Cognitive Computation
https://doi.org/10.1007/s12559-021-09871-4

Tri-Partition State Alphabet-Based Sequential Pattern for Multivariate Time Series

Zhi-Heng Zhang[1] · Fan Min[2] · Gong-Suo Chen[1] · Shao-Peng Shen[1] · Zuo-Cheng Wen[1] · Xiang-Bing Zhou[1]

Received: 15 October 2020 / Accepted: 13 April 2021

Abstract
Recently, the advancement of cognitive computing and three-way decisions has enabled in-depth sequential pattern understanding through temporal association analysis. The main challenge is to obtain concise patterns that express richer semantics for multivariate time series (MTS) analysis. In this paper, we propose a tri-partition state alphabet-based sequential pattern (Tri-SASP) for MTSs. First, a tri-wildcard gap inserted between each pair of adjacent states enhances the flexibility of the method. Second, a given set of states is partitioned into positive (POS), negative (NEG) and boundary (BND) regions. The states in POS can only be used to construct a Tri-SASP, the states in NEG can only be matched by a tri-wildcard gap, and the states in BND can be used in both ways. Finally, horizontal and vertical algorithms are proposed to obtain frequent Tri-SASPs in a breadth-first manner. The experimental results on four real-world datasets show that (1) the discovered Tri-SASPs and temporal rules can enrich human cognition; (2) the two tri-partition strategies can bring us very meaningful and varied Tri-SASPs; and (3) the two algorithms are effective and scalable.

Keywords Cognitive computation · MTS · Sequential pattern discovery · Three-way decisions

Introduction

Three-way decisions (3WDs) [1, 2], originating from rough set theory [3], have become a focus of cognitive computation [4, 5]. Through the idea of "thinking in threes" [6], numerous novel machine learning theories such as granular computing [7], concept analysis [8], fuzzy sets [9], interval sets [10] and cost-sensitive learning [11, 12], as well as applications such as recommendation systems [13, 14], active learning [15], classification [16], clustering [17], pattern discovery [18], target recognition [19], attribute reduction [20] and email filtering [21], have been extensively studied. The discovery of novel associations and patterns [22] to promote brain-like computer cognition and decisions brings opportunities and challenges for cognitive computation [23].

✉ Fan Min
minfan@swpu.edu.cn

[1] School of Information and Engineering, Sichuan Tourism University, No. 459 Hongling Road, Longquanyi District, Chengdu City, Sichuan Province 610100, China

[2] School of Computer Science, Southwest Petroleum University, No. 8 Xindu Avenue, Xindu District, Chengdu City, Sichuan Province 610500, China

Figure 1 shows some existing works on frequent sequential pattern discovery initiated by Agrawal and Srikant [24]. Recently, this direction has attracted much attention [25, 26]. Numerous studies of this issue focus on sequence databases [27]. GSP [28] was improved from AprioriAll [24], and both of them were inspired by frequent itemset mining [29]. To avoid the drawbacks (e.g., too many candidates) of the horizontal GSP algorithm, Spade [30], Spam [31], Lapin [32] and PrefixSpam [33] were proposed on the basis of vertical database representation [34]. To obtain concise results, closed [35], maximal [36] and generator frequent sequential patterns [37] were designed. To extract richer patterns, weighted [38], high-utility [39], uncertain [40] and fuzzy [41] methods were defined.

For pattern discovery on MTSs, there are along-first DTA [42], improved from TARD [43] and across-first STAP [44]. As important results of temporal association analysis [45], the rules generated by STAP are more explainable than those generated by DTA. This is because the components of DTA are subsequence clusters, where subsequences are similar but different. The components of STAP are states that are unambiguous at each time stamp. As the foundation of frequency computing, matching conditions of patterns such as non-overlapping [46], one-off [47] and general [48] are

Published online: 18 May 2021

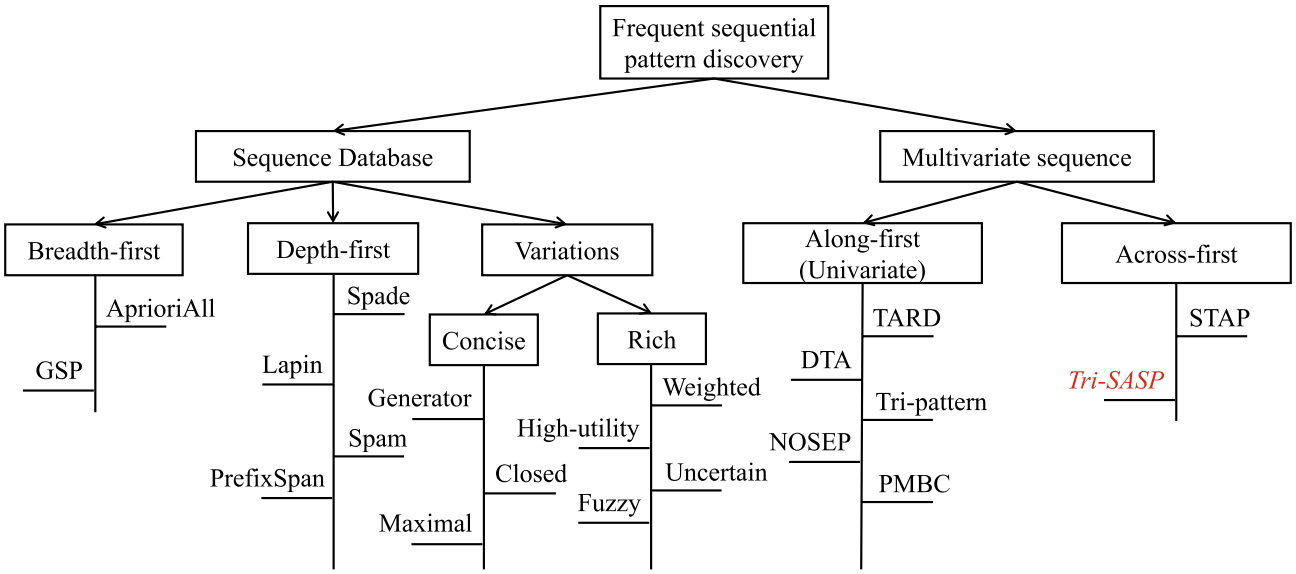


Fig. 1 Techniques of frequent sequential pattern discovery (Tri-SASP is our contribution)

popular strategies. Naturally, there is still a need to enrich the semantics of the pattern further.

In this paper, we propose a new type called a tri-partition state alphabet-based sequential pattern (Tri-SASP) to enrich the pattern type on MTSs. A state is defined as a set of attribute-value pairs. For example, the set {(Temperature, High), (Humidity, Medium)}, also denoted as (Temperature, High) $\wedge$ (Humidity, Medium), is a state of weather. Through inserting a tri-wildcard gap Δ between each pair of adjacent states, the flexibility can be greatly enhanced. Tri-SASP generalizes a tri-pattern [18], which is based on univariate sequences in MTSs. It generalizes STAP through a tri-partition state alphabet with two reasonable strategies. A Tri-SASP is frequent in an MTS if the occurrences of each state exceed a user-specified threshold α and the occurrences of the Tri-SASP exceed another threshold γ. Therefore, the problem is to discover all frequent Tri-SASPs from a given MTS and a tri-partition state alphabet Σ.

First, inspired by three-way decision theory [2], a given set of states can be divided into three disjoint regions, namely, the POS, BND and NEG ones, namely, we have that (1) $\Sigma = \text{POS} \cup \text{BND} \cup \text{NEG}$ and (2) $\text{POS} \cap \text{BND} = \text{POS} \cap \text{NEG} = \text{BND} \cap \text{NEG} = \emptyset$. Here, we discuss two strategies, called support-partition (SP) and compression-partition (CP). For the SP strategy, two thresholds α and β ($\alpha \leq \beta$) are used to obtain Σ_{SP}. If the frequency of a state is not less than β, it belongs to POS. If the frequency is less than α, it belongs to NEG. Otherwise, it belongs to BND. In other words, the set of frequent states is partitioned into POS and BND, while the set of infrequent ones is NEG. The main idea of the SP strategy is to measure the state significance by its frequency. For the CP strategy, only α is used to generate Σ_{CP} through defining frequent closed and maximal states. POS is the set of frequent maximal states. BND is the difference set of frequent closed states and maximal states. NEG is the difference set of all frequent states and closed states. We have $\Sigma_{\text{CP}} \subseteq \Sigma_{\text{SP}}$, and the NEG of Σ_{SP} is not stored. The main idea of the CP strategy is that longer (or more special) states are more significant.

