparison on DNA2 with *minsup* = 800

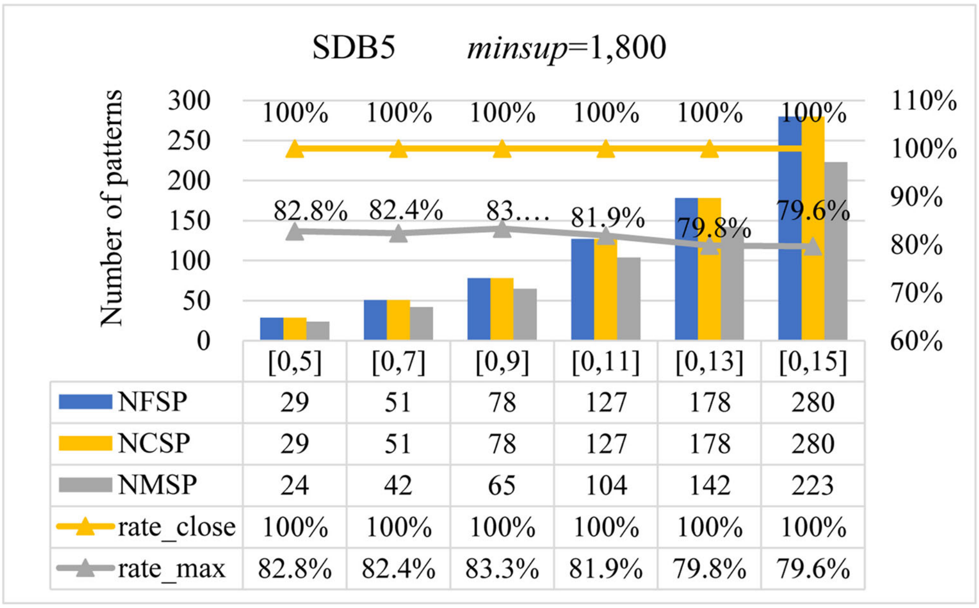


	[0,1]	[0,2]	[0,3]	[0,4]	[0,5]	[0,6]
▬ NFSP	12	21	36	98	317	1563
▬ NCSP	12	21	36	98	317	1563
▬ NMSP	7	13	19	49	150	669
▲ rate_close	100%	100%	100%	100%	100%	100%
▲ rate_max	58.3%	61.9%	52.8%	50.0%	47.3%	42.8%

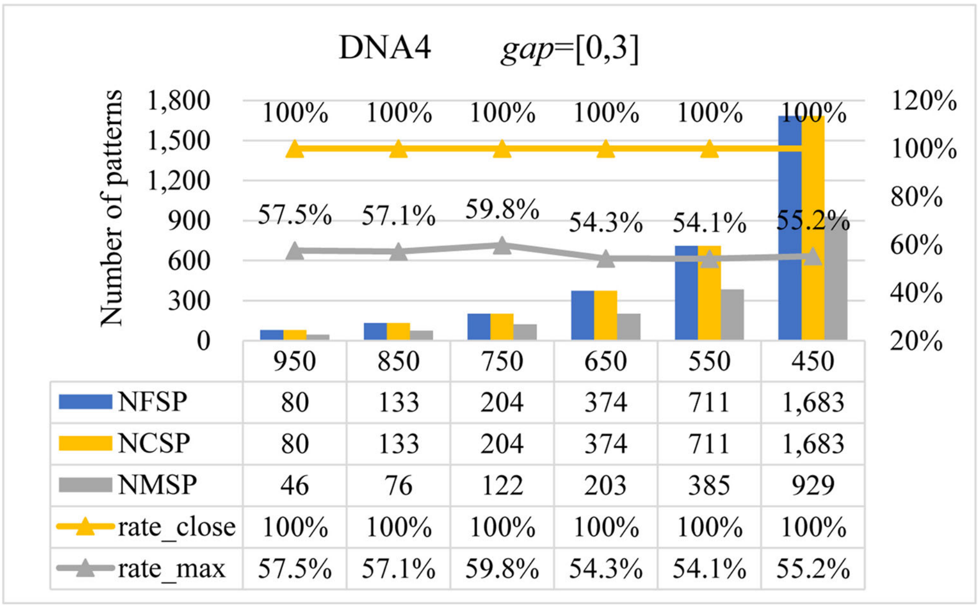
2. The less |Σ| is, the better compressing ability of NMSP mining is. From Table 5, we know that |Σ| of DNA2, DNA4, and TSS are all 4, and |Σ| of SDB5 and SDB2 are both 20. From Figs. 14–18, NMSP mining remains about 50% to 65% frequent patterns with |Σ| = 4, while NMSP mining remains about 75% to 85% with |Σ| = 20. For example, on DNA2, NMSP mining remains 58.3% frequent patterns with *gap* = [0,1] and *minsup* = 800, on DNA4, remains 57.5% frequent patterns with *gap* = [0,3] and *minsup* = 950, and on TSS, remains 56.7% frequent patterns with *gap* = [0,3] and *minsup* = 950. However, on SDB5, NMSP mining remains 82.8% frequent patterns with *gap* = [0,5] and *minsup* = 1,800, and on SDB2, remains 77.3% frequent patterns with *gap* = [0,15] and *minsup* = 2,000. The reasons are as follows. When the number of the frequent patterns is the same with different size of Σ, apparently, the less |Σ| is, the longer the maximal length of the frequent patterns is. Since the subpatterns of the frequent patterns are also frequent patterns, which could be compressed by NMSPs. Therefore, many short frequent patterns can be compressed when |Σ| is less.

In summary, NMSP mining has better compression performance than nonoverlapping closed pattern mining in many cases.

