Knowledge-Based Systems 196 (2020) 105812

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

NetNCSP: Nonoverlapping closed sequential pattern mining

Youxi Wu [a,b,d,*], Changrui Zhu [a], Yan Li [c], Lei Guo [b], Xindong Wu [e,f]

[a] School of Artificial Intelligence, Hebei University of Technology, Tianjin 300401, China
[b] State Key Laboratory of Reliability and Intelligence of Electrical Equipment, Hebei University of Technology, Tianjin 300401, China
[c] School of Economics and Management, Hebei University of Technology, Tianjin 300401, China
[d] Hebei Key Laboratory of Big Data Computing, Tianjin 300401, China
[e] Mininglamp Academy of Sciences, Mininglamp Technology, Beijing 100084, China
[f] Key Laboratory of Knowledge Engineering with Big Data (Hefei University of Technology), Ministry of Education, Hefei 230009, China

ARTICLE INFO

Article history:
Received 5 January 2020
Received in revised form 22 March 2020
Accepted 22 March 2020
Available online 31 March 2020

Keywords:
Sequential pattern mining
Closed pattern mining
Nonoverlapping sequence pattern
Periodic wildcard gaps
Nettree
COVID-19

ABSTRACT

Sequential pattern mining (SPM) has been applied in many fields. However, traditional SPM neglects the pattern repetition in sequence. To solve this problem, gap constraint SPM was proposed and can avoid finding too many useless patterns. Nonoverlapping SPM, as a branch of gap constraint SPM, means that any two occurrences cannot use the same sequence letter in the same position as the occurrences. Nonoverlapping SPM can make a balance between efficiency and completeness. The frequent patterns discovered by existing methods normally contain redundant patterns. To reduce redundant patterns and improve the mining performance, this paper adopts the closed pattern mining strategy and proposes a complete algorithm, named Nettree for Nonoverlapping Closed Sequential Pattern (NetNCSP) based on the Nettree structure. NetNCSP is equipped with two key steps, support calculation and closeness determination. A backtracking strategy is employed to calculate the nonoverlapping support of a pattern on the corresponding Nettree, which reduces the time complexity. This paper also proposes three kinds of pruning strategies, inheriting, predicting, and determining. These pruning strategies are able to find the redundant patterns effectively since the strategies can predict the frequency and closeness of the patterns before the generation of the candidate patterns. Experimental results show that NetNCSP is not only more efficient but can also discover more closed patterns with good compressibility. Furtherly, in biological experiments NetNCSP mines the closed patterns in SARS-CoV-2 and SARS viruses. The results show that the two viruses are of similar pattern composition with different combinations.

1. Introduction

Sequential pattern mining (SPM) refers to discover the subsequences (also known as patterns) that satisfy the threshold from given sequences [1–3]. It has been widely applied in various fields, such as big data mining [4,5], big data intelligence [6], e-commerce shopping analysis [7], biological sequence analysis [8], and event analysis [9]. To handle some specific issues, many methods have been proposed, such as negative SPM [10,11], maximal frequent pattern mining [12,13], three-way pattern mining [14,15], closed SPM [16,17], gap constraint SPM [18,19].

Gap constraint SPM is an important branch of traditional SPM. In traditional SPM, a sequence database is a set of sequences, each sequence is a list of elements, and each element is a set of items. For example, <a(abc)(ac)c(cd)> is a sequence in traditional SPM, since (abc) is an element in the sequence. In gap constraint SPM, a sequence database can be a sequence, and each sequence is a list of items. For example, <aabcacccd> is a sequence in gap constraint SPM. More importantly, traditional SPM does not calculate the number of occurrences of a pattern in a sequence, while gap constraint SPM does. For example, apparently, pattern "ac" occurs in sequence <a(abc)(ac)c(cd)> more than once. But traditional SPM neglects the repetition, which leads to the loss of support information. For example, in long-length sequences such as DNA, virus, and consumption records, the repetitive occurrences indicate the frequency of patterns, which reflect information discrimination between patterns, and are of high research value.

Gap constraint SPM (or gap constraint sequence pattern mining) can avoid finding many useless patterns by setting gap constraints, while repetitive SPM cannot [20,21] since it does not set gap constraint. Gap constraint SPM is also different from frequent substring mining [22] and n-gram text mining [23], since the latter mine the pattern without gaps. One key issue of gap constraint SPM is to calculate the support (the number of occurrences) of a

* Corresponding author.
E-mail address: wuc567@163.com (Y. Wu).

https://doi.org/10.1016/j.knosys.2020.105812
0950-7051/

		1	2	3	4	5	6	7	8	
S_1	=	A	A	T	C	A	T	C	A	
		A		T	C	A	·	·	·	The first occurrence
		·	A	T	C	A	·	·	·	The second occurrence
		·	·	·	·	A	T	C	A	The third occurrence

Fig. 1. The occurrences of pattern P in sequence S_1.

pattern in a sequence, which is pattern matching problem [24–26]. Thus, gap constraint is also called a wildcard gap or flexible wildcards [27,28] in pattern matching fields. If a pattern has many gaps that are the same, the pattern is called a pattern with periodic gaps [29], described as $P = p_1[a, b]p_2 \cdots p_{m-1}[a, b]p_m$, where a and b ($0 \le a \le b$) are the minimum and maximum gap constraints, respectively, and m indicates the pattern length [30]. The size of gap constraints can be flexibly set by users, which leads to various applications, such as correlation analysis between DNA and diseases based on gap constraints [29]. Li et al. [31] proposed an effective method to mine the patterns with gap constraints that can be used for feature extraction for sequence classifications [32]. However, introducing gap constraints not only makes the mining method more flexible but also makes the results more complex because the number of patterns increases exponentially as the pattern length increases. To solve this problem, the nonoverlapping condition [33] was proposed, which allows the same sequence letter to match and rematch pattern letters at different positions. Introducing the nonoverlapping condition not only reduces the number of occurrences but also makes the unique patterns richer. Our previous studies have proved that the nonoverlapping SPM is a complete mining method which satisfies the Apriori property [34,35]. Example 1 is used to explain the periodic gap pattern, gap constraint mining, and nonoverlapping support.

Example 1. Given a sequence database SDB={S_1=AATCATCA, S_2=AATGACTACTCAA, S_3=ATCAGATCAG}. Pattern P=A[0, 2]T[0, 2]C[0, 2]A is a periodic gapped pattern. To make it easier to describe, an occurrence will be described in the form of a sequence landmark. For example, the first occurrence of P in S_1 is $s_1s_3s_4s_5$, which can be written as <1,3,4,5>. In this way, the other 2 occurrences of P in S_1 are <2,3,4,5> and <5,6,7,8>. The results are shown in Fig. 1.

Similarly, there are four occurrences of P in both S_2 and S_3. If we ignore the repetition, the support of P_1 in SDB is 3. If we take the repetition into consideration, the support is 3+4+4=11. The latter method considers occurrences in detail. However, this method also encounters some problems. For example, there are three occurrences for pattern Q= A[0, 2]T in sequence S_4= AAATCCC, but there are nine occurrences for its super-pattern Q'= A[0, 2]T[0, 2]C. This example shows that the number of patterns increases exponentially with increasing pattern length, which does not satisfy the Apriori property [19]. To solve the above problem, the nonoverlapping condition was proposed [33]. With the nonoverlapping condition, any two occurrences cannot use the same sequence letter in the same position. For example, <1,3,4,5> and <5,6,7,8> are two nonoverlapping occurrences for pattern P in sequence S_1, since sequence letter s_5 is matched twice with p_4 and p_1, respectively. However, <1,3,4,5> and <2,3,4,5> are two overlapping occurrences, since sequence letters s_3, s_4, and s_5 are reused by p_2, p_3, and p_4, respectively.

Our previous work proposed an effective algorithm NOSEP and reported that the nonoverlapping SPM has better performance than other state-of-the-art gap constraint SPM methods in finding useful patterns in biology sequences and avoiding under-expression and over-expression in time series [34]. However, all frequent patterns discovered by NOSEP can be furtherly compressed. An illustrative example is shown as follows.

Example 2. When $minsup$=2, gap=[0, 2], with the nonoverlapping condition, there are 16 frequent patterns in the sequence S_1= AATCATCA, which are {"A", "AA", "AT", "AC", "TC", "CA", "AAA", "AAC", "AAT", "ATA", "ATC", "ACA", "AA", "TC", "AATA", "AACA", "ATCA"}. The supports of these patterns are sup("A")=4, sup("AA")=3, and 2 for the remaining patterns. Patterns "AT", "TC", "AAT", and "ATC" are unclosed patterns, since these patterns are sub-patterns of pattern "AATC" and their supports are all 2. In this way, 16 frequent patterns can be compressed into 6 closed patterns, "A", "AA", "AATC", "AATA", "AACA", and "ATCA". This example shows that closed patterns can effectively compress frequent patterns without losing support information.

To reduce redundant patterns and improve the mining speed, this paper adopts the closed pattern mining strategy to obtain lossless compression of frequent patterns. The contributions of this paper are as follows:

1. The problem of nonoverlapping periodic gapped closed SPM is addressed, and a complete algorithm NetNCSP (Nettree for Nonoverlapping Closed Sequential Pattern) is proposed.
2. To calculate the support, NetNCSP employs the backtracking strategy to match the pattern in the Nettree structure, which reduces the time complexity. More importantly, NetNCSP adopts three pruning strategies to find closed patterns. We show that NetNCSP is a complete algorithm that satisfies the Apriori property.
3. A large number of comparative experiments show that NetNCSP is not only more efficient, but also possesses remarkable pattern compressibility.