Second, a Tri-SASP is constructed with the given $\Sigma \in \{\Sigma_{\text{SP}}, \Sigma_{\text{CP}}\}$ and tri-wildcard gap. Given the lower bound $\underline{G}$ and upper bound $\overline{G}$ ($0 \leq \underline{G} \leq \overline{G}$), namely, the minimal and maximal gap constraints, the tri-wildcard gap can be denoted as $\Delta = [\underline{G}, \overline{G}]$. The number of time stamps between each pair of adjacent states must be no less than $\underline{G}$ and no more than $\overline{G}$. Consequently, the matching conditions can be defined as follows: (1) if a state comes from POS, it can be a component of the Tri-SASP but cannot be matched by the tri-wildcard gap; (2) if a state belongs to NEG, it can be matched by the tri-wildcard gap but cannot be a component of the Tri-SASP; and (3) if a state is contained in BND, it can either be a component of the pattern or matched by the tri-wildcard gap. In other words, we have (1) all components of a Tri-SASP must be states from POS $\cup$ BND; and (2) the states of POS never appear in these time stamps between each pair of adjacent states.

Third, two algorithms, horizontal and vertical algorithms, are proposed to obtain frequent Tri-SASPs. The general matching condition is introduced here because it maintains the complete occurrences of Tri-SASPs. With the a priori/down-closure property, namely, that if a Tri-SASP is infrequent, all of its super-patterns must be infrequent, the two proposed algorithms are able to obtain frequent patterns rapidly and exactly. The horizontal algorithm computes the

support of a Tri-SASP by re-scanning the MTS. The vertical algorithm obtains this support by computing the interaction of two TPLists belonging to two sub-patterns. The TPList of a Tri-SASP is a set of tuples that firmly maintain the occurrences and positions of the last state of the Tri-SASP.

Experiments were carried out on four real-world datasets, including air quality, central air-conditioning and oil electric submersible pump (ESP) sanding diagnosis and working diagnosis. The results show that: (1) contained in the state transition figure, both frequent Tri-SASPs and confident temporal association rules can provide new patterns and associations for expert decisions; (2) the SP and CP strategies bring us new knowledge about which states are more frequent and special, respectively; (3) Tri-SASPs obtained with the above two tri-partition strategies are greatly different; and (4) the horizontal and vertical algorithms are better for uniform and non-uniform datasets, respectively.

The contributions of this paper include the following:

- a new type of sequential pattern with a tri-partition state alphabet;
- two tri-partition strategies for a given state set;
- the strategy of pattern construction as well as the condition of tri-wildcard gap matching; and
- horizontal and vertical techniques for Tri-SASP support computing.

The rest of the paper is organized as follows: "Related Work" reviews previous works, including three-way decision and pattern discovery. "Tri-SASP and Temporal Association Analysis" proposes the concepts of the new Tri-SASP and temporal association rules. "Methods" presents the design of horizontal and vertical algorithms for discovering frequent Tri-SASPs. "Experiments" discusses the experimental results on scalability, readability and diversity. Finally, concluding remarks are presented in "Conclusion".

Related Work

In this section, we first review the theory and applications of 3WDs. Second, works regarding sequence pattern discovery on univariate, multivariate and sequence databases are presented.

Three-Way Decisions

In rough set theory [3], using an equivalence relation, the universe is partitioned into three disjoint regions. The positive, negative and boundary regions are the sets of objects that are definitely, definitely not and possibly members of the target set, respectively. However, this strict division often hinders its application in practice. Probabilistic rough set models such as decision-theoretical rough sets (DTRS) [49] and variable-precision rough sets [50] were introduced to address this issue. Among them, DTRS is a sound theory based on Bayesian decision making. The required parameters are systematically determined based on the costs of various decisions [2, 49].

The 3WDs have substantially advanced by becoming a divide-and-conquer methodology [2] rather than a concrete technique. One task of this methodology is to construct a tri-section or tri-partition of the universal set. The other task is to act upon objects in one or more regions by developing appropriate strategies. 3WDs are a class of effective methods and heuristics commonly used in human problem solving and information processing.

Recently, various theories have been inspired by the 3WD. Three-way formal concept analysis [51] is constructed using the three-way operators and their inverses. Three-way cognition computing [5, 7] focuses on concept learning via multi-granularity from the viewpoint of cognition. The three-way fuzzy sets method [9] constructs a three-way, three-valued or three-region approximation with a pair of thresholds on the fuzzy membership function. Three-way decision space [52] unifies decision measurement, decision conditions and evaluation functions. Sequential three-way decisions [13] is an iteration process that eventually leads to two-way decisions. Some generalized three-way decision models [53–55] are quite popular.

Additionally, there are abundant applications. Three-way recommender systems have been applied in both classification and regression [14] of user ratings. Three-way active learning [56] finds a tradeoff between the teacher cost of acquiring class labels and the misclassification cost. Three-way clustering [17] introduces a new strategy for overlapping clustering based on the DTRS model. Tri-partition neighborhood covering reduction [16, 57, 58] facilitates robust classification, and three-way spam filtering [21] reduces the email misclassification costs. Three-way face recognition [19] develops a sequential strategy to address the imbalanced misclassification cost and the problem of insufficient high-quality facial image information. Finally, an interesting type of pattern called a tri-pattern, which is based on the tri-partition alphabet, has been proposed for univariate time series, text and biological data analysis.

Sequence Pattern Discovery

Figure 2 shows the relationships among univariate sequences, multivariate sequences and sequence databases. For transaction data, the original association analysis problems and methods [29] were proposed to find interesting market patterns, for example, in beer and diaper sales at WalMart. For sequence databases, namely, transaction data with additional temporal information, the patterns in

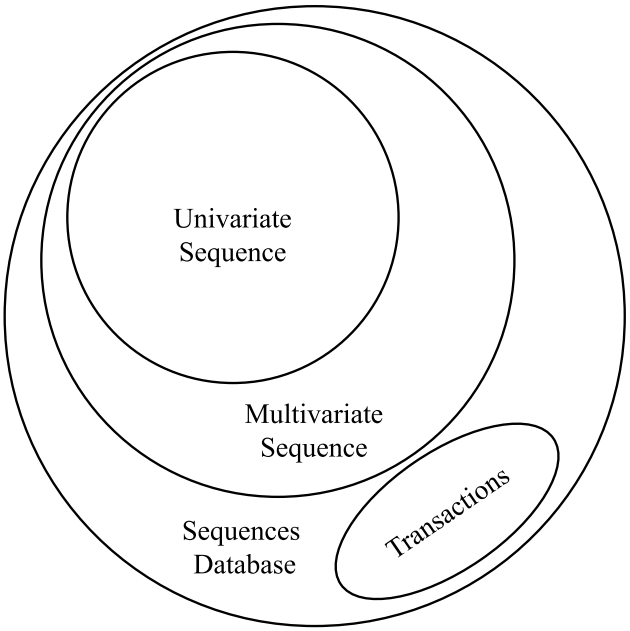


Fig. 2 The relationships among four types of data

the interactions of two systems at different times are significant. For example, different customers may buy various commodities over a period of time. Based on this type of dataset, sequential patterns such as {milk, bread} → {beer, diapers, toothbrush} → {towel} can be discovered. Because each customer has his own commodity sequence, the support of a given sequential pattern is defined as the fraction whose denominator and numerator are the number of all commodity sequences and those containing the pattern, respectively [24].