Fig. 15 Comparison on SDB5 with *minsup* = 1800



	[0,5]	[0,7]	[0,9]	[0,11]	[0,13]	[0,15]
▬ NFSP	29	51	78	127	178	280
▬ NCSP	29	51	78	127	178	280
▬ NMSP	24	42	65	104	142	223
▲ rate_close	100%	100%	100%	100%	100%	100%
▲ rate_max	82.8%	82.4%	83.3%	81.9%	79.8%	79.6%



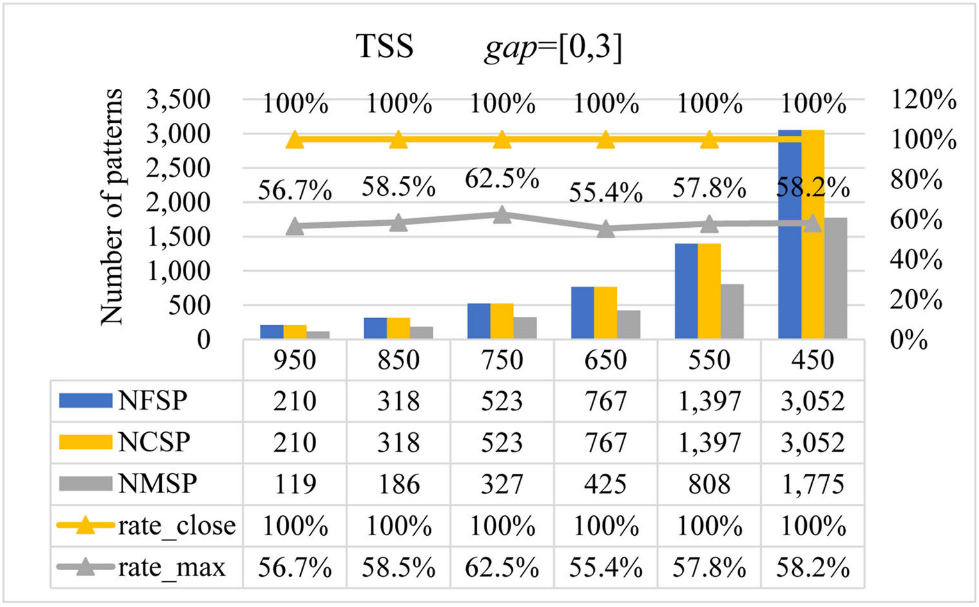
	950	850	750	650	550	450
▬ NFSP	80	133	204	374	711	1,683
▬ NCSP	80	133	204	374	711	1,683
▬ NMSP	46	76	122	203	385	929
▲ rate_close	100%	100%	100%	100%	100%	100%
▲ rate_max	57.5%	57.1%	59.8%	54.3%	54.1%	55.2%

Fig. 16 Comparison on DNA4 with *gap* = [0,3]

5.5 Case study

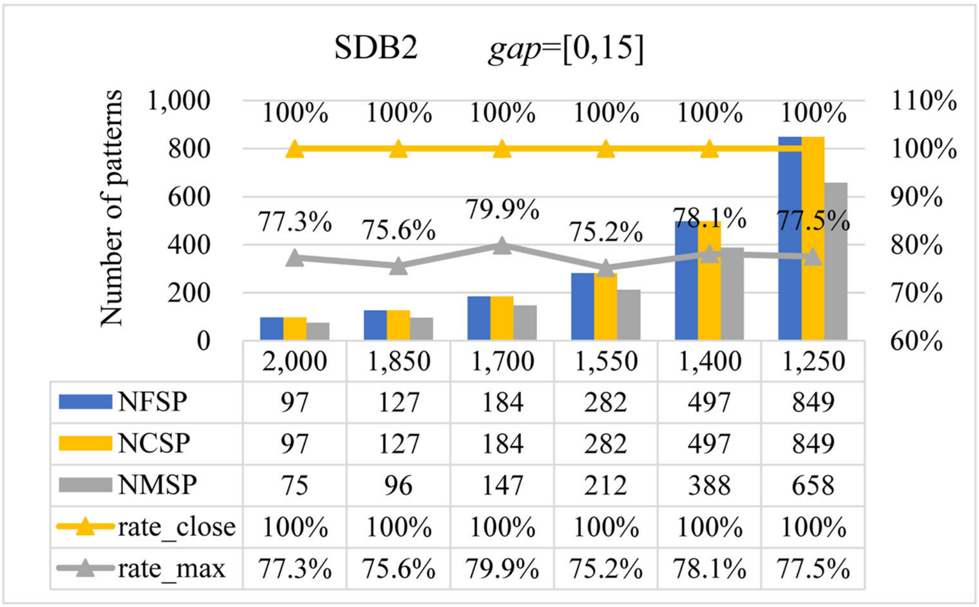
On February 11, 2020, the novel coronavirus pneumonia was named as COVID-19 by the World Health Organization (WHO). Meanwhile, the International Committee on Taxonomy of Viruses announced the official name of a new coronavirus: Severe Acute Respiratory Syndrome Coronavirus 2 (SARS-CoV-2). The disease caused a global disaster, infecting more than 200,000,000 people and killing more than 4,500,000 by August 29, 2021. Many researchers have studied SARS-CoV-2 from different aspects. For example, Nawaz et al. [74] adopted sequential pattern mining method to find hidden patterns which can be used to examine the evolution and variations in COVID-19 strains.

The DNA of the three viruses, SARS-CoV-1, SARS-CoV-2, and MERS-CoV, is single sequence. To study their complete genome, in this section, we set gap = [0,4] and $minsup$ = 2500. We employ intersection, union, and rate to evaluate the similarity of each two sequences, where intersection of two sets of mined patterns A and B is the set of patterns which are in both A and B, union is the set of patterns which are in A or B, and $rate = \frac{t}{u}$, where t and u are the number of patterns in intersection and union, respectively. We report the number of frequent patterns and NMSPs on two sequences with different lengths. The



	950	850	750	650	550	450
▬ NFSP	210	318	523	767	1,397	3,052
▬ NCSP	210	318	523	767	1,397	3,052
▬ NMSP	119	186	327	425	808	1,775
▲ rate_close	100%	100%	100%	100%	100%	100%
▲ rate_max	56.7%	58.5%	62.5%	55.4%	57.8%	58.2%

Fig. 17 Comparison on TSS with *gap* = [0,3]