The rest of this paper is organized as follows. Section 2 introduces the related work. Section 3 defines the problem. Section 4 proposes the NetNCSP algorithm, and demonstrates the completeness, complexities, and Apriori property. Section 5 makes a comparative experimental analysis. Section 6 draws the conclusion of this paper.

2. Related work

Agrawal et al. [36] proposed SPM. Based on this research, many achievements have been made, such as high utility mining [37], contrast SPM [38,39], and closed SPM [40,41]. Closed SPM can effectively compress the frequent patterns [42,43]. For example, assuming that the supports of patterns "A", "AT", and "ATC" are equal, then patterns "A" and "AT" are called redundant patterns. Hence, patterns "A", "AT", "ATC" can be compressed into "ATC" without losing support information. Besides closed pattern mining, there are other methods to achieve pattern compression, such as generator mining [44] and maximal pattern mining [45]. Generator mining aims to find the set of patterns with minimal length, while closed pattern mining focuses on finding the set of patterns with maximal length. Maximal pattern mining finds the set of patterns whose super-patterns are infrequent. Closed SPM has become a research hotspot because of its impressive compression performance [46,47] and has been widely used in many essential fields, such as recommendation systems [48], clustering analysis [49–51], genetic engineering [52], disease diagnosis [53], and software engineering [54,55]. However, these studies ignored the repetitions that may contain more relevant information in long sequences. Noticing this disadvantage, Ding

et al. [33] proposed the CloGSgrow algorithm, which studied the repetitive occurrences of patterns. CloGSgrow calculates the occurrences of the super-patterns based on those of the sub-patterns and the results show that the repetitive occurrence pattern is compressible, which is of high research value for long sequences. Li et al. [31] added periodic gap constraints and the experiments show that introducing gap constraints improves the mining results.

Another important branch of SPM is gap constraint SPM. This method aims to discover subsequences from sequences that satisfy the gap constraints and support threshold. Existing studies are based on three types of conditions, no-condition [29,56], one-off condition [8,57], and nonoverlapping condition [33,34,58]. The no-condition allows sequence letters to be reused by patterns for an unlimited time. Zhang et al. [29] first proposed SPM under no-condition in DNA sequences. However, pattern mining under no-condition does not satisfy the Apriori property, which means it is necessary to expand the search space to mine all frequent patterns [19]. The one-off condition allows the sequence letters to be matched no more than once [59]. In Example 1, there is only one occurrence $<1,3,4,5>$ of P in S_1 with the one-off condition. Pattern mining under the one-off condition was applied to biological sequence mining [8], which is an NP-Hard problem, since its computational complexity is the same as that of an iterative shuffle problem [60]. Therefore, heuristic strategies are applied to calculate the pattern support. Hence, pattern mining under the one-off condition belongs to approximate mining. The nonoverlapping condition allows the sequence letters to be rematched by the different pattern letters and to be matched no more than once by the same pattern letter. Although the nonoverlapping condition is more complex than the other two, our previous studies have demonstrated that the pattern mining under the nonoverlapping condition is not only a complete mining method with the Apriori property but can also discover more valuable patterns than the other two conditions [34,35]. Consequently, pattern mining under the nonoverlapping condition outperforms those of the no-condition and one-off condition. A comparison of the related research work is shown in Table 1.

As can be seen from Table 1, Reference [34] is the closest to this paper. The differences between Reference [34] and this paper are as follows. To obtain the maximum pattern support, NETGAP prunes the invalid nodes after obtaining a nonoverlapping occurrence [34]. Hence, the time complexity of NETGAP is $O(m \times m \times n \times W)$, where m, n, and W are the length of pattern and sequence, and the maximum gap, respectively. However, in this paper, we propose the BackTr algorithm which employs a backtracking strategy to calculate the pattern support without pruning the invalid nodes, which will reduce the time complexity to $O(m \times n \times W)$. In addition, Reference [34] focused on mining the frequent patterns, while this paper mines the closed patterns and adopts three pruning strategies, inheriting, predicting, and determining to predict the frequency and closeness of the patterns.

3. Problem definition

Definition 1. A sequence with length n can be described as $S=s_1 \cdots s_i \cdots s_n$, where $s_i (1 \le i \le n) \in \Sigma$, Σ denotes the set of items, and $|\Sigma|$ indicates the size. A periodic gap constraint pattern P with length m can be written as $P=p_1[a,b]p_2 \cdots [a,b]p_j \cdots [a,b]p_m$, where a and b are integers ($0 \le a \le b$) that indicate the minimum and maximum gaps, respectively.

Definition 2. $L=< l_1, l_2 \cdots l_m >$ is an occurrence of pattern P in sequence S, if and only if $1 \le l_1 < \cdots < l_m \le n$ and $a \le l_{j+1}-l_j-1 \le b$, where $s_{l_j} = p_j$ ($1 \le j \le m$ and $1 \le l_j \le n$). Suppose there is another occurrence $L' =< l'_1, l'_2 \cdots l'_m >$. L and L' are two nonoverlapping occurrences if and only if $\forall 1 \le j \le m$ and $l_j \ne l'_j$. The nonoverlapping support of pattern P in sequence S is represented by $sup(P, S)$.

Definition 3. If $sup(P, S)$ is no less than support threshold $minsup$, pattern P is called a frequent pattern.

Example 3. In Example 1, Σ={A, T, C, G}, $|\Sigma|$=4, $minsup$=2, and len=[1,7], the nonoverlapping occurrences of P in S_1 are $< 1, 3, 4, 5 >$ and $< 6, 7, 8, 9 >$. We can know that $sup(P, S_1)=2 \ge minsup$. Thus, if we mine the frequent pattern in S_1, P is a frequent pattern.

Definition 4. Suppose $L=< l_1, l_2 \cdots l_m >$ is an occurrence. If $minlen \le l_m - l_1 + 1 \le maxlen$, then L is an occurrence that satisfies the length constraints, where $minlen$ and $maxlen$ are the minimum and maximum length constraints, respectively.

Definition 5. Given pattern $P = p_1[a,b]p_2[a,b] \cdots [a,b]p_m$, and letters r and l. $Q = Pr = p_1[a,b]p_2[a,b] \cdots [a,b]p_m[a,b]r$ is defined as the right gap super-pattern of P, and P is the prefix sub-pattern of Q, i.e. prefix(Q)=P. Similarly, $R=lP=l[a,b]P_1[a,b]P_2[a,b] \cdots [a,b]p_m$ is defined as the left gap super-pattern of P and P is the suffix sub-pattern of R, i.e. suffix(R)=P. If prefix(Q)=suffix(R)=P, R and Q can be connected into a super-pattern T with length m+2, where $T = R \oplus Q = lPr$. This process of generating the super-pattern from sub-patterns is called pattern growth [64].

Example 4. Suppose pattern P=A[0, 2]T. Patterns R=G[0, 2]A[0, 2]T and Q= A[0, 2]T[0, 2]C are the left and right gap super-patterns of P, respectively. Therefore, patterns R and Q can be connected into super-pattern T with length 4, i.e. $T=R \oplus Q$=G[0, 2]A[0, 2]T[0, 2]C.

Definition 6. Suppose P is a frequent pattern. P is a closed pattern if there is no super-pattern P' of P which satisfies $sup(P')=sup(P)$; otherwise, P is an unclosed pattern (or a redundant pattern).

Example 5. In Example 1, P=A[0, 2]T[0, 2]C[0, 2]A. One of its super-patterns is P'=A[0, 2]T[0, 2]C[0, 2]A[0, 2]G. The nonoverlapping occurrences of P and P' in S_3 are {$<1,2,3,4>$, $<4,7,8,9>$} and {$<1,2,3,4,5>$, $<4,7,8,9,10>$}, respectively, i.e. $sup(P, S_1)=sup(P', S_1)=2$. Hence, pattern P is a redundant pattern.

4. Nonoverlapping closed SPM algorithm

Section 4.1 proposes the BackTr algorithm to calculate the pattern support. Section 4.2 introduces the principle of the pattern growth strategy to generate candidate patterns. Section 4.3 proposes three pruning strategies to determine closed patterns. We show the NetNCSP algorithm in Section 4.4.

4.1. Support calculating

Given a sequence and a pattern with gap constraints, all occurrences can be represented by a Nettree [65] which is an extended tree structure with multiple roots and parents. Since the nodes with the same label can appear on a Nettree for multiple times, n_j^i is used to represent node i in the j-th level. A path from a root to a leaf in the Nettree corresponds to an occurrence of the pattern in the sequence. The problem of calculating the support of pattern P in sequence S with the nonoverlapping condition means that all Nettree nodes cannot be reused in the same level [34]. The above properties make Nettree the most suitable for representing the nonoverlapping occurrences of a pattern. In our previous work [34], NETGAP was proposed employing Nettree, which is a complete method to calculate the nonoverlapping occurrence. However, the weakness of NETGAP is the lower efficiency since

Table 1
Related work.