To handle the problem of frequent sequential pattern discovery on sequence databases, many works, such as AprioriAll [24] and GSP [28], were successively proposed by Agrawal and Srikant. The contributions of GSP compared to those of AprioriAll are the extension of time constraints, a sliding window and a taxonomy hierarchy technique. According to the search strategy, existing algorithms can be divided into breadth-first and depth-first algorithms. Breadth-first algorithms such as GSP generate all candidate k-sequences by using $(k-1)$-sequences. Depth-first algorithms such as Spade [30], Lapin [32], Spam [31] and PrefixSpam [33] were designed to recursively extend each sequence containing single items. For example, let the set of all items be {A, B, C}; a breadth-first algorithm will first consider the 1-sequences {A}, {B} and {C}. Then, the algorithm will consider the 2-sequences {A} → {A}, {A} → {B}, {A} → {C}, {A, B}, {A, C} and so on. A depth-first algorithm will explore potential sequential patterns following this order: {A}, {A, B}, {A, B, C}, {A} → {A}, {A} → {A, B} and so on.

According to the database structure, existing algorithms can also be divided into horizontal and vertical algorithms. The horizontal algorithms such as GSP scan the whole sequence database to calculate the support of candidate patterns. Another problem for GSP is that it may generate patterns that do not exist in the database. The third is that at every moment it must retain all frequent patterns. The vertical algorithms such as Spade utilize a vertical database representation indicating the itemsets in which each item appears in the sequence database. More details and examples can be found in [27]. A vertical representation can be regarded as a kind of compression of a horizontal one. The performance of algorithms based on vertical representations is better with a more sparse sequence database, and vice versa.

For MTSs, sequential patterns can be classified into along-first and across-first patterns. In terms of along-first methods such as DTA [42] and TARD [43], various sequence clusters are obtained by frequent subsequences mining and clustering successively within each univariate sequence. Then, all subsequence clusters are used to construct the expected pattern and those that are frequent will be found. For the across-first STAP, different frequent states are discovered. Each state contains at least one variable.

Then, all frequent states, treated as 1-sequential patterns, are used to construct longer patterns. In Table 1, one example of an along-first sequential pattern is $\{(a_1, \text{LL}), (a_1, \text{LLN}), (a_1, \text{LNN})\} \rightarrow \{(a_3, \text{NHH}), (a_3, \text{HHNL})\} \rightarrow \{(a_2, \text{NLLN}), (a_2, \text{LLNN})\}$. One example of an across-first pattern is $\{(a_1, \text{L}), (a_2, \text{H})\} \rightarrow \{(a_1, \text{N}), (a_2, \text{L}), (a_3, \text{L})\} \rightarrow \{(a_3, \text{H})\}$. Compared to patterns from sequence databases, those from MTSs are naturally different in two respects:

- Pattern length. Here, the length of a sequential pattern is the number of states, irrespective of the length of the states. For a sequence database, A sequence (A_1, A_2, ...,

Table 1 An example of a symbolic MTS

T (4/2019)	A		
	Temperature (a_1)	PM2.5 (a_2)	CO_2 (a_3)
20 (t_1)	L	H	N
21 (t_2)	L	N	L
22 (t_3)	N	L	L
23 (t_4)	N	L	N
24 (t_5)	H	N	H
25 (t_6)	H	N	H
26 (t_7)	L	L	N
27 (t_8)	N	H	L
28 (t_9)	H	N	L
29 (t_{10})	L	H	N

A_n) is called a k-sequence if it contains k items, or, in other words, if $k = |A_1| + |A_2| + ... + |A_n|$. The lengths of patterns ({A, B}) and ({A}, {B}) are 1 and 2 in our work, respectively. However, their lengths are both 2 in works based on sequence databases.

- Pattern support. For sequential patterns in sequence databases, the occurrences are counted according to the number of matched transactions. For the ones in MTSs, the occurrences are counted according to the number of matched position sequences. The a priori property is not available for this number. For example, the number of occurrences of B[0, 2] A is 3, namely, (2, 3), (2, 5) and (4, 5), which is larger than that of (B), namely, (2) and (4). Fortunately, there are three matching conditions, general, one-off and non-overlapping sequences, proposed below to ensure this property.

In terms of univariate sequences, the matching condition is the foundation of pattern discovery. There are three kinds of pattern matching conditions, i.e., non-overlapping [46], one-off [47] and general [48]. For example, given a univariate sequence $S = A^1B^2A^3B^4A^5$ of length 5, a periodic wildcard gap [0, 2] and a pattern $P = A[0, 2]B[0, 2]A$, the occurrences of this pattern are (1, 2, 3), (1, 2, 5), (1, 4, 5) and (3, 4, 5). The number of occurrences counted under the general condition is 4. Because each of the characters in the univariate sequence can be used only once under the one-off condition, there is only one occurrence, such as (1, 2, 3) for P. Under the non-overlapping condition, the number of occurrences of P is 2, namely, (1, 2, 3) and (3, 4, 5). In a word, the non-overlapping condition is less restrictive than the one-off condition, while it is more specific than the general one for pattern mining.

Tri-SASP and Temporal Association Analysis

In this section, we first discuss the formal description of an MTS. Second, two strategies for the state tri-partition alphabet are proposed. Third, the Tri-SASP is defined by the given tri-partition state alphabet. Fourth, two algorithms of frequent Tri-SASP discovery are proposed. Finally, some running examples of algorithms are presented to improve readability.

Data Model

With the development of sensor technology, the collection of time series data becomes convenient and reliable. In many applications, we are initially interested in the following data model:

Definition 1 An MTS is a quadruple

$$S = (T, A, V = \cup_{a \in A} V_a, f), \tag{1}$$

where $T = \{t_1, t_2, \ldots, t_n\}$ is a finite set of time points, $A = \{a_1, a_2, \ldots, a_m\}$ is a finite set of variables, V_a is the set of value ranges of variable a, and $f : T \times A \rightarrow V$ is the mapping function.

If the value ranges of all attributes are symbolic, we call it a symbolic MTS. We further assume that $t_{i+1} - t_i = t_i - t_{i-1}$ $(2 \leq i \leq n-1)$. For brevity, $\forall i \in [1, n], j \in [1, m], f(t_i, a_j) = f_{i,j}$ indicates the value of a_j at t_i. Moreover, let $f_{i,*} = \{(a_1, f_{i,1}), (a_2, f_{i,2}), \ldots, (a_m, f_{i,m})\}$.

Example 1 Table 1 shows an example of an MTS. Here, $T = \{t_1, \cdots, t_{10}\} = \{20/4/2019, ..., 29/4/2019\}$ indicates that there are daily data for 10 consecutive days. Each row represents a time point, and each column represents a kind of sensor. $A = \{a_1, a_2, a_3\}$ is the set of sensors, where a_1, a_2, and a_3 represent temperature, PM2.5 and CO_2, respectively. The value ranges are $V_{a_1} = V_{a_2} = V_{a_3} = \{H, L, N\}$, where H, N and L indicate high, normal and low, respectively. $f(t_1, a_2) = f_{1,2} = H$ indicates that the value of PM2.5 on 20/4/2019 is high. Moreover, $f_{1,*} = \{(a_1, L), (a_2, H), (a_3, N)\}$.

Strategies for the Tri-Partition State Alphabet

First, the definition of a state is fundamental for our three-way state alphabet. Generally, one can define the system state with some or all of the variables and their values in an MTS. A state is a set of attribute-value pairs and has the following two forms:

Definition 2 Given an MTS $S = (T, A, V = \cup_{a \in A} V_a, f)$, a state with the set form is

$$s = \{(a, v_a) | a \in A' \subseteq A, v_a \in V_a \subseteq V\}, \tag{2}$$

while a state with the logic form is

$$s = \wedge_{a \in A' \subseteq A, v_a \in V_a} (a, v_a), \tag{3}$$

subject to

$$\forall 1 \leq i \neq j \leq |s|, a_i \neq a_j.$$

For brevity, the set of all states is denoted as

$$\mathcal{L} = 2^{\{(a, v_a) | a \in \subseteq A, v_a \in V_a \subseteq V\}}.$$

When $s \supseteq s'$, we call s a *sub-state* of s'. Accordingly, s' is called a *super-state* of s. A state is a sub-state (and super-state) of itself.