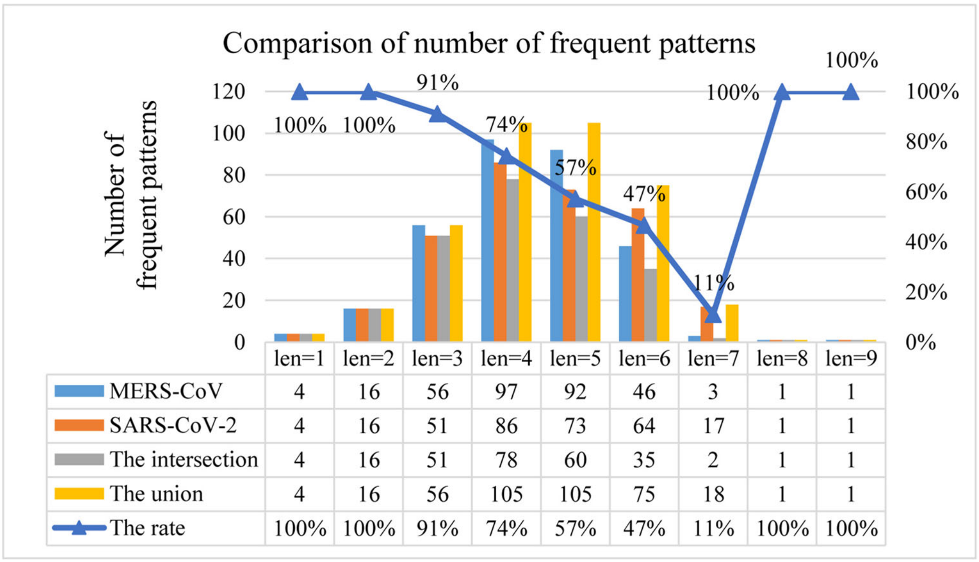
	2,000	1,850	1,700	1,550	1,400	1,250
▬ NFSP	97	127	184	282	497	849
▬ NCSP	97	127	184	282	497	849
▬ NMSP	75	96	147	212	388	658
▲ rate_close	100%	100%	100%	100%	100%	100%
▲ rate_max	77.3%	75.6%	79.9%	75.2%	78.1%	77.5%

Fig. 18 Comparison on SDB2 with *gap* = [0,15]

comparison results between MERS-CoV and SARS-CoV-1, SARS-CoV-1 and SARS-CoV-2, and MERS-CoV and SARS-CoV-2 are shown in Figs. 19, 20, 21, 22, 23 and 24.

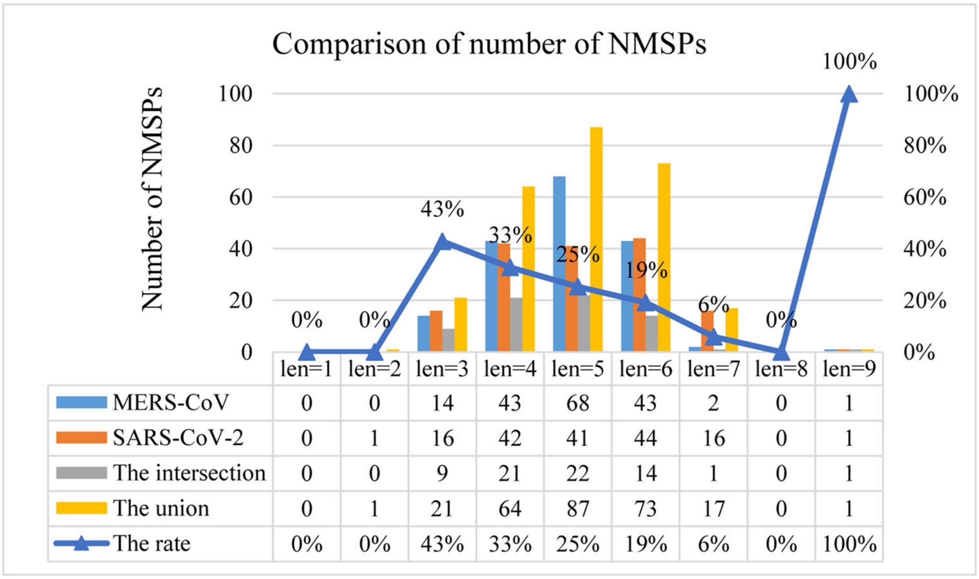
The results indicate the following observations.

1. Both frequent pattern mining and NMSP mining show that most of the same patterns are short patterns. In Figs. 19 and 20, there are 316 and 313 frequent patterns and 171 and 161 NMSPs in MERS-CoV and SARS-CoV-2, respectively. In these patterns, 248 frequent patterns and 68 NMSPs are the same, and most of them are short patterns. For example, when the pattern lengths are less than 6, we know that 209 frequent patterns and 52 NMSPs are the same. Meanwhile, the same phenomenon can also be found in Figs. 21 to 24. The possible reasons are as follows. The basic structure of MERS-CoV, SARS-CoV-1 and SARS-CoV-2 are the same. Therefore, the short NMSPs are the same. However, the three viruses have many different characteristics, which lead to different frequent patterns and NMSPs in long patterns.
2. NMSP mining not only effectively compresses the frequent patterns, but also is easier to find the difference between each two viruses. For example, there are 268 and 313 frequent patterns in SARS-CoV-1 and SARS-CoV-2, respectively, while there are 150 and 161 NMSPs in SARS-CoV-1 and SARS-CoV-2, respectively. Therefore, NMSPs remain no more than 60% frequent patterns. More importantly, NMSP



	len=1	len=2	len=3	len=4	len=5	len=6	len=7	len=8	len=9
▬ MERS-CoV	4	16	56	97	92	46	3	1	1
▬ SARS-CoV-2	4	16	51	86	73	64	17	1	1
▬ The intersection	4	16	51	78	60	35	2	1	1
▬ The union	4	16	56	105	105	75	18	1	1
▲ The rate	100%	100%	91%	74%	57%	47%	11%	100%	100%

Fig. 19 Comparison of number of frequent patterns between MERS-CoV and SARS-CoV-2