Research	Pattern type	Type of condition	Pruning strategy	Mining type	Periodic gap constraint	Repetitions of pattern
Yan et al. [61]	Closed	Ignore	Other	Exact	No	Ignore
Wang et al. [62]	Closed	Ignore	Other	Exact	No	Ignore
Lam et al [63]	Compressed	On–off condition	Apriori	Approximate	No	Ignore
Wu et al [8]	Frequent	On–off condition	Apriori	Approximate	Yes	Capture
Li et al. [31]	Closed	No-condition	Other	Exact	Yes	Capture
Zhang et al. [29]	Frequent	No-condition	Apriori-like	Exact	Yes	Capture
Wu et al. [34]	Frequent	Nonoverlapping condition	Apriori	Exact	Yes	Capture
This paper	Closed	Nonoverlapping condition	Apriori, other	Exact	Yes	Capture

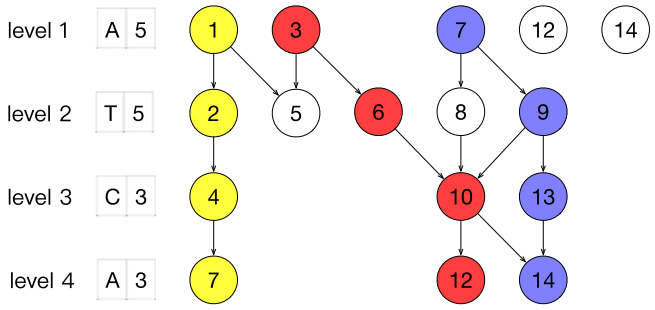


Fig. 2. A Nettree.

Algorithm 1 BackTr.

Input: candidate pattern P and sequence S
Output: support sup

```
Create a periodic gap Nettree according to P and S;
sup=0;
for each root do
    occ=Get a root–leaf path according to the leftmost child with backtracking strategy;
    sup++;
    Delete occ;
end for
return sup;
```

NETGAP must prune the invalid nodes after obtaining a nonoverlapping occurrence. Based on the above reasons, we propose the BackTr algorithm, which is of superior efficiency. Examples 6 and 7 will illustrate the principles of NETGAP and BackTr, respectively.

Example 6. Given pattern P=A[0,3]T[0,3]C[0,3]A and sequence S_1=ATACTTATTCGACA, a Nettree can be created as shown in Fig. 2. The first step of NETGAP is to prune the invalid nodes which are n_2^5, n_1^{12}, and n_1^{14}. Then NETGAP selects the first root node and finds a root–leaf path employing the leftmost child strategy. In Fig. 2, it is easy to obtain the first root–leaf path $< n_1^1, n_2^2, n_3^4, n_4^7 >$, marked in yellow. The corresponding occurrence is <1,2,4,7>. Then, NETGAP deletes nodes n_1^1, n_2^2, n_3^4, and n_4^7. After that, NETGAP finds the invalid nodes in the new Nettree. It is clear that there are no invalid nodes at that time. Then NETGAP obtains the second root–leaf path $< n_1^3, n_2^6, n_3^{10}, n_4^{12} >$, marked in red and its corresponding occurrence is <3,6,10,12>. After pruning nodes n_1^3, n_2^6, n_3^{10}, and n_4^{12}, NETGAP finds an invalid node n_2^8 and prunes it. Finally, NETGAP obtains the third occurrence <7,9,13,14>, which is marked in blue. Hence, NETGAP gets three nonoverlapping occurrences.

Example 7. In this example, we use the same pattern and sequence as in Example 6. BackTr does not need to find and prune invalid nodes n_2^5, n_1^{12}, and n_1^{14}, and gets the first occurrence <1,2,4,7>. After that, BackTr selects the second root n_1^3 and finds its first child node n_2^5, which has no child node. In that case, the algorithm backtracks to node n_1^3 and finds its second child which is node n_2^6. Thus, BackTr will get another occurrence <3,6,10,12>. Similarly, BackTr can find the third occurrence <7,9,13,14>. After that, there is no occurrence in the rest of the Nettree. Hence, BackTr also gets the same three nonoverlapping occurrences as NETGAP.

From Examples 6 and 7, it can be concluded that the two algorithms employ different methods to find the same nonoverlapping occurrences. However, NETGAP needs to find and prune the invalid nodes for three times, while BackTr does not need to prune these nodes, which will reduce the time complexity.

BackTr is given in Algorithm 1.

Theorem 1. *BackTr is complete.*

Proof. Our previous work showed that the complete algorithm should iteratively find the minimum occurrence [34]. BackTr iteratively selects the leftmost child from the minimum root to get the minimum occurrence. Hence, BackTr is complete.

Theorem 2. *In the worst case, the space and time complexities of BackTr are both $O(m \times n \times w)$, and the average space and time complexities are $O(m \times n \times w/r/r)$, where m, n, w, r are the pattern length, sequence length, $b-a+1$, and item number $|\Sigma|$, respectively.*

Proof. Obviously, each node in a Nettree will be accessed at most once. Hence, the time complexity of BackTr is consistent with the space complexity of a Nettree. A Nettree has m levels. Each level has a maximum of n nodes. Each node has a maximum of w children. Therefore, the space complexity of a Nettree is $O(m \times n \times w)$ in the worst case. Hence, the time complexity of a Nettree is also $O(m \times n \times w)$. On average, each level has n/r nodes and each node has w/r children. Hence, the average space and time complexities are $O(m \times n \times w/r/r)$.

As we know, the average time complexity of the NETGAP algorithm is $O(m \times n \times w/r/r)$ [34]. Hence, BackTr outperforms NETGAP.

4.2. Candidate pattern generating

Traditional candidate pattern generation methods include breadth-first and depth-first. In this paper, the pattern growth strategy can effectively reduce the generation of redundant patterns. An example is as follows.

Example 8. In Example 1, with *minsup*=2 and *gap*=[0,3], there are nine frequent patterns in S_3 with length 2: {"AA", "AT", "AC", "AG", "TC", "TA", "TG", "CA", "CG"}. With breadth-first or depth-first strategy, 9 × 4=36 candidate patterns with length 3 will be generated. On the other hand, since "TT" is not frequent, super-patterns "ATT" and "TTG" are also not frequent according to the Apriori property and can be pruned. According to the pattern

growth strategy, there are 14 candidate patterns: {"AAA", "AAT", "AAG", "AAC", "CAA", "CAT", "CAC", "CAG", "TAA", "TAT", "TAC", "TAG", "TCA", "TCG"}. This example illustrates that the pattern growth strategy outperforms the breadth-first and depth-first strategies.

4.3. Closed pattern determining

In this subsection, we propose three pruning strategies to find closed patterns.

Although BackTr can reduce the time complexity to calculate the support, it is also very complex. Therefore, we propose an inheriting strategy to predict the closeness of a pattern. The unclosed patterns will be pruned before support calculation using BackTr.

4.3.1. Inheriting

Definition 7. Given pattern $P=p_1p_2\cdots p_m$, and letters l and r. If there is a right gap super-pattern $Q=Pr$ which satisfies $sup(Q)=sup(P)$, then P is called a right unclosed pattern. In the same way, if there is a left gap super-pattern $R=lP$ and $sup(R)=sup(P)$, P is called a left unclosed pattern. Otherwise, P is called a right or left closed pattern.

Example 9. In Example 1, pattern P_1=T[0, 2]A has two nonoverlapping occurrences, $\langle 2, 4\rangle$ and $\langle 7, 9\rangle$ in S_3=ATCAGATCAG. By traversing the left gap of all occurrences, it can be found that there is a common letter "A". Therefore, there are two nonoverlapping occurrences for super-pattern P_1'=A[0, 2]T[0, 2]A in S_3, i.e. $sup(P_1', S_3)=sup(P_1, S_3)=2$. Hence, P_1 is a left unclosed pattern. Similarly, there exists a letter "G" in all occurrences of the right gaps of P_1, i.e. P_1''=T[0, 2]A[0, 2]G and $sup(P_1'', S_3)=sup(P_1, S_3)=2$. Hence, P_1 is a right unclosed pattern. However, there are two nonoverlapping occurrences for pattern P_2=A[0,3]G in S_3, i.e. $\langle 1, 5\rangle$ and $\langle 6, 10\rangle$. Yet, there is no common letter found in the left and right gaps of P_2. Hence, P_2 is a closed pattern.

Theorem 3. *If sub-pattern P is a left unclosed pattern in sequence S, then all its right super-patterns $Q = Pr$ are also left unclosed patterns. Therefore, pattern Q can be safely pruned. The same strategy can be applied to the right side.*

Proof. Suppose pattern P is a left unclosed pattern, which means there is a super-pattern $P'=lP$ whose support is the same as that of pattern P, i.e. $sup(P', S)=sup(P, S)$. We will show that pattern $Q=Pr$ has a super-pattern $Q'=lQ=lPr$ whose support is the same as that of pattern Q, i.e. $sup(Q', S)=sup(Q, S)$.

Suppose $< l_1', l_2'\cdots l_m', l_r >$ is an occurrence of pattern Q. $< l_1', l_2'\cdots l_m' >$ is an occurrence of pattern P. Since P is a left unclosed pattern, i.e. $sup(lP, S)= sup(P, S)$, we know that $< l_l, l_1', l_2'\cdots l_m' >$ is an occurrence of pattern lP. Therefore, $< l_l, l_1', l_2'\cdots l_m', l_r >$ is an occurrence of pattern lPr. Thus, $sup(Q', S)=sup(Q, S)$. Hence, pattern Q is also a left unclosed pattern.