Example 2 State $s_8 = \{(a_1, \mathrm{L}), (a_2, \mathrm{H})\}$ or equivalently $(a_1, \mathrm{L}) \wedge (a_2, \mathrm{H})$ is a valid state, but $\{(a_1, \mathrm{L}), (a_1, \mathrm{H})\}$ is not. When we know $s_8 \supseteq s_1 = \{(a_1, \mathrm{L})\}$, we have that s_8 is a sub-state of s_1. Moreover, we also say that s_8 is more specific than s_1 while s_1 is more general than s_8.

We are usually interested in whether or not a state matches an MTS at a time point. Therefore, we propose the concept of state matching as follows: Let $S = (T, A, V = \cup_{a \in A} V_a, f)$ be an MTS. State $s = \{(a, v_a) | a \in A' \subseteq A, v_a \in V_a\}$ matches S at $t \in T$ iff $s \subseteq f_{t,*}$, namely, $\forall\, a \in A'$, $f(t, a) = v_a$. We further denote the state matching as a function

$$m(S, t, s) = \begin{cases} 1, & \text{if } s \text{ matches } S \text{ at } t; \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

Hence, its support is given by

$$Sup(s, S) = \frac{\sum_{t \in T} m(S, t, s)}{|T|}. \tag{5}$$

$Sup(s, S)$ can be abbreviated as $Sup(s)$ when S is specified in context. For brevity, given a support threshold α, the set of frequent states is

$$\mathcal{L}^{\mathrm{gen}} = \{s \in \mathcal{L} | Sup(s) \geq \alpha\}.$$

Second, we propose two types of tri-partition state alphabet Σ, corresponding to two strategies. One is called support partition (SP), and the other is called compression partition (CP). Practically, Σ is specified by the user. No matter what kind of Σ we obtain, the following assumptions must be established:

- $\mathrm{POS} \cup \mathrm{BND} \cup \mathrm{NEG} = \Sigma$.
- $\mathrm{POS} \cap \mathrm{BND} = \emptyset$, $\mathrm{POS} \cap \mathrm{NEG} = \emptyset$ and $\mathrm{BND} \cap \mathrm{NEG} = \emptyset$.

Let α and β be two support thresholds. Given an MTS S and $s \in \Sigma$, the SP strategy is

$$\begin{cases} s \in \mathrm{POS}, & \text{if } Sup(s) \geq \beta; \\ s \in \mathrm{BND}, & \text{if } \alpha \leq Sup(s) < \beta; \\ s \in \mathrm{NEG}, & \text{otherwise.} \end{cases} \tag{6}$$

In this case, $\mathrm{POS} \cup \mathrm{BND} \cup \mathrm{NEG} = \mathcal{L}^{\mathrm{gen}} \cup \mathrm{NEG} = \mathcal{L}$. The states in NEG are too numerous to be presented. All states containing N, e.g., $(a_1, \mathrm{L}) \wedge (a_2, \mathrm{N})$, are ignored at first. This is because N indicates normal, which contains little or no information. Once we remove the value N, the number of candidate states will greatly decrease. In other words, all states containing N must belong to NEG.

Example 3 In accordance with Tables 1 and 2 shows an example of the SP strategy. State $s_9 = (a_1, \mathrm{L}) \wedge (a_3, \mathrm{L})$ belongs to NEG, because $Sup(s_9) = \frac{1}{10} < \alpha$. The significance levels of $(a_1, \mathrm{L}) \wedge (a_2, \mathrm{N}) \wedge (a_3, \mathrm{L})$ and s_9 are almost equivalent. When we do not consider N for each variable, the number of all states is only 26, where s_1, s_2 belong to POS, s_3 to s_8 belong to BND and the rest belong to NEG. In contrast, this number will be 63 when N is preserved.

The CP strategy is formally presented as follows: Initially, the set of frequent maximal states is

$$\mathcal{L}^{\max} = \{s \in \mathcal{L}^{\mathrm{gen}} | \nexists s' \in \mathcal{L}^{\mathrm{gen}}, s' \supseteq s\},$$

and the set of frequent closed states is

$$\mathcal{L}^{\mathrm{clo}} = \{s \in \mathcal{L}^{\mathrm{gen}} | \nexists s' \in \mathcal{L}^{\mathrm{gen}}, s' \supseteq s, Sup(s') = Sup(s)\}.$$

Consequently, the CP strategy is

$$\begin{cases} \mathrm{POS} &= \mathcal{L}^{\max}; \\ \mathrm{BND} &= \mathcal{L}^{\mathrm{clo}} - \mathcal{L}^{\max}; \\ \mathrm{NEG} &= \mathcal{L}^{\mathrm{gen}} - \mathcal{L}^{\mathrm{clo}}. \end{cases} \tag{7}$$

In this situation, the states in NEG are frequent but redundant. The union of the three regions is $\mathcal{L}^{\mathrm{gen}} \subseteq \mathcal{L}$.

Example 4 According to Tables 1 and 3 shows the state tri-partition results of the CP strategy. Similarly, all states containing N are ignored for brevity. In comparison with Table 2, s_6, belonging to BND, instead belongs to NEG. s_1, belonging to POS, instead belongs to BND. s_5, s_7 and s_8, belonging to BND, instead belong to POS.

Finally, Table 4 shows a comparison of the SP and CP strategies. Given a set of states, such as $\mathcal{L}$ and $\mathcal{L}^{\mathrm{gen}}$, the SP strategy needs two thresholds α and β, where β is additionally introduced. In contrast, the CP strategy needs only α, which maintains the simplicity of the inputs.

Table 2 An example of the SP strategy ($\Sigma = \mathcal{L}, \alpha = 0.2, \beta = 0.4$)

POS	BND	NEG
$s_1 = (a_1, \mathrm{L})(0.4)$	$s_3 = (a_1, \mathrm{H})(0.3)$	The infrequent
$s_2 = (a_3, \mathrm{L})(0.4)$	$s_4 = (a_2, \mathrm{H})(0.3)$	
	$s_5 = (a_2, \mathrm{L})(0.3)$	
	$s_6 = (a_3, \mathrm{H})(0.2)$	
	$s_7 = (a_1, \mathrm{H}) \wedge (a_3, \mathrm{H})(0.2)$	
	$s_8 = (a_1, \mathrm{L}) \wedge (a_2, \mathrm{H})(0.2)$	

Table 3 An example of the CP strategy ($\Sigma = \mathcal{L}^{\text{gen}}, \alpha = 0.2$)

POS	BND	NEG
$s_5 = (a_2, \text{L})(0.3)$	$s_3 = (a_1, \text{H})(0.3)$	$s_6 = (a_3, \text{H})(0.2)$
$s_2 = (a_3, \text{L})(0.4)$	$s_1 = (a_1, \text{L})(0.4)$	
$s_7 = (a_1, \text{H}) \wedge (a_3, \text{H})(0.2)$	$s_4 = (a_2, \text{H})(0.3)$	
$s_8 = (a_1, \text{L}) \wedge (a_2, \text{H})(0.2)$		

Pattern and Rule

First, in the previous work [44], $\forall s \in \mathcal{L}^{\text{gen}}$ is a component of STAP, while in our work, the components of Tri-SASP come from the POS and BND of the three-way state alphabet Σ.

Definition 3 Given a three-way state alphabet $\Sigma = \text{POS} \cup \text{BND} \cup \text{NEG}$ and a tri-wildcard gap $\Delta = [\underline{G}, \overline{G}]$, $0 \le \underline{G} \le \overline{G}$, a Tri-SASP is

$$P = s_1 \Delta s_2 \ldots \Delta s_k = s_1 [\underline{G}, \overline{G}] s_2 \ldots [\underline{G}, \overline{G}] s_k, \tag{8}$$

where k is the length of P; $\forall i \in [1, k], s_i \in \text{POS} \cup \text{BND}$; $\underline{G}$ and $\overline{G}$ are called the minimal and maximal *gap constraints* of P, respectively; and no states of POS may occur in the interval Δ.

Example 5 With Table 2, pattern $s_2 \Delta s_3 \Delta s_6$ is a Tri-SASP. With Table 3, this pattern is not a Tri-SASP, because the state s_6 belongs to NEG region under the CP strategy. In other words, all patterns containing state s_6 are not Tri-SASPs using Table 3.