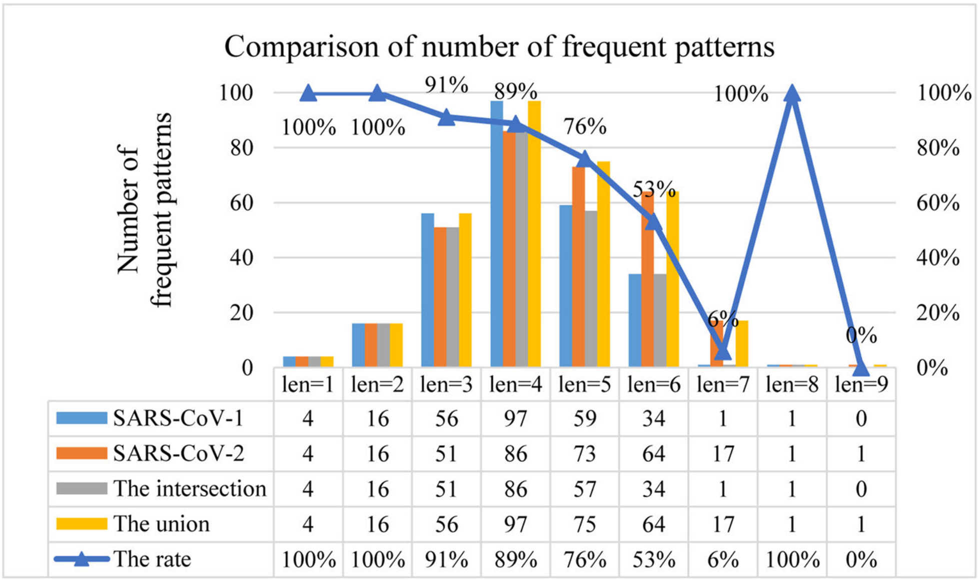


	len=1	len=2	len=3	len=4	len=5	len=6	len=7	len=8	len=9
MERS-CoV	0	0	14	43	68	43	2	0	1
SARS-CoV-2	0	1	16	42	41	44	16	0	1
The intersection	0	0	9	21	22	14	1	0	1
The union	0	1	21	64	87	73	17	0	1
The rate	0%	0%	43%	33%	25%	19%	6%	0%	100%

Fig. 20 Comparison of number of NMSPs between MERS-CoV and SARS-CoV-2

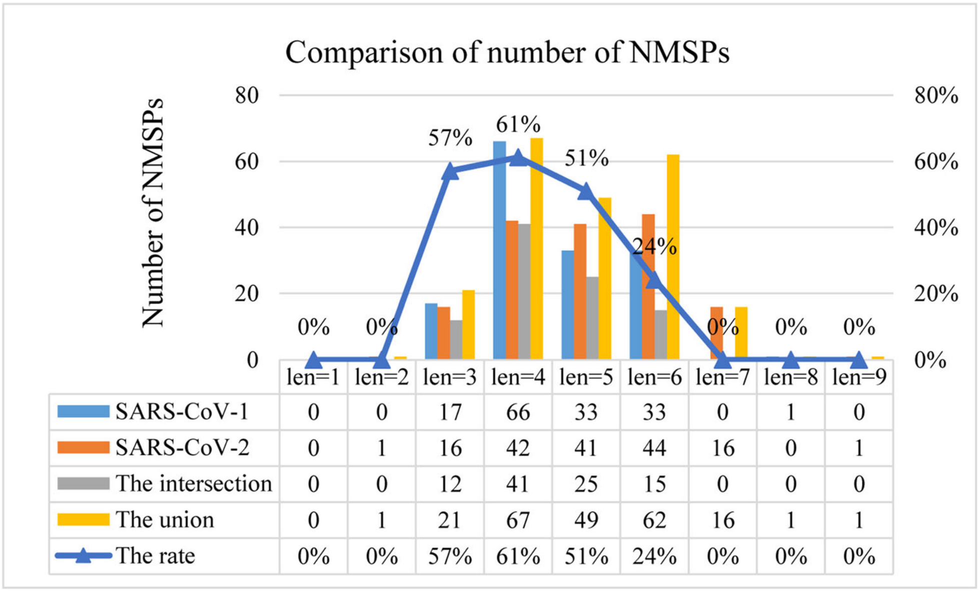
mining is easier to find different patterns from each two viruses. For example, there are both 97 frequent patterns with length 4 in MERS-CoV and SARS-CoV-1, respectively. Among them, 87 patterns are the same. Therefore, the intersection and union are 87 and 117, respectively. Thus, the rate is 74% in frequent patterns. However, there are 43 and 66 NMSPs with length 4 in MERS-CoV and SARS-CoV-1, respectively. The intersection and union are 33 and 76, respectively, resulting in a rate of 43% in NMSPs. Hence, NMSP mining makes it easier to find the difference between each two viruses.

3. SARS-CoV-1 is more similar to SARS-CoV-2 than MERS-CoV. From Fig. 20, there are 171 and 161 NMSPs in MERS-CoV and SARS-CoV-2, respectively. Among them, 68 patterns are the same. Therefore, the intersection and union are 68 and 263, respectively. Thus, the total similarity between MERS-CoV and SARS-CoV-2 is about 26%. However, there are 150 and 161 NMSPs in SARS-CoV-1 and SARS-CoV-2, respectively. The intersection and union are 93 and 218, respectively, resulting in a rate of SARS-CoV-1 and SARS-CoV-2 is 43%. Hence, compared with MERS-CoV, SARS-CoV-1 is more similar to SARS-CoV-2.
4. SARS-CoV-1 is more similar to MERS-CoV than SARS-CoV-2. From the above analysis, we know that the total similarity between MERS-CoV and SARS-CoV-2 is about 26%. However, from Fig. 24, there are 171 and 150 NMSPs in MERS-CoV and SARS-CoV-1,