It should be noticed that only the unclosed property can be inherited not the closed property.

Example 10. In S_2=AATGACTACTCAA, there are three nonoverlapping occurrences, <3>, <7>, and <10> for pattern "T" in S_2, i.e. $sup("T", S_2)=3$. Since there are also three nonoverlapping occurrences, $\langle 3, 5\rangle$, $\langle 7, 8\rangle$, and $\langle 10, 12\rangle$ for the right gap super-pattern "T[0, 2]A" of pattern "T", i.e. $sup("T[0, 2]A", S_2)=sup("T", S_2)=3$, pattern "T" is a right unclosed pattern. In addition, since "T" is a right unclosed pattern, then "A[0, 2]T" is also a right unclosed pattern according to Theorem 3. The verification is as follows. There are three nonoverlapping occurrences, $\langle 1, 3\rangle$, $\langle 5, 7\rangle$, and $\langle 8, 10\rangle$ for pattern "A[0, 2]T", and three nonoverlapping occurrences, <1,3,5>, <5,7,8>, and <8,10,12> for pattern "A[0, 2]T[0, 2]A" in S_2, i.e. $sup("A[0, 2]T[0, 2]A", S_2)=sup("A[0, 2]T", S_2)=3$. Hence, pattern "A[0, 2]T" is a right unclosed pattern, which is consistent with Theorem 3.

Theorem 4. *If pattern P is either a left or a right unclosed pattern, then P is an unclosed pattern.*

Proof. If pattern P is a left unclosed pattern, then there is a super-pattern $P'=lP$ which satisfies $sup(P, S)= sup(P', S)$. Thus, according to Definition 6, pattern P is an unclosed pattern. The same strategy can be applied to the right side.

Example 11. In Example 9, pattern "T" is a right unclosed pattern, since there exists a right gap super-pattern "T[0, 2]A", that $sup("T[0, 2]A", S_2)= sup("T", S_2)=3$.

In the following subsections, we will propose two strategies to detect the closeness of the frequent patterns.

4.3.2. Predicting

Theorem 5. *Let pattern Q be the pattern with the highest support of all the patterns that can be connected with pattern P. If $sup(P) > sup(Q)$, then P is a closed pattern.*

Proof. Knowing that $sup(P)>sup(Q)$, $min(sup(P), sup(Q))$ is $sup(Q)$. Patterns P and Q can generate super-pattern R by pattern growth. According to Apriori, $sup(R)\leq min(sup(P), sup(Q))$. Thus, $sup(R)\leq sup(Q)<sup(P)$. Therefore, there is no super-pattern R of P that satisfies $sup(R)=sup(P)$. Hence, P is a closed pattern.

Example 12. In Example 1, patterns "AA", "AT", "AG", and "AC" are the frequent patterns with length 2 in sequence S_3. It is known that, $sup("AA")=3$ and $sup("AC")=sup("AT")=sup("AG")= 2$. The support of the pattern "AA" is greater than that of other patterns. According to Theorem 3, pattern "AA" is a closed pattern.

4.3.3. Determining

Theorem 6. *If P is both a left and right closed pattern, then P is a closed pattern.*

Proof. Suppose P is a left closed pattern which means that there is no super-pattern lP that satisfies $sup(lP) = sup(P)$. Similarly, suppose P is a right closed pattern, which means there is no super-pattern Pr that satisfies $sup(Pr) = sup(P)$. Hence, P is a closed pattern according to Definition 6.

Example 13. In Example 1, $sup("A", S_2)=6$, there is no left or right gap super-pattern of pattern "A" that has the same support as pattern "A". Thus, pattern "A" is a left closed pattern and a right closed pattern. Hence, pattern "A" is a closed pattern.

4.4. NetNCSP

In this subsection, we propose NetNCSP. At the beginning, NetNCSP traverses the sequence to find the frequent letters, and stores them in candidate set C. In the following procedures, three pruning strategies are applied to check the closeness of pattern P in C. In the first step, if P is an unclosed pattern according to Theorem 3, then NetNCSP will add P to temporary candidate set C_1 and restart from the first step with another P in C. Otherwise, NetNCSP goes to the second step. In the second step, BackTr will

calculate the nonoverlapping support of P and store the result in *sup*. If *sup* is less than *minsup*, NetNCSP will go to the first step with another P in C. Otherwise, NetNCSP adds P to C_1 and goes to the third step. In the third step, NetNCSP will check the closeness of pattern P according to Theorems 5 and 6. If the pattern is closed, NetNCSP will add P to nonoverlapping closed pattern set C_P. After that, NetNCSP will restart from the first step with another P in C. After all patterns in C being traversed, NetNCSP employs the pattern growth strategy to generate the candidate set using C_1. NetNCSP will stop until C is empty. All closed patterns are stored in C_P.

Algorithm 2 NetNCSP.

Input: sequence S, support threshold *minsup*, gap constraint *gap*, length constraint *len*
Output: nonoverlapping closed pattern set C_P

```
Traverse S and store all frequent letters in candidate set C;
while C <> NULL do
    for each P in C do
        if inheriting(P)==unclosed then
            C1=C1∪P;
            continue;
        end if
        sup=BackTr(P, S);
        if sup<minsup then
            continue;
        end if
        C1=C1∪P;
        if predicting(P)==closed or determining(P)==closed then
            CP=CP∪P;
        end if
    end for
    C=patterngrowth(C1);
end while
return CP
```

Theorem 7. *The time complexity of NetNCSP is $O(m \times N \times w \times t)$ in the worst case and $O(m \times n \times w \times t/r/r)$ on average, where t is the number of runs of the BackTr algorithm.*

Proof. According to Theorem 1, the time complexity of BackTr is $O(m \times n \times w)$ in the worst case, and $O(m \times n \times w/r/r)$ on average. Since BackTr runs t times, the time complexity of NetNCSP is $O(m \times n \times w \times t)$ and the average time complexity of NetNCSP is $O(m \times n \times w \times t/r/r)$.

Theorem 8. *The space complexity of NetNCSP, the same as that of BackTr, is $O(m \times n \times w)$ in the worst case and $O(m \times n \times w/r/r)$ in the average case.*

Proof. The space of the candidate patterns and the closed patterns can be neglected. Therefore, the space complexity of NetNCSP is the same as that of BackTr. According to Theorem 1, the space complexity of NetNCSP is $O(m \times n \times w)$ in the worst case and $O(m \times n \times w/r/r)$ in the average case.

5. Experimental analysis

Section 5.1 explains the benchmark datasets. Section 5.2 introduces the competitive algorithms. Section 5.3 shows the mining performance of different strategies, such as candidate generation strategies, pruning strategies, and support calculation strategies. Section 5.4 verifies the mining capability of the proposed algorithm and the competitive algorithms. Section 5.5 furtherly reports biological application in COVID-19.

All experiments are conducted on a computer with Intel Core i5, 1.6 GHz CPU, 8 GB 1600 MHz DDR3 memory, and Mac OS (10.14.5) operating system. All the algorithms are developed in Visual Studio Code 1.36.1 which also runs as the experimental environment.

5.1. Benchmark datasets

Table 2 explains the benchmark datasets used in the following experiments.

5.2. Baseline methods

1. NetNCSP-noinh and NetNCSP-nocheck: To verify the efficiency of pruning strategies, NetNCSP-noinh and NetNCSP-nocheck are proposed. NetNCSP-noinh removes the inheriting strategy, while NetNCSP-nocheck removes the predicting and determining strategies to determine closed patterns according to the definition.
2. NetNCSP-netgap: To analyze the effect of BackTr, NetNCSP-netgap is proposed, which applies NETGAP strategy to mine the closed patterns.
3. NetNCSP-bf and NetNCSP-df: To analyze the effect of pattern growth strategy, NetNCSP-bf and NetNCSP-df are proposed to generate candidate patterns according to breadth-first and depth-first strategies, respectively.
4. NOSEP and CloGSgrow: To analyze the differences between NetNCSP and classical SPM algorithms, we employ NOSEP and CloGSgrow as competitive algorithms which were proposed in References [33] and [34], respectively.
5. NetNCSP-nogap: To analyze the effect of gap constraint, NetNCSP-nogap is proposed to mine continuous patterns without gap.

The datasets and all algorithms can be downloaded from http://wuc.scse.hebut.edu.cn.

5.3. Mining performance

In this subsection, we will verify the mining performance of NetNCSP with different strategies. Five competitive algorithms are selected, NetNCSP-bf, NetNCSP-df, NetNCSP-noinh, NetNCSP-nocheck, and NetNCSP-netgap. We use five databases to carry out the experiments, DNA2, DNA4, DNA5, Potato_virus and Ebola_virus. The parameters are *len* =[1,2000], *gap*= [0,200] and *minsup*=2000. The running time, support calculation times, and closeness determination times are shown in Figs. 3–5.