Figure 3 shows the tri-partition and actions for a given state set. States in POS can only be used to construct a Tri-SASP. States in NEG can only be matched by a tri-wildcard gap. States in BND can be used to either construct a Tri-SASP or be matched by a tri-wildcard gap.

Second, we present the pattern matching rules for Tri-SASPs. Naturally, a Tri-SASP $P = s_1 \Delta s_2 \ldots \Delta s_k$ always occurs in an MTS S with a position sequence

$$I = i_1 i_2 \ldots i_k,$$

where $\forall j \in [1, k-1]$, $i_1 \in [1, |T|]$ and $0 \le \underline{G} \le i_{j+1} - i_j - 1 \le \overline{G}$. Hence, we can present the formal description of the matching rule as follows:

Table 4 Comparison of the SP and CP strategies

Name	Strategy	Inputs	POS ∪ BND ∪ NEG
SP	Eq. (6)	α, β	$\mathcal{L}$ (POS ∪ BND = $\mathcal{L}^{\text{gen}}$)
CP	Eq. (7)	α	$\mathcal{L}^{\text{gen}}$

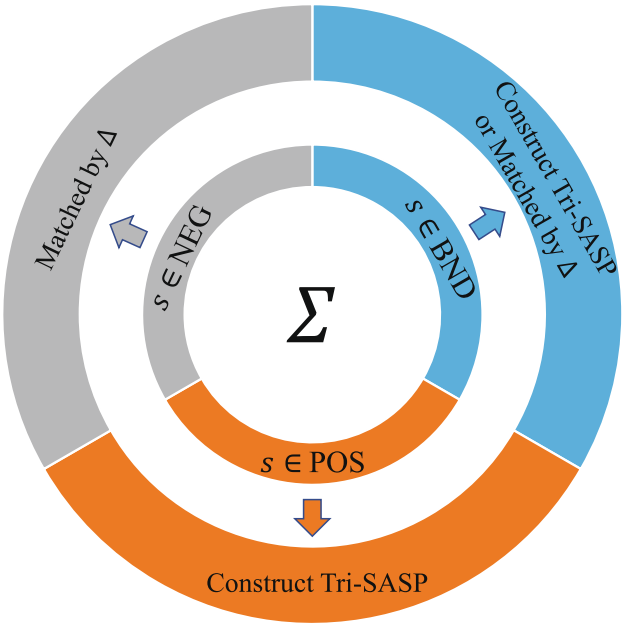


Fig. 3 Tri-partition and actions of the state alphabet

Given $S = (T, A, V, f)$, $\Sigma = \text{POS} \cup \text{BND} \cup \text{NEG}$, $\Delta = [\underline{G}, \overline{G}]$, $P = s_1 \Delta s_2 \ldots s_k$ and $I = i_1 i_2 \ldots i_k$ as a position sequence, the matching of P on S at I is a Boolean function

$$m(S, P, I) = \begin{cases} 1, & \text{if } \forall 1 \le j \le k, s_j \text{ matches } S \text{ at } t_{i_j}; \\ 0, & \text{otherwise,} \end{cases} \tag{9}$$

where $s_j \in \text{POS} \cup \text{BND}, \forall j \in [1, k-1], q \in [i_j, i_{j+1}]$ and $\forall s \in \text{POS}$, $s \notin f_{t_q,*}$ or $m(t_q, s) = 0$.

Example 6 With Tables 1 and 2, let $\Delta = [0, 6]$, and let $P = s_2[0, 6]s_8 = (a_3, \text{L})[0, 6](a_1, \text{L}) \wedge (a_2, \text{H})$ be a Tri-SASP. The set of positions matching (a_3, L) is $\{t_2, t_3, t_8, t_9\}$. The set matching $(a_1, \text{L}) \wedge (a_2, \text{H})$ is $\{t_1, t_{10}\}$. Therefore, the candidate position sequences matching P are $t_3 t_{10}$, $t_8 t_{10}$ and $t_9 t_{10}$. The position sequence $t_2 t_{10}$ does not match P because the diversity between positions t_2 and t_{10} exceeds $\Delta = [0, 6]$, i.e., $10 - 2 - 1 = 7 \notin [0, 6]$. The states of POS are $s_1 = (a_1, \text{L})$ and $s_2 = (a_3, \text{L})$, whose matching position sets are $\{t_1, t_2, t_7, t_{10}\}$ and $\{t_2, t_3, t_8, t_9\}$, respectively. Because positions t_2, t_7 and t_8 occur between t_3 and t_{10}, position sequence $t_3 t_{10}$ does not match P. The only correct position sequence matching P is $t_9 t_{10}$. Similarly, let $P' = s_8[0, 6]s_2$; the candidate position sequences matching P' are $t_1 t_2$, $t_1 t_3$ and $t_1 t_8$, and the correct one is $t_1 t_2$.

Additionally, in the most ideal case, there are $\overline{G} - \underline{G} + 1$ types of i_{j+1} for each i_j. Moreover, there are $|T|$ total time points in MTS S. Hence, the set of total position sequences of P in S is denoted as $\mathcal{I}$, and its cardinality is

Table 5 Comparisons among Tri-SASPs and existing patterns

Name	Partition strategy	Alphabet (Σ)	Component	Gap matching	Data model
Tri-SASP	Tri-partition	{POS, BND, NEG}	POS ∪ BND	BND ∪ NEG	Multivariate ($\|A\| > 1$)
STAP	None	$\mathcal{L}^{\text{gen}}$	$\mathcal{L}^{\text{gen}}$	$\mathcal{L}^{\text{gen}}$	Multivariate ($\|A\| > 1$)
Tri-pattern	Tri-partition	{POS, BND, NEG}	POS ∪ BND	BND ∪ NEG	Univariate ($\|A\| = 1$)
Weak-wildcard pattern [59]	Binary partition	{POS, NEG}	POS	NEG	Univariate ($\|A\| = 1$)

$$|\mathcal{I}| = |T| \times (\overline{G} - \underline{G} + 1)^{k-1}.$$

Example 7 With Tables 1 and 2, let $\Delta = [0, 6]$ and let there be a Tri-SASP $P = s_2[0, 6]s_8$ of length 2; the cardinality of its total position sequences is $10 \times (6 - 0 + 1)^{2-1} = 70$.

Consequently, we find that the support of Tri-SASP P is

$$Sup(S, P) = \frac{|\mathcal{I}_P = \{I \in \mathcal{I} | m(S, P, I) = 1\}|}{|\mathcal{I}|}. \tag{10}$$

Example 8 With Tables 1 and 2, let $P = s_2[0, 6]s_8$, $\mathcal{I}_P = \{t_9 t_{10}\}, |\mathcal{I}_P| = 1$; hence, the support of P is $\frac{1}{70}$.

Then, let $\gamma \in [0, 1]$ be the threshold specified by users; the set of frequent Tri-SASPs is

$$\mathcal{P} = \{P | Sup(S, P) \geq \gamma\}.$$

Third, we present the pattern growth operations of Tri-SASP. For this purpose, the definitions of sub-patterns and super-patterns are necessary.

Definition 4 Given two Tri-SASPs $P_1 = s_1 \Delta s_2 \ldots s_m$ and $P_2 = s'_1 \Delta s'_2 \ldots s'_n$, $m \leq n$, P_1 is called a sub-pattern of P_2 iff

$$\exists \text{ a sequence } i_1 i_2 \ldots i_m, \text{ s.t. } \forall j \in [1, m], s_j \subseteq s'_{i_j}, \tag{11}$$

where $\forall j \in [1, m-1], i_{j+1} = i_j + 1$. For brevity, this relationship is denoted as $P_1 \sqsubseteq P_2$.

Example 9 With Tables 1 and 2, let $P_1 = s_1 \Delta\ s_2 \Delta\ s_3 \Delta s_4$, $P_2 = s_3 \Delta s_4$, and $P_3 = s_7 \Delta s_8$. We have $P_2 \sqsubseteq P_1$ and $P_2 \sqsubseteq P_3$. For the latter, this is because $s_3 \subseteq s_7$ and $s_4 \subseteq s_8$.