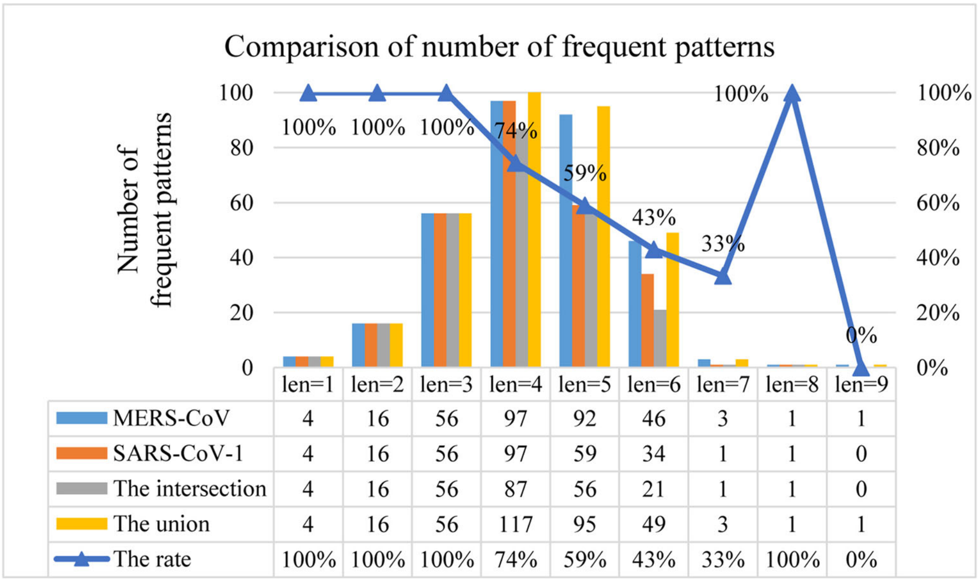
	len=1	len=2	len=3	len=4	len=5	len=6	len=7	len=8	len=9
SARS-CoV-1	4	16	56	97	59	34	1	1	0
SARS-CoV-2	4	16	51	86	73	64	17	1	1
The intersection	4	16	51	86	57	34	1	1	0
The union	4	16	56	97	75	64	17	1	1
The rate	100%	100%	91%	89%	76%	53%	6%	100%	0%

Fig. 21 Comparison of number of frequent patterns between SARS-CoV-1 and SARS-CoV-2



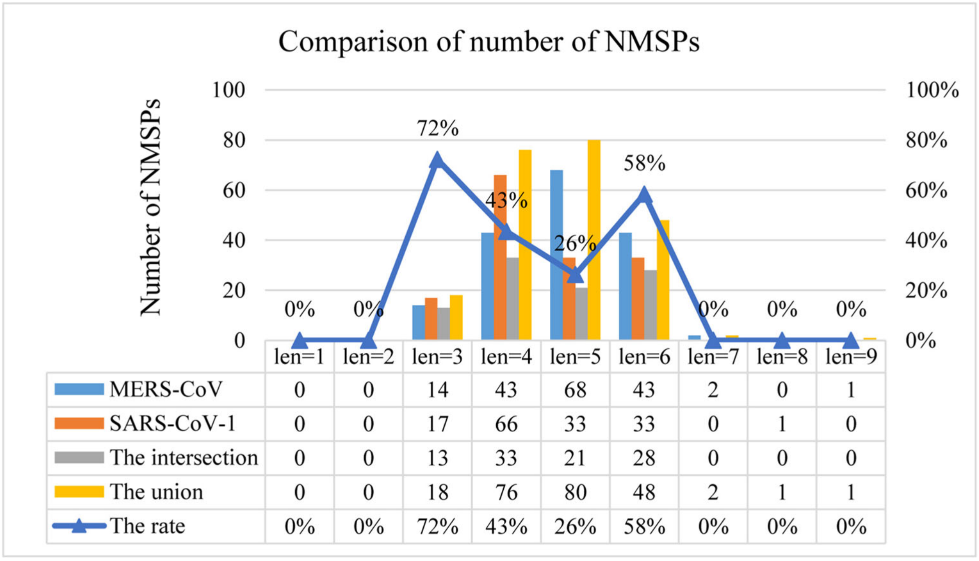
	len=1	len=2	len=3	len=4	len=5	len=6	len=7	len=8	len=9
SARS-CoV-1	0	0	17	66	33	33	0	1	0
SARS-CoV-2	0	1	16	42	41	44	16	0	1
The intersection	0	0	12	41	25	15	0	0	0
The union	0	1	21	67	49	62	16	1	1
The rate	0%	0%	57%	61%	51%	24%	0%	0%	0%

Fig. 22 Comparison of number of NMSPs between SARS-CoV-1 and SARS-CoV-2



	len=1	len=2	len=3	len=4	len=5	len=6	len=7	len=8	len=9
MERS-CoV	4	16	56	97	92	46	3	1	1
SARS-CoV-1	4	16	56	97	59	34	1	1	0
The intersection	4	16	56	87	56	21	1	1	0
The union	4	16	56	117	95	49	3	1	1
The rate	100%	100%	100%	74%	59%	43%	33%	100%	0%

Fig. 23 Comparison of number of frequent patterns between MERS-CoV and SARS-CoV-1



	len=1	len=2	len=3	len=4	len=5	len=6	len=7	len=8	len=9
MERS-CoV	0	0	14	43	68	43	2	0	1
SARS-CoV-1	0	0	17	66	33	33	0	1	0
The intersection	0	0	13	33	21	28	0	0	0
The union	0	0	18	76	80	48	2	1	1
The rate	0%	0%	72%	43%	26%	58%	0%	0%	0%

Fig. 24 Comparison of number of NMSPs between MERS-CoV and SARS-CoV-1

respectively. Among them, 95 patterns are the same. Therefore, the intersection and union are 95 and 226, respectively. Thus, the total similarity between MERS-CoV and SARS-CoV-1 is about 42%. Hence, the similarity between MERS-CoV and SARS-CoV-1 is higher than that between MERS-CoV and SARS-CoV-2.

6 Conclusion

This paper studies NMSP mining. As a pattern compression technology, without changing mining parameters, NMSP mining can compress the pattern set and reduce redundant patterns. To solve the problem, this paper proposes the NetNMSP algorithm which employs the backtracking strategy to calculate the support, the pattern join strategy to generate the candidate patterns, and the screening method to find NMSPs. To calculate the support, we propose the Netback algorithm which adopts the backtracking method to find an occurrence with no need to find and prune invalid nodes in the Nettree. Since NMSP mining satisfies the Apriori property, NetNMSP adopts the pattern join strategy to generate the candidate patterns. Meanwhile, NetNMSP employs the screening method to find NMSPs to improve the mining efficiency. Experiments on DNA and protein sequence datasets verify that NetNMSP can exactly mine all NMSPs in sequence datasets. Moreover, it shows that not only does NetNMSP outperform other competitive algorithms, but also NMSP mining has better compression performance than closed sequential pattern mining. Experiments on sales datasets validate that our algorithm guarantees the best scalability on large scale datasets. More importantly, we mine NMSPs and frequent patterns in SARS-CoV-1, SARS-CoV-2 and MERS-CoV. The results show that NMSP mining is easier to find the differences between the virus sequences.