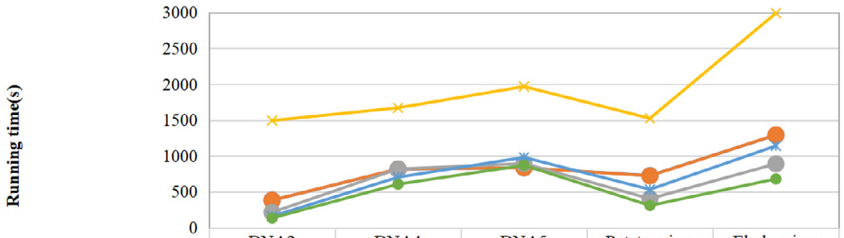
The results indicate the following observations:

1. The inheriting, predicting, and determining strategies are significantly effective. From Fig. 3, it is clear that NetNCSP is faster than NetNCSP-noinh and NetNCSP-nocheck. NetNCSP, NetNCSP-noinh, and NetNCSP-nocheck run 308.9, 403.3, and 1523.4 s in Potato_virus, respectively. From Fig. 4, NetNCSP-noinh and NetNCSP-nocheck calculate support 1150 and 5460 times, respectively, while NetNCSP only calculates 856 times. From Fig. 5, NetNCSP-noinh and NetNCSP-nocheck determine closeness 1364 and 5460 times, respectively, while NetNCSP only needs 916 times. The experiments verify that NetNCSP employs the inheriting, predicting, and determining strategies, which improve the mining efficiency significantly. Hence, NetNCSP outperforms NetNCSP-noinh and NetNCSP-nocheck.

Table 2
Benchmark datasets.

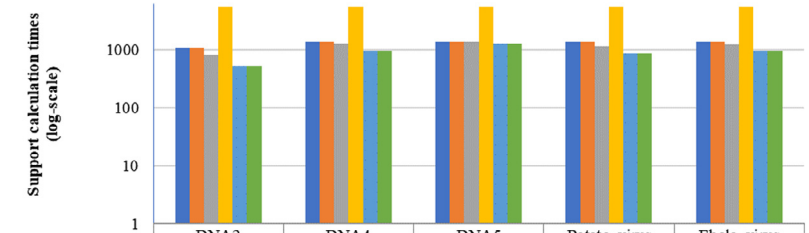
Dataset	Type	From	Length
AX8291741[a]	DNA	Homo sapiens (human)	10,011
DNA1[b]	DNA	Homo sapiens AL158070	6,000
DNA2	DNA	Homo sapiens AL158070	8,000
DNA3	DNA	Homo sapiens AL158070	10,000
DNA4	DNA	Homo sapiens AL158070	12,000
DNA5	DNA	Homo sapiens AL158070	14,000
DNA6	DNA	Homo sapiens AL158070	16,000
Potato_virus[c]	Virus	Potato virus Y Wilga MV99	9,699
Ebola_virus[d]	Virus	Reston Ebola virus	18,891
SARS-CoV-2[e]	Virus	Severe acute respiratory syndrome coronavirus 2(COVID-19)	29,903
SARS[f]	Virus	Severe acute respiratory syndrome-related coronavirus	29,751

[a]AX829174 is used in References [31] and [29] for mining frequent patterns and closed patterns, which can be downloaded from https://www.ncbi.nlm.nih.gov/nuccore/AX829174.1/.
[b]DNA1-6 databases are used in Reference [34], which can be downloaded from https://www.ncbi.nlm.nih.gov/nuccore/AL158070.11.
[c]Potato_virus Y is used in Reference [52] to mine continuous closed patterns and closed patterns with gap constraints and can be downloaded from https://www.ebi.ac.uk/ena/data/view/Taxon:1107954.
[d]Ebola_virus is commonly used in biological sequence analysis and can be downloaded with Potato_virus from https://www.ebi.ac.uk/ena/data/view/Taxon:129003.
[e]This sequence was reported in Reference [66], and can be downloaded from https://www.ncbi.nlm.nih.gov/nuccore/MN908947.
[f]This sequence can be downloaded from https://www.ncbi.nlm.nih.gov/nuccore/30271926.



	DNA2	DNA4	DNA5	Potato_virus	Ebola_virus
NetNCSP-bf	382.1	815.3	834.9	725.6	1290.7
NetNCSP-df	382.1	817.8	835.3	722.7	1293.4
NetNCSP-noinh	213.0	817.8	894.5	403.3	890.1
NetNCSP-nocheck	1494.0	1669.9	1967.8	1523.4	2990.3
NetNCSP-netgap	158.6	702.7	980.6	532.4	1140.3
NetNCSP	133.6	605.3	869.7	308.9	679.0

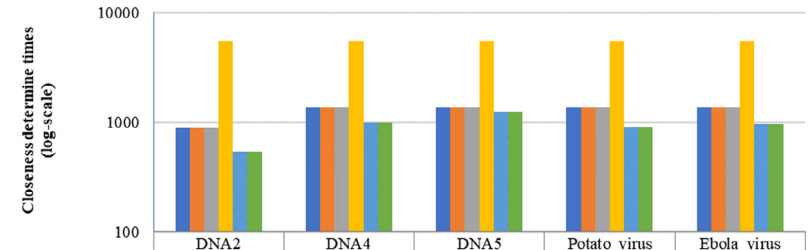
Fig. 3. Running time with different strategies.



	DNA2	DNA4	DNA5	Potato_virus	Ebola_virus
NetNCSP-bf	1048	1364	1364	1364	1364
NetNCSP-df	1048	1364	1364	1364	1364
NetNCSP-noinh	796	1262	1338	1150	1230
NetNCSP-nocheck	5460	5460	5460	5460	5460
NetNCSP-netgap	530	962	1246	856	940
NetNCSP	530	962	1246	856	940

Fig. 4. Support calculation times with different strategies.

2. The BackTr strategy is significantly effective. According to Figs. 3–5, NetNCSP and NetNCSP-netgap have the same closeness determination times and support calculation times, but the running time of NetNCSP is less than that of NetNCSP-netgap. For example, in Fig. 3, the running time of NetNCSP and NetNCSP-netgap are 679.0 and 1140.3 s in Ebola_virus, respectively. The reason is that NetNCSP employs the *BackTr* strategy, and reduces the running time by pruning invalid nodes, thus reducing the time complexity from $O(m \times m \times n \times W)$ to $O(m \times n \times W)$. Hence, NetNCSP outperforms NetNCSP-netgap.
3. The pattern growth strategy is significantly effective. From Fig. 3, we can see that NetNCSP is faster than NetNCSP-bf and NetNCSP-df. For example, NetNCSP runs 679.0 s in Ebola_virus, while NetNCSP-bf and NetNCSP-df run 1290.7 and 1293.4 s, respectively. The reason is that NetNCSP calculates 940 candidate patterns, while NetNCSP-bf and



	DNA2	DNA4	DNA5	Potato_virus	Ebola_virus
NetNCSP-bf	895	1364	1364	1364	1364
NetNCSP-df	895	1364	1364	1364	1364
NetNCSP-noinh	895	1364	1364	1364	1364
NetNCSP-nocheck	5460	5460	5460	5460	5460
NetNCSP-netgap	542	992	1260	916	964
NetNCSP	542	992	1260	916	964

Fig. 5. Closeness determination times with different strategies.

NetNCSP-df calculate 1364 candidate patterns. The reason lies in that NetNCSP employs pattern growth strategy to generate the candidate patterns, while NetNCSP-bf and NetNCSP-df employ the breadth-first and depth-first strategies, respectively. The pattern growth strategy is more effective than the other two, which is consistent with the analysis in Example 8. Hence, NetNCSP outperforms NetNCSP-bf and NetNCSP-df.

In conclusion, NetNCSP has better performance than all the competitive algorithms.

5.4. Mining capability

In this subsection, we carry out two experiments to verify the nonoverlapping closed pattern mining ability and the performance of NetNCSP.

We used the DNA1 database to conduct the first experiment to mine patterns with pattern lengths 2 to 7. The parameters are *len*=[1,1000], *gap*=[0,100], and *minsup*=1200. The results are shown in Figs. 6–8.

To compare the mining capability of algorithms in different databases, six sequences are included to mine closed patterns with length 5, which are DNA1, DNA3, DNA6, AX829174, Potato_virus, and Ebola_virus. The parameters are *len*=[1,1000], *gap*=[0,100], and *minsup*=1200. The results are shown in Figs. 9–11.