In contrast, given a Tri-SASP $P = s_1 \Delta s_2 \Delta \ldots \Delta s_k$ of length k, the number of its sub-patterns is

$$(\sum_{l=1}^{k} \sum_{i=1}^{k-l+1} \prod_{j=1}^{l} (2^{|s_{i+j-1}|} - 1)) - 1.$$

Example 10 With Tables 1 and 2, let $P = s_1 \Delta s_2 \Delta s_3 = (a_1, \text{L}) \wedge (a_2, \text{H}) \wedge (a_3, \text{N}) \Delta (a1, \text{N}) \wedge (a_3, \text{L}) \Delta (a_2, \text{N}) \wedge (a_3, \text{H})$. The number of all sub-patterns of P is $(2^3 - 1 + 2^2 - 1 + 2^2 - 1) + ((2^3 - 1) \times (2^2 - 1) + (2^2 - 1) \times (2^2 - 1)) + (2^3 - 1) \times (2^2 - 1) \times (2^2 - 1) - 1 = 13 + (21 + 9) + 42 - 1 = 84$.

Next, we can present the rule of the pattern growth operation as follows:

Definition 5 The pattern growth operation between two Tri-SASPs $P_1 = s_1 \Delta s_2 \ldots s_m$ and $P_2 = s'_1 \Delta s'_2 \ldots s'_n$ is

$$P_1 \otimes P_2 = s_1 \Delta s_2 \ldots s_m \Delta s'_1 \Delta s'_2 \ldots s'_n. \tag{12}$$

This growth operation is subject to

(1) $P_1 \otimes P_2 \neq P_2 \otimes P_1$;
(2) $P_1 \otimes P_2 \otimes P_3 = P_1 \otimes (P_2 \otimes P_3)$;
(3) if $P_3 = P_1 \otimes P_2$, P_3 is called a Tri-SASP, $P_1, P_2 \sqsubseteq P_3$; and
(4) if $\mathcal{P}$ and $\mathcal{P}'$ are two sets of Tri-SASPs, then

$$\mathcal{P} \otimes \mathcal{P}' = \{P \otimes P' | P \in \mathcal{P}, P' \in \mathcal{P}'\}.$$

Example 11 With Tables 1 and 2, let $P_1 = s_2, P_2 = s_3, P_3 = s_4$ and $P_4 = s_3 \Delta s_4$. First, $P_1 \otimes P_2 = s_2 \Delta s_3 \neq s_3 \Delta s_2 = P_2 \otimes P_1$. Second, $P_1 \otimes P_2 \otimes P_3 = s_2 \Delta s_3 \Delta s_4 = P_1 \otimes (P_2 \otimes P_3)$. Third, $P_4 = s_3 \Delta s_4 = P_2 \otimes P_3$, and with Definition 4, we know $P_2, P_3 \sqsubseteq P_4$. Specifically, the result of $s_1 \Delta s_2 \otimes s_2 \Delta s_3$ is $s_1 \Delta s_2 \Delta s_2 \Delta s_3$ instead of $s_1 \Delta s_2 \Delta s_3$. The latter can be constructed by $s_1 \Delta s_2 \otimes s_3$, $s_1 \otimes s_2 \Delta s_3$, or $s_1 \otimes s_2 \otimes s_3$.

Table 5 shows the comparisons among the Tri-SASP, STAP, tri-pattern and weak-wildcard pattern. There are five aspects of these patterns, as follows:

- **Partition strategy.** Except for STAP, whose alphabet is not partitioned, all of the patterns introduce a partition strategy. Tri-SASPs and tri-patterns are based on a tri-partition alphabet. Weak-wildcard patterns are based on a binary partition alphabet.
- **Alphabet.** The components of the Tri-SASP and STAP are called states. However, the state alphabet is tri-partitioned into POS, BND and NEG regions with user-specified strategies. More details can be found in Table 4. For the tri-pattern, there are strong, medium and weak elements, while for the weak-wildcard pattern, we have only strong and weak ones. A tri-pattern is a generalization

of the weak-wildcard pattern, while a Tri-SASP is also a generalization of the tri-pattern obtained by extending a single attribute to multiple attributes.

- **Component construction.** With a given alphabet, different sequential patterns have various construction strategies. For the Tri-SASP, the components come from POS ∪ BND. For STAP, its components belong to $\mathcal{L}^{\text{gen}}$, which is the set of frequent states. For a tri-pattern, its components also come from POS ∪ BND. For a weak-wildcard pattern, the components belong only to POS.
- **Gap matching condition.** These four sequential patterns are all based on the general one. The differences are that (1) the Tri-SASP, tri-pattern and weak-wildcard pattern do not permit the elements of POS to occur in the wildcard gap between each pair of adjacent components; (2) the wildcard gaps of the Tri-SASP and tri-pattern allow the elements of BND ∪ NEG to occur in itself; (3) the wildcard gap of STAP allows all frequent states to occur in it; and (4) the wildcard gap of the weak-wildcard pattern permits only the elements of NEG to occur.
- **Data model.** The Tri-SASP and STAP are obtained for an MTS, whose number of attributes is larger than 1. The Tri-pattern and weak-wildcard pattern are discovered in a univariate time series, whose attribute number is merely 1.

Finally, a new type of temporal association rule can be obtained with our Tri-SASP.

Definition 6 Given a pattern $P = s_1 \Delta s_2 \dots s_k$ and a dividing position index $i \in [2, k]$, let $P_{[*,i)} = s_1 \Delta\ s_2 \dots s_{i-1}$ and $P_{[i,*]} = s_i \Delta\ s_{i+1} \dots s_k$. The type of rule is

$$r : P_{[*,i)} \overset{\Delta}{\Longrightarrow} P_{[i,*]}, \tag{13}$$

where $\overset{\Delta}{\Longrightarrow}$ can be abbreviated as $\Longrightarrow$ when Δ is specified. The confidence of r is

$$c(r) = \frac{Sup(P_{[.,i)} \otimes P_{[i,*]})}{Sup(P_{[*,i)})} = \frac{|\mathcal{I}_P|}{|\mathcal{I}_{P_{[*,i)}}| \times |\Delta|^{k-i+1}}. \tag{14}$$

Example 12 With Tables 1 and 2, letting $P = s_2[0, 6]s_8$, we find that $\mathcal{I}_P = 1$ and $Sup(P) = \frac{1}{70}$. There is only one rule, $r : s_2 \Rightarrow s_8$, that we can obtain. Hence, the confidence of r is

$$c(r) = \frac{|\mathcal{I}_P|}{|\mathcal{I}_{s_2}| \times (6 - 0 + 1)^{2-2+1}} = \frac{1}{4 \times 7} = \frac{1}{28} \neq \frac{1}{70}.$$

When we obtain $Sup(P) = \frac{1}{70}$, we can explain it as "the probability of P happening is $\frac{1}{70}$." When we obtain $c(r) = \frac{1}{28}$, we can explain it as "if state s_1 happens, the probability of state s_8 happening within a 0 to 6 time delay is $\frac{1}{28}$."

Due to the temporal property of MTSs, a Tri-SASP P of length k can generate $k - 1$ rules at most. There is a confidence threshold λ for r such that if the confidence of r is no less than λ, we say that r is a confident rule. Consequently, the set of all confident rules can be formally described as

$$R = \{r | c(r) \geq \lambda\}.$$

Generally, a rule $r : P_1 \overset{\Delta}{\Longrightarrow} P_2$ is read as "if sequential pattern P_1 happens, then the probability of P_2 happening within a delay of $\underline{G}$ to $\overline{G}$ is $c(r)$". When $n = 2$, this rule can be read as "if state s_1 happens, then the probability of s_2 happening within a delay of $\underline{G}$ to $\overline{G}$ is $c(r)$." Compared to classical association rules in transaction databases, our Tri-SASP rules have three major differences:

- There is a time delay $\Delta = [\underline{G}, \overline{G}]$ between the anterior and posterior parts.
- Both the anterior and posterior parts of the rule are Tri-SASPs.
- Each state of a Tri-SASP belongs to POS or BND.

Problem Statement

Problem 1 Frequent Tri-SASP discovery.