Acknowledgements This work was partly supported by National Natural Science Foundation of China (Nos. 61976240, 52077056, 917446209), National Key Research and Development Program of China (2016YFB1000901), and Natural Science Foundation of Hebei Province, China (Nos. F2020202013, E2020202033).

References

1. Gan W, Lin JC-W, Fournier-Viger P, Chao H-C, Yu SP (2019) A survey of parallel sequential pattern mining. ACM Trans Knowl Discov Data 13(3):25
2. Fournier-Viger P, Gomariz A, Gueniche T, Soltani A, Wu CW, Tseng VS (2014) SPMF: A Java open-source pattern mining library. J Mach Learn Res 15(1):3389–3393
3. Qiang J, Qian Z, Li Y, Yuan Y, Wu X (2020) Short text topic modeling techniques, Applications, and Performance: A Survey. IEEE Transactions on Knowledge and Data Engineering (TKDE). https://doi.org/10.1109/TKDE.2020.2992485
4. Liu D, Wu Y, Jiang H (2016) FP-ELM: An online sequential learning algorithm for dealing with concept drift. Neurocomputing 207:322–334
5. Wu M, Wu X (2019) On big wisdom. Knowl Inf Syst 58(1):1–8
6. Wang T, Duan L, Dong G, Bao Z (2020) Efficient mining of outlying sequence patterns for analyzing outlierness of sequence data. ACM Trans Knowl Discov Data 14(5):62
7. Truong T, Duong H, Le B, Fournier-Viger P (2019) FMAxclo-HUSM: An efficient algorithm for mining frequent closed and maximal high utility sequences. Eng Appl Artif Intell 85:1–20
8. Lee G, Yun U, Ryu KH (2014) Sliding window based weighted maximal frequent pattern mining over data streams. Expert Syst Appl 41(2):694–708
9. Vo B, Pham S, Le T, Deng Z-H (2017) A novel approach for mining maximal frequent patterns. Expert Syst Appl 73:178–186
10. Wu Y, Luo L, Li Y, Guo L, Fournier-Viger P, Zhu X, Wu X (2021) NTP-Miner: Nonoverlapping three-way sequential pattern mining. ACM Transactions on Knowledge Discovery from Data. https://doi.org/10.1145/3480245
11. Wu Y, Wang X, Li Y, Guo L, Li Z, Zhang J, Wu X (2021) OWSP-Miner: Self-adaptive one-off weak-gap strong pattern mining. ACM Transactions on Management Information Systems. https://doi.org/10.1145/3476247
12. Cheng S, Wu Y, Li Y, Yao F, Min F (2021) TWD-SFNN: Three-Way decisions with a single hidden layer feedforward neural network. Inf Sci 579:15–32
13. Zhang Z, Min F, Chen G, Shen S, Wen Z, Zhou X (2021) Tri-partition state alphabet-based sequential pattern for multivariate time series. Cognitive Computation. https://doi.org/10.1007/s12559-021-09871-4
14. Dong X, Gong Y, Cao L (2020) e-RNSP: An efficient method for mining repetition negative sequential patterns. IEEE Trans Cybern 50:2084–2096
15. Dong X, Qiu P, Lu J, Cao L, Xu T (2019) Mining top-k useful negative sequential patterns via learning. IEEE Trans Neural Netw Learn Syst 30(9):2764–2778
16. Wu Y, Wang L, Ren J, Ding W, Wu X (2014) Mining sequential patterns with periodic wildcard gaps. Appl Intell 41(1):99–116
17. Fournier-Viger P, Yang P, Kiran RU, Ventura S, Luna JM (2021) Mining local periodic patterns in a discrete sequence. Inf Sci 544:519–548
18. Wang L, Bao X, Zhou L (2018) Redundancy reduction for prevalent co-location patterns. IEEE Trans Knowl Data Eng 30(1):142–155
19. Wu Y, Wang Y, Li Y, Zhu X, Wu X (2021) Top-k self-adaptive contrast sequential pattern mining. IEEE Transactions on Cybernetics. https://doi.org/10.1109/TCYB.2021.3082114
20. Duan L, Tang G, Pei J, Bailey J, Dong G, Nguyen V, Campbell A, Tang C (2016) Efficient discovery of contrast subspaces for object explanation and characterization. Knowl Inf Syst 47(1):99–129
21. Wang T, Duan L, Dong G, Bao Z (2020) Efficient mining of outlying sequence patterns for analyzing outlierness of sequence data. ACM Trans Knowl Discov Data 14(5):62
22. He Z, Zhang S, Wu J (2019) Significance-based discriminative sequential pattern mining. Expert Syst Appl 122:54–64
23. Wu Y, Zhu C, Li Y, Guo L, Wu X (2020) NetNCSP: Nonoverlapping closed sequential pattern mining. Knowl-Based Syst 196(105812)
24. Yun U, Nam H, Kim J, Kim H, Baek Y, Lee J, Yoon E, Truong TC, Vo B, Pedrycz W (2020) Efficient transaction deleting approach of pre-large based high utility pattern mining in dynamic databases. Fut Gener Comput Syst 103:58–78
25. Choi H-J, Park CH (2019) Emerging topic detection in twitter stream based on high utility pattern mining. Expert Syst Appl 115:27–36