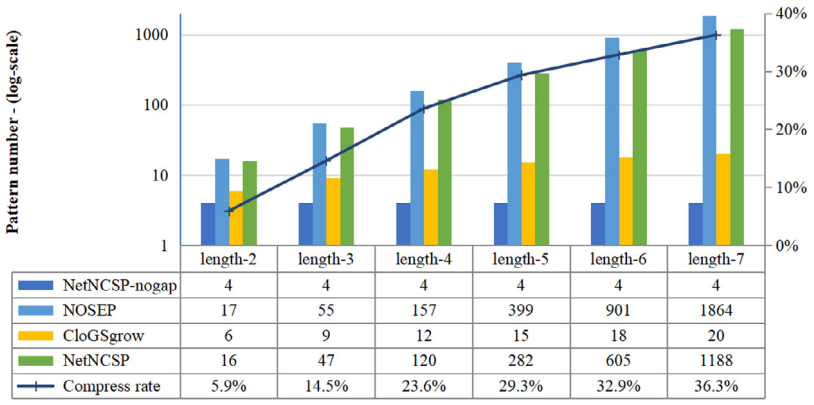
The results indicate the following observations:

1. NetNCSP outperforms NOSEP since NetNCSP not only compresses the patterns effectively but also achieves higher efficiency. For example, in Fig. 6, when the pattern length is 7, NOSEP gets 1864 patterns, while NetNCSP gets 1188. The running time of NOSEP and NetNCSP are 431.1 and 351.8 s, respectively. Similar phenomena can be discovered in different databases. For example, in Fig. 10, the running time of NOSEP and NetNCSP are 263.8 and 189.6 s in AX829174, respectively. The reasons are as follows. Firstly, NetNCSP adopts a more efficient backtracking strategy in calculating pattern support, and reduces the time complexity from $O(m \times m \times n \times W)$ to $O(m \times n \times W)$. More importantly, NetNCSP adopts three effective pruning strategies. The predicting and inheriting strategies prune the redundant unclosed patterns before calculating support of candidate patterns, and the determining strategy determines the closeness of frequent patterns avoiding the generation of candidate pattern of longer length. Hence, NetNCSP outperforms NOSEP.
2. NetNCSP outperforms CloGSgrow since NetNCSP mines more closed patterns and is more efficient in terms of ATC (Average mining Time per Closed pattern). For example, in Fig. 6, when the pattern length is 5, NetNCSP mines 282 closed patterns and CloGSgrow only gets 15. Although Fig. 7 shows that when the pattern length is 5, the running time of NetNCSP and CloGSgrow are 64.1 and 5.6 s respectively, the ATCs of NetNCSP and CloGSgrow are 64.1/282=0.2 and 5.6/15=0.4 s, respectively. Similar phenomena can be discovered in different databases. For example, in Figs. 9 and 10, CloGSgrow takes 83.3 s to get 33 closed patterns in Ebola_virus, while NetNCSP takes 453.8 s to get 912. The ATCs of CloGSgrow and NetNCSP are 2.5 and 0.5 s, respectively. The reasons are as follows. First, NetNCSP, as a complete algorithm, mines the complete closed patterns. Thus, more closed patterns can be discovered. Second, NetNCSP adopts three pruning strategies to reduce the running time effectively. As a result, NetNCSP outperforms CloGSgrow.
3. NetNCSP can compress the patterns effectively. For example, in Fig. 6, NOSEP and NetNCSP find 399 frequent patterns and 282 closed patterns with length 5, respectively. Thus, the compression rate is (399-282)/399= 29.3%. In Fig. 9, NOSEP and NetNCSP mine 1364 frequent patterns and 912 closed patterns in Ebola_virus, respectively. Thus, the compression rate is 33.1%. The reason is that NOSEP mines the complete set of the frequent patterns which contains redundant patterns, while NetNCSP mines a subset of the closed patterns according to three closeness determination pruning strategies.
4. NetNCSP reduces the support calculation times effectively. For example, Fig. 8 shows that when mining patterns with length 7, NetNCSP calculates the support for 1443 times, while NOSEP needs 2134 times. Similar phenomena can be found in different databases. For example, Fig. 11 shows that NetNCSP calculates support for 932 times in Ebola_virus, while NOSEP needs 1364 times. The reason is that NetNCSP adopts the predicting and inheriting strategies to prune the redundant patterns, thus, reducing the support calculation times. As a result, NetNCSP reduces support calculation times.

In conclusion, NetNCSP compresses the patterns effectively and outperforms competitive algorithms.

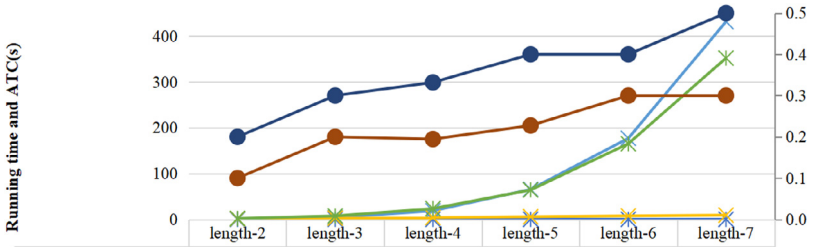
5.5. Biological application

Recently, a severe respiratory syndrome named COVID-19 spreads around the world, and over 300,000 people worldwide



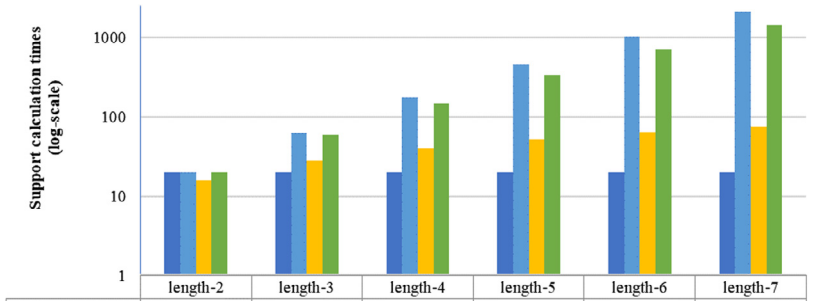
	length-2	length-3	length-4	length-5	length-6	length-7
NetNCSP-nogap	4	4	4	4	4	4
NOSEP	17	55	157	399	901	1864
CloGSgrow	6	9	12	15	18	20
NetNCSP	16	47	120	282	605	1188
Compress rate	5.9%	14.5%	23.6%	29.3%	32.9%	36.3%

Fig. 6. Pattern number with different pattern length.



	length-2	length-3	length-4	length-5	length-6	length-7
NetNCSP-nogap	0.1	0.1	0.1	0.1	0.1	0.1
NOSEP	0.5	3.9	18.5	65.5	176.9	431.1
CloGSgrow	1.4	2.6	4.0	5.6	7.5	9.1
NetNCSP	2.0	7.5	23.3	64.1	165.1	351.8
ATC-CloGSgrow	0.2	0.3	0.3	0.4	0.4	0.5
ATC-NetNCSP	0.1	0.2	0.2	0.2	0.3	0.3

Fig. 7. Running time and ATC with different pattern length.



	length-2	length-3	length-4	length-5	length-6	length-7
NetNCSP-nogap	20	20	20	20	20	20
NOSEP	20	63	177	457	1038	2134
CloGSgrow	16	28	40	52	64	76
NetNCSP	20	59	147	338	726	1443

Fig. 8. Support calculation times with different pattern length.

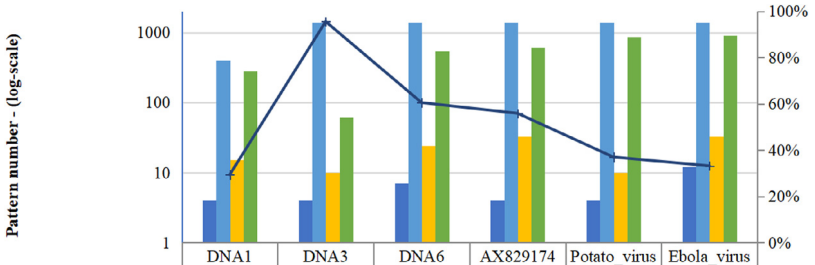
have been identified with the disease. COVID-19 is caused by the SARS-CoV-2 virus [66], which is reported to be closely related to a group of SARS-like coronaviruses (of 89.1% nucleotide similarity).

In this subsection, we use the proposed algorithm to study the similarities between the two viruses. NetNCSP is employed to mine closed frequent patterns from SARS-CoV-2 and SARS viruses. We set the parameters with *len* = [1,2000], *gap* = [0,200], and *minsup* = 2000 to mine patterns of length 2 to 10. The comparison of closed pattern number is reported in Fig. 12.

The results report the following observations. When the pattern length is less than 8, the number of closed patterns discovered from SARS-CoV-2 and SARS is almost the same, while when the pattern length is greater than 7, the number of closed patterns is significantly different. For example, in Fig. 12, when the pattern length is 4, the number of closed patterns discovered from SARS-CoV-2 and SARS are the same 18. When pattern length is 9, the closed patterns are 270 and 370, respectively. The possible reasons for this phenomenon are as follows. The basic pattern composition of SARS-CoV-2 and SARS is the same. Thus, the shorter patterns are nearly the same, while the longer ones have diversities with different pattern combinations.

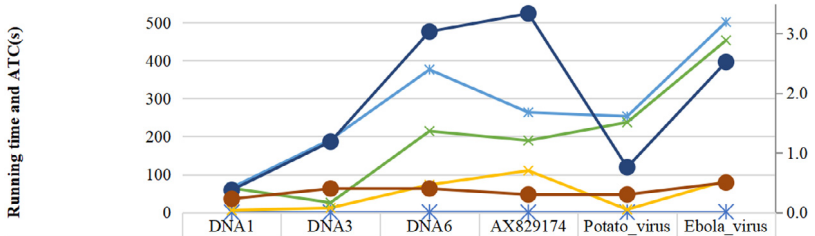
6. Conclusion

In this paper, we tackle the problem of nonoverlapping periodic gapped closed SPM and propose an effective closed pattern mining algorithm, named NetNCSP, which has two major steps, support calculation and closeness determination. In the



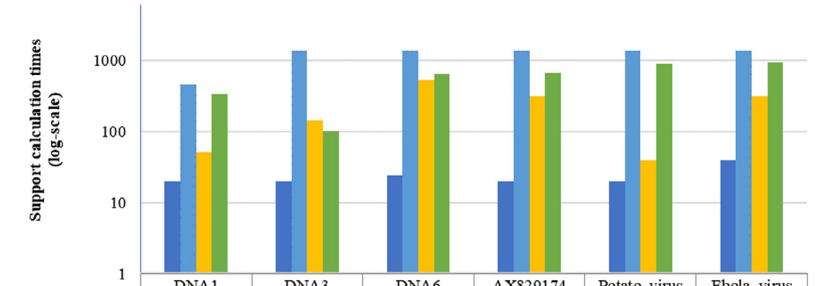
	DNA1	DNA3	DNA6	AX829174	Potato_virus	Ebola_virus
NetNCSP-nogap	4	4	7	4	4	12
NOSEP	399	1364	1364	1364	1364	1364
CloGSgrow	15	10	24	33	10	33
NetNCSP	282	61	539	602	860	912
Compress rate	29.3%	95.5%	60.5%	55.9%	37.0%	33.1%

Fig. 9. Pattern number with different databases.