Input: $S = (T, A, V, f)$, $\Delta = [\underline{G}, \overline{G}]$, Σ = POS ∪ BND ∪ NEG, α and γ.

Output: $\mathcal{P} = \{P | Sup(s \in P, S) \geq \alpha, Sup(P, S) \geq \gamma\}$.

Σ = POS ∪ BND ∪ NEG can be obtained through the process of frequent state mining [29] and tri-partitions. In the worst situation ($\alpha = 0$), the number of all possible frequent states is

$$\mathcal{M} = (\prod_{a \in A} (|V_a| + 1)) - 1.$$

Given a Tri-SASP $P = s_1 \Delta s_2 \dots s_k$, for each $j \in [1, k]$ we have up to $\mathcal{M}$ types of state for s_j. Hence, there are $\mathcal{M}^k$ types of P in total. Therefore, the time complexity of obtaining all P with length k is $|\mathcal{I}| \times \mathcal{M}^k$. Therefore, the time complexity of finding all possible P with lengths ranging from 1 to $|T| = n$ is

$$\sum_{k=1}^{n} |\mathcal{I}| \times \mathcal{M}^k.$$

This equation shows a high time complexity, so an important proposition is made to greatly reduce the search space.

Proposition 1 *If two Tri-SASPs are subject to* $P \sqsubseteq P'$,

$$Sup(P') \leq Sup(P).$$

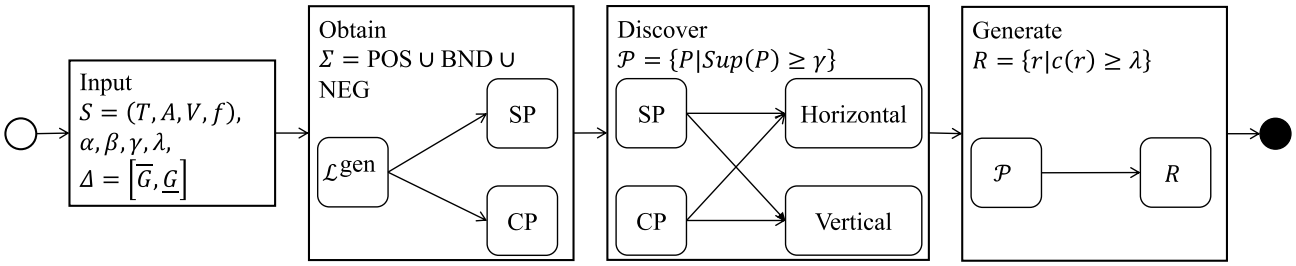


Fig. 4 The framework of frequent Tri-SASP discovery

Proof We only need to consider the condition where P contains one fewer component than P'. Formally, let $P = s_1 \Delta s_2 \ldots s_{k-1}$ and $P' = s_1 \Delta s_2 \ldots s_k$. Therefore, we have

$$|\mathcal{I}'| = |\mathcal{I}| \times |\Delta|.$$

On the other hand, each match of P corresponds to at most $|\Delta|$ matches of P', i.e.,

$$|\mathcal{I}'_P| \leq |\mathcal{I}_P| \times |\Delta|.$$

Therefore, we have

$$Sup(P') = \frac{|\mathcal{I}'_P|}{|\mathcal{I}'|} \leq \frac{|\mathcal{I}_P|}{|\mathcal{I}|} = Sup(P).$$

Problem 2 Confident temporal association rule discovery.
Input: $\mathcal{P} = \{P | Sup(s \in P, S) \geq \alpha, Sup(P, S) \geq \gamma\}$ and λ.
Output: $R = \{r | c(r) \geq \lambda\}$.

The time complexity of Problem 2 is much less than that of Problem 1. This is because the most time-consuming process, namely, frequent Tri-SASP discovery, has been solved in the later problem. Hence, the cost of generating a rule is constant. In the worst situation, there are a total of $\sum_{k=1}^{n} \mathcal{M}^k$ Tri-SASPs (k is the length of a Tri-SASP). Therefore, we have up to

$$\sum_{k=1}^{n} (\mathcal{M}^k \times (k-1))$$

temporal association rules to be generated.

Additionally, the search space of confident rules can be reduced by the following proposition: Given a Tri-SASP $P = s_1 \Delta s_2 \ldots s_k$ and two indices $2 \leq i < j \leq k$, we can obtain $P_1 = s_1 \Delta s_2 \ldots s_{i-1}$, $P_2 = s_i \Delta s_{i+1} \ldots s_k$, $P'_1 = s_1 \Delta s_2 \ldots s_{j-1}$, $P'_2 = s_j \Delta s_{j+1} \ldots s_k$; then, we have

Proposition 2 *The confidence of $r : P_1 \rightarrow P_2$ is no larger than that of $r' : P'_1 \rightarrow P'_2$, namely*

$$c(r) \leq c(r'). \tag{15}$$

Proof With Eq. (13), we know $c(r) = \frac{Sup(P_1 \otimes P_2)}{Sup(P_1)}$ and $c(r') = \frac{Sup(P'_1 \otimes P'_2)}{Sup(P'_1)}$. First, $P_1 \otimes P_2 = P'_1 \otimes P'_2 = P$. Second, because $i < j$, we have $P_1 \sqsubseteq P'_1$. Therefore, we have $c(r) \leq c(r')$.

Methods

In this section, we first present the framework of frequent Tri-SASP discovery. With the SP and CP strategies (see Eq. (6) and Eq. (7)), Σ = POS ∪ BND ∪ NEG can be initialized. Second, we discuss the design of the proposed horizontal and vertical methods. Third, we propose a technique for confident temporal association rule discovery. Finally, a running example for all the above algorithms is presented.

Frequent Tri-SASP Discovery

Figure 4 shows the general process of frequent Tri-SASP discovery. This framework consists of three major phases: (1) obtaining the three-way state alphabet; (2) discovering frequent Tri-SASPs; and (3) generating confident temporal association rules. In the first phase, the set of frequent states $\mathcal{L}^{\text{gen}}$ can be initially obtained by the frequent itemsets mining techniques [29]. The threshold α is used here.

With the specified SP (see Eq. (6)) or CP (see Eq. (7)) strategies, the three-way state alphabet Σ = POS ∪ BND ∪ NEG can be obtained. For the SP strategy, we need to scan $\mathcal{L}^{\text{gen}}$ only once to obtain regions POS and BND. There are two reasons why it is unnecessary to compute and store NEG: (1) computing NEG is too time consuming to be considered; and (2) even if we obtain NEG, the states in this region still cannot be used to construct patterns. Hence, if we chose the SP strategy, NEG indeed exists in Σ but is actually empty for the algorithm.

For the CP strategy, the first step is to compute $\mathcal{L}^{\text{clo}}$ and $\mathcal{L}^{\max}$ with $\mathcal{L}^{\text{gen}}$ and α. According to Eq. (7), the positive region is just $\mathcal{L}^{\max}$, which is the most specific. The boundary region is the difference set of

the closed state set and maximal state set, namely, $\mathcal{L}^{\text{clo}} - \mathcal{L}^{\max} = \{s | s \in \mathcal{L}^{\text{clo}}, s \notin \mathcal{L}^{\max}\}$. The negative region is the difference set of the general state set and closed state set, namely, $\mathcal{L}^{\text{gen}} - \mathcal{L}^{\text{clo}} = \{s | s \in \mathcal{L}^{\text{gen}}, s \notin \mathcal{L}^{\text{clo}}\}$.

Algorithm 1 shows the process of frequent Tri-SASP discovery. Line 2 shows that only states in NEG cannot be the components of patterns. All frequent patterns are searched in a width-first/levelwise manner. In other words, longer frequent Tri-SASPs will be checked iff all of the shorter frequent Tri-SASPs are obtained.

Because the process of support computing is the most time consuming, a pre-pruning technique is proposed on Lines 8-10. Here, P.tail is actually the maximal suffix of P; namely, if $P = s_1 \Delta s_2 \dots s_k$, P.tail will be $s_2 \Delta s_3 \dots s_k (k \geq 2)$. According to the a priori/down-closure property (see Proposition 1), if P.tail is not frequent, P must be infrequent. In terms of Line 11, we propose two type of techniques to compute the support of the given Tri-SASP. One of them, called horizontal support computing (H-*Sup*), is presented in Algorithm 2. The other, called vertical support computing (V-*Sup*), is presented in Algorithm 3.