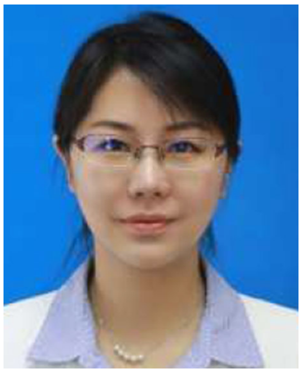
26. Lin JC-W, Pirouz M, Djenouri Y, Cheng C-F, Ahmed U (2020) Incrementally updating the high average-utility patterns with pre-large concept. Appl Intell 50(11):3788–3807
27. Wu Y, Geng M, Li Y, Guo L, Li Z, Fournier-Viger P, Zhu X, Wu X (2021) HANP-Miner: High average utility nonoverlapping sequential pattern mining. Knowl-Based Syst 229(107361)
28. Gan W, Lin JC-W, Fournier-Viger P, Chao H-C, Yu P. S. (2020) HUOPM: High-Utility occupancy pattern mining. IEEE Trans Cybern 50(3):1195–1208
29. Sumalatha S, Subramanyam R (2020) Distributed mining of high utility time interval sequential patterns using mapreduce approach. Expert Syst Appl 141(112967)
30. Wu Y, Tong Y, Zhu X, Wu X (2018) NOSEP: Nonoverlapping Sequence pattern mining with gap constraints. IEEE Trans Cybern 48(10):2809–2822
31. Zhang M, Kao B, Cheung DW, Yip KY (2007) Mining periodic patterns with gap requirement from sequences. ACM Trans Knowl Discov Data 1(2):7
32. Wang Y, Wu Y, Li Y, Yao F, Fournier-Viger P, Wu X (2021) Self-adaptive nonoverlapping sequential pattern mining. Applied Intelligence. https://doi.org/10.1007/s10489-021-02763-y
33. Wu Y, Fu S, Jiang H, Wu X (2015) Strict approximate pattern matching with general gaps. Appl Intell 42(3):566–580
34. Shi Q, Shan J, Yan W, Wu Y, Wu X (2020) NetNPG: Nonoverlapping pattern matching with general gap constraints. Appl Intell 50(6):1832–1845
35. Liu H, Wang L, Liu Z, Zhao P, Wu X (2018) Efficient pattern matching with periodical wildcards in uncertain sequences. Intell Data Anal 22:829–842
36. Tan C-D, Min F, Wang M, Zhang H-R, Zhang Z-H (2016) Discovering patterns with weak-wildcard gaps. IEEE Access 4:4922–4932
37. Miao S, Vespier U, Cachucho R, Meng M, Knobbe A (2016) Predefined pattern detection in large time series. Inf Sci 329:950–964
38. Wei L, Xing P, Shi G, Ji Z-L, Zou Q (2017) Fast prediction of protein methylation sites using a sequence-based feature selection technique. IEEE/ACM Trans Comput Biol Bioinform 16(4):1264–1273
39. Wu Y, Wang Y, Liu J, Yu M, Liu J, Li Y (2019) Mining distinguishing subsequence patterns with nonoverlapping condition. Cluster Comput 22(3):5905–5917
40. Ding B, Lo D, Han J, Khoo S-C (2009) Efficient mining of closed repetitive gapped subsequences from a sequence database. IEEE 25th International Conference on Data Engineering, pp 1024–1035
41. Wu Y, Li S, Liu J, Guo L, Wu X (2018) NETASPNO: Approximate strict pattern matching under nonoverlapping condition. IEEE Access 6:24350–24361
42. Wu Y, Shen C, Jiang H, Wu X (2017) Strict pattern matching under non-overlapping condition. Sci China Inf Sci 60(1):012101
43. Lin JC-W, Ahmed U, Srivastava G, Wu JM-T, Hong T-P, Djenouri Y (2021) Linguistic frequent pattern mining using a compressed structure. Appl Intell 51(7):4806–4823
44. Jiang H, Chen X, He T, Chen Z, Li X (2018) Fuzzy clustering of crowdsourced test reports for apps. ACM Trans Internet Technol 18(2):1–28
45. Ghosh S, Feng M, Nguyen H, Li J (2016) Hypotension risk prediction via sequential contrast patterns of ICU blood pressure. IEEE J Biomed Health Inf 20(5):1416–1426
46. Ghosh S, Li J, Cao L, Ramamohanarao K (2017) Septic shock prediction for ICU patients via coupled HMM walking on sequential contrast patterns. J Biomed Inform 66:19–31
47. Noor S, Guo Y, Shah SHH, Fournier-Viger P, Nawaz MS (2020) Analysis of public reactions to the novel Coronavirus (COVID-19) outbreak on Twitter. Kybernetes 50(5):1633–1653
48. Gan W, Lin JC-W, Zhang J, Fournier-Viger P, Chao H-C, Tseng VS, Yu PS (2021) A survey of utility-oriented pattern mining. IEEE Trans Knowl Data Eng 33(4):1306–1327
49. Fournier-Viger P, Li J, Lin JC-W, Chi TT, Kiran RU (2020) Mining cost-effective patterns in event logs. Knowl-Based Syst 191(105241)
50. Karim MR, Cochez M, Beyan OD, Ahmed CF, Decker S (2018) Mining maximal frequent patterns in transactional databases and dynamic data streams: a spark-based approach. Inf Sci 432:278–300
51. Min F, Zhang Z-H, Zhai W-J, Shen R-P (2020) Frequent pattern discovery with tri-partition alphabets. Inf Sci 507:715–732
52. Gan W, Lin JC-W, Zhang J, Chao H-C, Fujita H, Yu P. S. (2020) ProUM: Projection-based utility mining on sequence data. Inf Sci 513:222–240
53. Song W, Jiang B, Qiao Y (2018) Mining multi-relational high utility itemsets from star schemas. Intell Data Anal 22(1):143–165
54. Nam H, Yun U, Yoon E, Lin JC-W (2020) Efficient approach of recent high utility stream pattern mining with indexed list structure and pruning strategy considering arrival times of transactions. Inf Sci 529:1–27
55. Kim H, Yun U, Baek Y, Kim J, Vo B, Yoon E, Fujita H (2021) Efficient list based mining of high average utility patterns with maximum average pruning strategies. Inf Sci 543(8):85–105
56. Yun U, Kim D, Yoon E, Fujita H (2018) Damped window based high average utility pattern mining over data streams. Knowl-Based Syst 144(15):188–205
57. Song W, Liu Y, Li J (2014) Mining high utility itemsets by dynamically pruning the tree structure. Appl Intell 40(1):29–43
58. Ghosh S, Li J, Cao L, Ramamohanarao K (2017) Septic shock prediction for ICU patients via coupled HMM walking on sequential contrast patterns. J Biomed Inform 66:19–31
59. Wu Y, Wang Y, Li Y, Zhu X, Wu X (2021) Self-adaptive nonoverlapping contrast sequential pattern mining. IEEE Transactions on Cybernetics. https://doi.org/10.1109/TCYB.2021.3082114
60. Nam H, Yun U, Yoon E, Lin JC-W (2020) Efficient approach for incremental weighted erasable pattern mining with list structure. Expert Syst Appl 143(113087)
61. Piri S, Delen D, Liu T, Paiva W (2018) Development of a new metric to identify rare patterns in association analysis: The case of analyzing diabetes complications. Expert Syst Appl 94:112–125
62. Le B, Duong H, Truong T, Fournier-Viger P (2017) FCLoSM, FGenSM: Two efficient algorithms for mining frequent closed and generator sequences using the local pruning strategy. Knowl Inf Syst 53:71–107
63. Yun U, Lee G, Ryu KH (2014) Mining maximal frequent patterns by considering weight conditions over data streams. Knowl-Based Syst 55:49–65
64. Lee G, Yun U, Ryang H, Kim D (2016) Approximate maximal frequent pattern mining with weight conditions and error tolerance. Int J Pattern Recogn Artif Intell 30(6):1650012
65. Li C, Yang Q, Wang J, Li M (2012) Efficient mining of gap-constrained subsequences and its various applications. ACM Trans Knowl Discov Data 6(1):2
66. Wu Y, Fan J, Li Y, Guo L, Wu X (2020) NetDAP: (delta, gamma) -Approximate pattern matching with length constraints. Appl Intell 50(11):4094–4116
67. Lam HT, Morchen F, Fradkin D, Calders T (2014) Mining compressing sequential patterns. Stat Anal Data Mining: ASA Data Sci J 7(1):34–52
68. Wu Y, Lei R, Li Y, Guo L, Wu X (2021) HAOP-Miner: Self-adaptive high-average utility one-off sequential pattern mining. Expert Syst Appl 184(115449)
69. Xie F, Wu X, Zhu X (2017) Efficient sequential pattern mining with wildcards for keyphrase extraction. Knowl-Based Syst 115:27–39