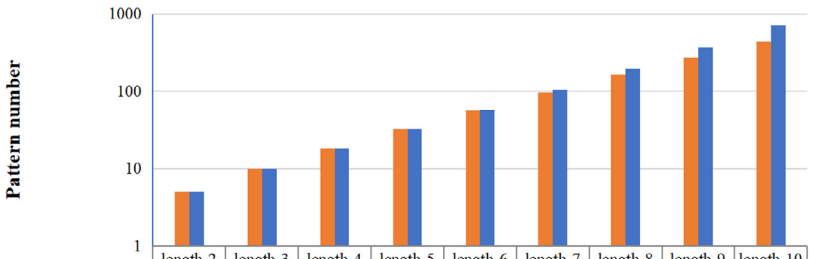
	DNA1	DNA3	DNA6	AX829174	Potato_virus	Ebola_virus
NetNCSP-nogap	0.1	0.2	0.8	0.3	0.3	1.7
NOSEP	65.5	192.3	376.6	263.8	253.4	502.1
CloGSgrow	5.6	11.9	72.8	110.2	7.6	83.3
NetNCSP	64.1	25.6	214.3	189.6	237.5	453.8
ATC-CloGSgrow	0.4	1.2	3.0	3.3	0.8	2.5
ATC-NetNCSP	0.2	0.4	0.4	0.3	0.3	0.5

Fig. 10. Running time and ATC with different databases.



	DNA1	DNA3	DNA6	AX829174	Potato_virus	Ebola_virus
NetNCSP-nogap	20	20	24	20	20	40
NOSEP	457	1364	1364	1364	1364	1364
CloGSgrow	52	144	532	316	40	316
NetNCSP	338	102	651	673	881	932

Fig. 11. Support calculation times with different databases.



	length-2	length-3	length-4	length-5	length-6	length-7	length-8	length-9	length-10
SARS-CoV-2	5	10	18	32	56	97	164	270	431
SARS	5	10	18	32	57	104	194	370	717

Fig. 12. Comparison of closed pattern number in different databases.

process of support calculation, NetNCSP adopts the BackTr algorithm with backtracking strategy to calculate the nonoverlapping support of patterns in Nettree. In the process of closeness determination, NetNCSP adopts three pruning strategies, inheriting, predicting, and determining. The inheriting strategy can avoid invalid calculations on the redundant patterns. The predicting strategy can reduce the number of candidate patterns effectively. The determining strategy determines the closeness of the frequent patterns and prunes the redundant patterns. NetNCSP is of lower time complexity than the state-of-the-art algorithms. Experiments are carried out on long-length sequence databases, such as DNA and virus. The experimental results show that NetNCSP has better performance than competitive algorithms and compresses the frequent patterns effectively. In the experiment of biological application, we employ NetNCSP in mining biological sequences in SARS-CoV-2 and SARS viruses, the results show that the two viruses are of similar pattern composition with different combinations.

In this paper, we focus on nonoverlapping closed sequential pattern mining in a long sequence. Experimental results report that NetNCSP can compress the frequent patterns effectively in long sequences with a small item set, such as DNA/virus sequences. However, we notice that, NetNCSP cannot compress the frequent patterns in a short sequence with a large item set, such as a protein sequence, and NetNCSP is more suitable when the gap constraints are large. Hence, generator mining and maximal pattern mining can be explored in the future.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

CRediT authorship contribution statement

Youxi Wu: Conceptualization, Methodology, Formal analysis, Supervision, Funding acquisition. **Changrui Zhu:** Software, Writing - original draft, Validation, Investigation, Data curation. **Yan Li:** Investigation, Writing - review & editing. **Lei Guo:** Validation, Resources. **Xindong Wu:** Supervision, Funding acquisition.

Acknowledgments

This work was partly supported by National Natural Science Foundation of China (61976240, 917446209), National Key Research and Development Program of China (2016YFB1000901), and Graduate Student Innovation Program of Hebei Province, China (CXZZBS2020024).

References

[1] R. Srikant, R. Agrawal, Mining sequential patterns: Generalizations and performance improvements, in: International Conference on Extending Database Technology, Springer, 1996, pp. 1–17.
[2] K.-i. Fukui, Y. Okada, K. Satoh, M. Numao, Cluster sequence mining from event sequence data and its application to damage correlation analysis, Knowl.-Based Syst. 179 (2019) 136–144.
[3] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, M.C. Hsu, FreeSpan: Frequent pattern-projected sequential pattern mining, in: Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2000, pp. 355–359.
[4] X. Wu, X. Zhu, G.Q. Wu, W. Ding, Data mining with big data, IEEE Trans. Knowl. Data Eng. 26 (1) (2014) 97–107.
[5] X. Wu, D. Theodoratos, Homomorphic pattern mining from a single large data tree, Data Sci. Eng. 1 (4) (2016) 203–218.
[6] M. Wu, X. Wu, On big wisdom, Knowl. Inf. Syst. 58 (1) (2019) 1–8.
[7] B. Le, M.-T. Tran, B. Vo, Mining frequent closed inter-sequence patterns efficiently using dynamic bit vectors, Appl. Intell. 43 (1) (2015) 74–84.
[8] X. Wu, X. Zhu, Y. He, A.N. Arslan, PMBC: Pattern mining from biological sequences with wildcard constraints, Comput. Biol. Med. 43 (5) (2013) 481–492.
[9] P. Fournier-Viger, J. Li, J.C.W. Lin, T.T. Chi, R.U. Kiran, Mining cost-effective patterns in event logs, Knowl.-Based Syst. (2019), http://dx.doi.org/10.1016/j.knosys.2019.105241.
[10] X. Dong, P. Qiu, J. Lü, L. Cao, T. Xu, Mining top-k useful negative sequential patterns via learning, IEEE Trans. Neural Netw. Learn. Syst. 30 (9) (2019) 2764–2778.
[11] X. Dong, Y. Gong, L. Cao, E-RNSP: An efficient method for mining repetition negative sequential patterns, IEEE Trans. Cybern. (2018), http://dx.doi.org/10.1109/TCYB.2018.2869907.
[12] U. Yun, G. Lee, K.H. Ryu, Mining maximal frequent patterns by considering weight conditions over data streams, Knowl.-Based Syst. 55 (2014) 49–65.
[13] Y. Guo, Y. Hao, M. Yu, Image retargeting quality assessment based on content deformation measurement, Signal Process., Image Commun. 67 (2018) 171–181.
[14] F. Min, Z.H. Zhang, W.J. Zhai, R.P. Shen, Frequent pattern discovery with tri-partition alphabets, Inform. Sci. 507 (2020) 715–732.
[15] C. Tan, F. Min, M. Wang, H. Zhang, Z. Zhang, Discovering patterns with weak-wildcard gaps, IEEE Access 4 (2016) 4922–4932.
[16] N. Pasquier, Y. Bastide, R. Taouil, L. Lakhal, Discovering frequent closed itemsets for association rules, in: International Conference on Database Theory, Springer, 1999, pp. 398–416.
[17] M. Fabrègue, A. Braud, S. Bringay, F. Le Ber, M. Teisseire, Mining closed partially ordered patterns, a new optimized algorithm, Knowl.-Based Syst. 79 (2015) 68–79.
[18] H. Yang, L. Duan, B. Hu, S. Deng, W. Wang, P. Qin, Mining top-k distinguishing sequential patterns with gap constraint, J. Softw. 26 (11) (2015) 2994–3009.
[19] Y. Wu, L. Wang, J. Ren, W. Ding, X. Wu, Mining sequential patterns with periodic wildcard gaps, Appl. Intell. 41 (1) (2014) 99–116.
[20] T.-L. Dam, H. Ramampiaro, K. Nørvåg, Q.-H. Duong, Towards efficiently mining closed high utility itemsets from incremental databases, Knowl.-Based Syst. 165 (2019) 13–29.
[21] J.-P. Qiang, P. Chen, W. Ding, F. Xie, X. Wu, Multi-document summarization using closed patterns, Knowl.-Based Syst. 99 (2016) 28–38.
[22] F. Zhu, X. Yan, J. Han, S.Y. Philip, Efficient discovery of frequent approximate sequential pattern, in: Seventh IEEE International Conference on Data Mining, 2007, pp. 751–756.
[23] J. Zhang, J. Guo, X. Yu, X. Yu, W. Guo, T. Zeng, L. Chen, Mining K-mers of various lengths in biological sequences, in: International Symposium on Bioinformatics Research and Applications, 2017, pp. 186–195.
[24] U. Manber, R. Baeza-Yates, An algorithm for string matching with a sequence of don't cares, Inform. Process. Lett. 37 (3) (1991) 133–136.
[25] P. Bille, I.L. Gørtz, H.W. Vildhøj, D.K. Wind, String matching with variable length gaps, Theoret. Comput. Sci. 443 (2012) 25–34.
[26] X. Wang, L. Chai, Q. Xu, Y. Yang, J. Li, J. Wang, Y. Chai, Efficient subgraph matching on large RDF graphs using mapreduce, Data Sci. Eng. 4 (1) (2019) 24–43.
[27] Y. Wu, Z. Tang, H. Jiang, X. Wu, Approximate pattern matching with gap constraints, J. Inf. Sci. 42 (5) (2016) 99–116.
[28] Y. Wu, S. Fu, H. Jiang, X. Wu, Strict approximate pattern matching with general gaps, Appl. Intell. 42 (3) (2015) 566–580.
[29] M. Zhang, B. Kao, D.W. Cheung, K.Y. Yip, Mining periodic patterns with gap requirement from sequences, ACM Trans. Knowl. Discov. Data 1 (2) (2007) 7.
[30] Q. Shi, J. Shan, W. Yan, Y. Wu, X. Wu, NetNPG: Nonoverlapping pattern matching with general gap constraints, Appl. Intell. (2020), http://dx.doi.org/10.1007/s10489-019-01616-z.
[31] C. Li, Q. Yang, J. Wang, M. Li, Efficient mining of gap-constrained subsequences and its various applications, ACM Transa. Knowl. Discov. Data 6 (1) (2012) 2.
[32] X. Chen, Y. Rao, H. Xie, F.L. Wang, Y. Zhao, J. Yin, Sentiment classification using negative and intensive sentiment supplement information, Data Sci. Eng. 4 (2) (2019) 109–118.
[33] B. Ding, D. Lo, J. Han, S.-C. Khoo, Efficient mining of closed repetitive gapped subsequences from a sequence database, in: 2009 IEEE 25th International Conference on Data Engineering, IEEE, 2009, pp. 1024–1035.
[34] Y. Wu, Y. Tong, X. Zhu, X. Wu, NOSEP: Nonoverlapping sequence pattern mining with gap constraints, IEEE Trans. Cybern. 48 (10) (2018) 2809–2822.
[35] Y. Wu, C. Shen, H. Jiang, X. Wu, Strict pattern matching under nonoverlapping condition, Sci. China Inf. Sci. 60 (1) (2017) 012101.
[36] R. Agrawal, R. Srikant, Mining sequential patterns, in: IEEE International Conference on Data Engineering, vol. 95, 1995, pp. 3–14.
[37] W. Song, B. Jiang, Y. Qiao, Mining multi-relational high utility itemsets from star schemas, Intell. Data Anal. 22 (1) (2018) 143–165.
[38] Y. Wu, Y. Wang, J. Liu, M. Yu, J. Liu, Y. Li, Mining distinguishing subsequence patterns with nonoverlapping condition, Cluster Comput. 22 (3) (2019) 5905–5917.