70. Wu Y, Tang Z, Jiang H, Wu X (2016) Approximate pattern matching with gap constraints. J Inf Sci 42(5):639–658
71. He R, Dobie F, Ballantine M, Leeson A, Li Y, Bastien N, Cutts T, Andonov A, Cao J, Booth TF, Plummer FA, Tyler S, Baker L, Li X (2004) Analysis of multimerization of the SARS coronavirus nucleocapsid protein. Biochem Biophys Res Commun 316(2):476–483
72. Wu F, Zhao S, Yu B, Chen YM, Wang W, Song ZG, Hu Y, Tao ZW, Tian JH, Pei YY, Yuan ML, Zhang YL, Dai FH, Liu Y, Wang QM, Zheng JJ, Xu L, Holmes EC, Zhang YZ (2020) A new coronavirus associated with human respiratory disease in China. Nature 579(7798):265–269
73. Zaki AM, van Boheemen S, Bestebroer TM, Osterhaus AD, Fouchier RA (2012) Isolation of a novel coronavirus from a man with pneumonia in Saudi Arabia. N Engl J Med 367(19):1814–1820
74. Nawaz MS, Fournier-Viger P, Shojaee A, Fujita H (2021) Using artificial intelligence techniques for COVID-19 genome analysis. Appl Intell 51(5):3086–3103

Yan Li received the Ph.D. degree in Management Science and Engineering from Tianjin University, Tianjin, China. She is an Associate Professor with the Hebei University of Technology, Tianjin. Her current research interests include data mining and supply chain management.

Shuai Zhang received the master degree candidate in School of Artificial Intelligence from the Hebei University of Technology, Tianjin, China. His current research interest includes data mining.

Lei Guo received the Ph.D. degree in Theory and New Technology of Electrical Engineering from the Hebei University of Technology, Tianjin, China. He is currently a Ph.D. Supervisor and a Professor with the Hebei University of Technology. His current research interests include Bioengineering, neural engineering, and pattern recognition.



Jing Liu received the Ph.D. degree in xxx from the yyyy, Tianjin, China. She was a post doctor from the University of Iowa. She is currently a Professor with the Hebei University of Technology. Her current research interests include pattern mining and industrial artificial intelligence.

Youxi Wu received the Ph.D. degree in Theory and New Technology of Electrical Engineering from the Hebei University of Technology, Tianjin, China. He is currently a Ph.D. Supervisor and a Professor with the Hebei University of Technology. He has published more than 30 research papers in some journals, such as IEEE TCYB, ACM TKDD, ACM TMIS, SCIS, INS, JCST, KBS, EWSA, JIS, Neurocomputing, and APIN. He is a senior member of CCF and a member of IEEE. His current research interests include data mining and machine learning.

Xindong Wu received the Ph.D. degree from the University of Edinburgh, Edinburgh, U.K. He is a Ph.D. Supervisor, a Professor, and a Yangtze River Scholar with the Hefei University of Technology, Hefei, China. His current research interests include data mining, big data analytics, knowledge based systems, and Web information exploration. Dr. Wu is the Steering Committee Chair of the IEEE International Conference on Data Mining and the Editor-in-Chief of Knowledge and Information Systems. He is a Fellow of the American Association for the Advancement of Science and IEEE fellow.

Affiliations

Yan Li[1] · **Shuai Zhang**[2] · **Lei Guo**[3] · **Jing Liu**[2] · **Youxi Wu**[2,4] · **Xindong Wu**[5,6]

Yan Li
lywuc@163.com

Shuai Zhang
1090480465@qq.com

Lei Guo
guoshengrui@163.com

Jing Liu
liujing@scse.hebut.edu.cn

Xindong Wu
xwu@hfut.edu.cn

[1] School of Economics and Management, Hebei University of Technology, Tianjin 300401, China

[2] School of Artificial Intelligence, Hebei University of Technology, Tianjin 300401, China

[3] State Key Laboratory of Reliability and Intelligence of Electrical Equipment, Hebei University of Technology, Tianjin 300401, China

[4] Hebei Key Laboratory of Big Data Computing, Tianjin 300401, China

[5] Research Institute of Big Knowledge, Hefei University of Technology, Hefei 230009, China

[6] Mininglamp Software Systems, Beijing 100084, China