[39] S. Ghosh, J. Li, L. Cao, K. Ramamohanarao, Septic shock prediction for ICU patients via coupled HMM walking on sequential contrast patterns, J. Biomed. Inform. 66 (2017) 19–31.
[40] F. Fumarola, P.F. Lanotte, M. Ceci, D. Malerba, CloFast: Closed sequential pattern mining using sparse and vertical id-lists, Knowl. Inf. Syst. 48 (2) (2016) 429–463.
[41] F.A.M. Zaki, N.F. Zulkurnain, RARE: Mining colossal closed itemset in high dimensional data, Knowl.-Based Syst. 161 (2018) 1–11.
[42] T. Truong, H. Duong, B. Le, P. Fournier-Viger, FMaxCloHUSM: An efficient algorithm for mining frequent closed and maximal high utility sequences, Eng. Appl. Artif. Intell. 85 (2019) 1–20.
[43] B. Le, H. Duong, T. Truong, P. Fournier-Viger, M. F.C.loS, FGenSM: two efficient algorithms for mining frequent closed and generator sequences using the local pruning strategy, Knowl. Inf. Syst. 53 (1) (2017) 71–107.
[44] D. Lo, S.C. Khoo, J. Li, Mining and ranking generators of sequential patterns, in: Proceedings of the 2008 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics, 2008, pp. 553–564..
[45] H. Fasihy, M.H.N. Shahraki, Incremental mining maximal frequent patterns from univariate uncertain data, Knowl.-Based Syst. 152 (2018) 40–50.
[46] J. Zhang, Y. Wang, D. Yang, CCSpan: Mining closed contiguous sequential patterns, Knowl.-Based Syst. 89 (2015) 1–13.
[47] S. Akther, M.R. Karim, M. Samiullah, C.F. Ahmed, Mining nonredundant closed flexible periodic patterns, Eng. Appl. Artif. Intell. 69 (2018) 1–23.
[48] U. Niranjan, R. Subramanyam, V. Khanaa, Developing a web recommendation system based on closed sequential patterns, in: Information & Communication Technologies-International Conference, Springer, 2010, pp. 171–179.
[49] L. Abualigah, M. Shehab, M. Alshinwan, H. Alabool, Salp swarm algorithm: A comprehensive survey, Neural Comput. Appl. (2019), http://dx.doi.org/10.1007/s00521-019-04629-4.
[50] L. Abualigah, A.T. Khader, E.S. Hanandeh, Hybrid clustering analysis using improved krill herd algorithm, Appl. Intell. 48 (11) (2018) 4047–4071.
[51] L. Abualigah, A. Khader, E. Hanandeh, A. Gandomi, A novel hybridization strategy for krill herd algorithm applied to clustering techniques, Appl. Soft Comput. 60 (2017) 423–435.
[52] B. Lavanya, A. Murugan, A DNA based approach to find closed repetitive gapped subsequences from a sequence database, Int. J. Comput. Appl. 29 (5) (2011) 45–49.
[53] R. Dhanaseelan, M.J. Sutha, Diagnosis of coronary artery disease using an efficient hash table based closed frequent itemsets mining, Med. Biol. Eng. Comput. 56 (5) (2018) 749–759.
[54] L. Nie, H. Jiang, Z. Ren, Z. Sun, X. Li, Query expansion based on crowd knowledge for code search, IEEE Trans. Serv. Comput. 9 (5) (2016) 771–783.
[55] H. Jiang, X. Chen, T. He, Z. Chen, X. Li, Fuzzy clustering of crowdsourced test reports for apps, ACM Trans. Internet Technol. (TOIT) 18 (2) (2018) 1–28.
[56] F. Min, Y. Wu, X. Wu, The apriori property of sequence pattern mining with wildcard gaps, in: IEEE International Conference on Bioinformatics & Biomedicine Workshops, IEEE, 2011, pp. 138–143.
[57] F. Xie, X. Wu, X. Zhu, Efficient sequential pattern mining with wildcards for keyphrase extraction, Knowl.-Based Syst. 115 (2017) 27–39.
[58] Y. Tong, L. Zhao, D. Yu, S. Ma, Z. Cheng, K. Xu, Mining compressed repetitive gapped sequential patterns efficiently, in: International Conference on Advanced Data Mining and Applications, Springer, 2009, pp. 652–660..
[59] D. Guo, X. Hu, F. Xie, X. Wu, Pattern matching with wildcards and gap-length constraints based on a centrality-degree graph, Appl. Intell. 39 (1) (2013) 57–74.
[60] M.K. Warmuth, D. Haussler, On the complexity of iterated shuffle, J. Comput. System Sci. 28 (3) (1984) 345–358.
[61] X. Yan, J. Han, R. Afshar, CloSpan: Mining closed sequential patterns in large datasets, in: Proceedings of the 2003 SIAM International Conference on Data Mining, Society for Industrial and Applied Mathematics, SIAM, 2003, pp. 166–177.
[62] J. Wang, J. Han, C. Li, Frequent closed sequence mining without candidate maintenance, IEEE Trans. Knowl. Data Eng. 19 (8) (2007) 1042–1056.
[63] T. Hoang, F. Mörchen, D. Fradkin, T. Calders, Mining compressing sequential patterns, Stat. Anal. Data Min. 7 (1) (2014) 34–52.
[64] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, M.-C. Hsu, Mining sequential patterns by pattern-growth: The prefixspan approach, IEEE Trans. Knowl. Data Eng. 16 (11) (2004) 1424–1440.
[65] Y. Wu, S. Li, J. Liu, L. Guo, X. Wu, NETASPNO: Approximate strict pattern matching under nonoverlapping condition, IEEE Access 6 (2018) 24350–24361.
[66] F. Wu, S. Zhao, B. Yu, Y. Chen, W. Wang, Z. Song, Y. Hu, Z. Tao, J. Tian, Y. Pei, M. Yuan, Y. Zhang, F. Dai, Y. Liu, Q. Wang, J. Zheng, L. Xu, E. Holmes, Y. Zhang, A new coronavirus associated with human respiratory disease in China, Nature (2020) 1–5, http://dx.doi.org/10.1038/s41586-020-2008-3.