Algorithm 2 presents the process of the horizontal support computing technique. For each given Tri-SASP $P = s_1 \Delta s_2 \dots s_k$, its occurrence is counted by rescanning the MTS $S = (T, A, V, f)$. On Line 4, the number of position sequences matching s_1 and beginning with $t_i \in T$ is obtained. The function "Count(.)" is proposed to recursively determine the number. Once we locate s_i at t_j, the occurrence of s_{i+1} can be determined directly. Namely, s_{i+1} can occur only between $t_{j+\underline{G}+1}$ and $t_{j+\overline{G}+1}$.

For more details, parameters $\mu \in [1, |T| - k \times (\underline{G} + 1)]$ and $\nu \in [2, k]$ indicate the index of the time point and that of the state from P. Most importantly, Lines 16-20 implement the matching rule of Tri-SASP. In short, if there is a state from a POS region located between the occurrence positions of $s_{\nu-1}$ and s_ν, the current occurrence cannot be counted. In other words, only states in POS cannot be ignored.

The greatest difference between vertical and horizontal support computing approaches is the organization of the MTS. In terms of classic association analysis, with items and their transaction IDs (TIDs), the support of an itemset can be obtained by computing the intersection of the TID sets of all corresponding items. In this way, vertical-based algorithms do not need to re-scan the original dataset. Therefore, we can also maintain Tri-SASPs and their position sequences to obtain frequent Tri-SASPs. First, we define a novel vertical structure to store the information of the position sequences.

Algorithm 1 Discovering frequent Tri-SASPs.

Input: An MTS $S = (T, A, V, f)$, γ and $\Sigma = \text{POS} \cup \text{BND} \cup \text{NEG}$;
Output: $\mathcal{P} = \{P | Sup(P, S) \geq \gamma\}$;
Method: HD-triSTAP.

```
1: P ← ∅;
2: 1-P ← POS ∪ BND;
3: for (i ← 2; i < |T|; i++) do
4:    i-P ← ∅;
5:    for (each P' ∈ (i−1)-P) do
6:       for (each P'' ∈ 1-P) do
7:          P ← P' ⊗ P'';// See Eq. (12).
8:          if (P.tail ∉ (i−1)-P) then
9:             continue;// Go to Line 6.
10:         end if
11:         if (H-Sup(P) ≥ γ or V-Sup(P', P'') ≥ γ) then
12:            i-P ← i-P ∪ {P};
13:         end if
14:      end for
15:   end for
16:   if (i-P = ∅) then
17:      break;// Terminate the loop from Line 3.
18:   end if
19:   P ← P ∪ i-P;
20: end for
21: return ℙ^gen;
```

Algorithm 2 Computing support in a horizontal way.

Input: $S = (T, A, V, f)$, $P = s_1 \Delta s_2 \dots s_k$, $\Delta = [\underline{G}, \overline{G}]$;
Output: $Sup(P)$;
Method: H-*Sup*.

```
1: occ ← 0; // Initialize the occurrences
2: for (i ← 1; i ≤ |T| − k × (G+1); i++) do
3:    if (s_1 matches t_i) then
4:       occ ← occ+Count(S, P, G, Ḡ, i, 2);// See Lines 9-24.
5:    end if
6: end for
7: return Sup(P) ← occ / (|T| × (Ḡ − G + 1)^{k−1});
8: // A new function.
9: Count(S, P, G, Ḡ, μ, ν);// μ is the index of the time point,
   ν is the index of the state.
10: if (ν = k) then
11:    return 1; // All states of P have been matched.
12: end if
13: c ← 0; // Initialize the counter
14: for (i ← μ + G + 1; i ≤ μ + Ḡ and μ + i ≤ |T|; i++) do
15:    if (s_ν matches t_i) then
16:       for (j ← μ + G + 1; j < i; j++) do
17:          if (∃s ∈ POS, s matches t_j) then
18:             return c;
19:          end if
20:       end for
21:       c ← c + Count(S, P, G, Ḡ, μ + i, ν + 1);
22:    end if
23: end for
24: return c;
```

Definition 7 Given an MTS S and a Tri-SASP $P = s_1 \Delta s_2 \dots s_k$, the vertical structure of position sequences $\mathcal{I}_P$ is a set of tuples

$$P.\text{TPList} = \{(C(i_k), i_k) | i_k \in I \in \mathcal{I}_P\}, \tag{16}$$

where $C(i_k) = |\{I | I \text{ ends with } i_k\}|$, namely, the number of position sequences ending with i_k.

Because this vertical structure focuses on the information of the termination of position sequences, TPList represents the list of terminal positions.

Consequently, we are interested in the interaction of two TPLists. Given two Tri-SASPs $P = s_1 \Delta s_2 \dots s_k$ $(k \geq 1)$ and $P' = s'_1$, their TPLists are $P.\text{TPList} = \{(C(i_k), i_k) | i_k \in I \in \mathcal{I}_P\}$ and $P'.\text{TPList} = \{(1, i'_1) | i'_1 \in I' \in \mathcal{I}_{P'}\}$, respectively. Hence, the TPList of $P'' = P \otimes P' = s_1 \Delta s_2 \dots s_k \Delta s'_1$ is

$$P''.\text{TPList} = \{(C(i_k), i'_1) | i'_1 - i_k \in [\underline{G} + 1, \overline{G} + 1]\}. \tag{17}$$

Note that the second factor of $\otimes$ (P') must contain a single state. Moreover, when $|P| = 1$, a Tri-SASP is actually a state. In this way, the last position becomes the first in order to guarantee the correctness of support computing. Otherwise, if the length of P' is no less than 2, we must indicate the position of the first state, which would break the structure of the TPList.

Second, Algorithm 3 is proposed according to Eqs. (16) and (17). Lines 4-7 implement the Tri-SASP matching rule with a vertical TPList. There is a state in POS, and one of its terminal positions is between the last position of the first Tri-SASP (P') and that of the second (P'').

Finally, the space complexity of the vertical technique (V-*Sup*) is required because the TPList brings additional memory consumption. In the worst situation, the maximal space of the TPList of a Tri-SASP is $2n$. Therefore, the space complexity of Algorithm 3 is only

Algorithm 3 Support with the TPList technique.

Input: $P'.\text{TPList}$, $P''.\text{TPList}$, $\Delta = [\underline{G}, \overline{G}]$;
Output: $Sup(\mathcal{P})$;
Method: V-*Sup*.

1: $\mathcal{P}.\text{TPList} \leftarrow \emptyset$;
2: **for** (each element $(C(i'_{k-1}), i'_{k-1}) \in P'.\text{TPList}$) **do**
3: **for** (each element $(1, i''_1) \in P''.\text{TPList}$) **do**
4: **if** $(i''_1 \in [i'_{k-1} + \underline{G} + 1, i'_{k-1} + \overline{G} + 1])$ **then**
5: **if** $(\exists s \in \text{POS}$ and $(1, i^s_1) \in s.\text{TPList}$, s.t. $i^s_1 \in [i'_{k-1}, i''_1])$ **then**
6: continue;// Go to Line 3.
7: **end if**
8: $C(i_k) \leftarrow C(i'_{k-1})$ and $i_k \leftarrow i''_1$;// Eq. (17).
9: $P_k.\text{TPList} \leftarrow P_k.\text{TPList} \cup \{(C(i_k), i_k)\}$;
10: **end if**
11: **end for**
12: **end for**
13: **return** $Sup(P) \leftarrow \frac{\sum_{j=1}^{|P.\text{TPList}|} C(i^j_k)}{|T| \times (\overline{G} - \underline{G} + 1)^{k-1}}$;

$$O(\sum_{k=1}^{|T|} 2|T|) = O(|T| \times (|T| + 1)) = O(n^2).$$

Temporal Association Rule Discovery

Given a Tri-SASP of length k, we can obtain $k - 1$ confident rules at most $(k \geq 2)$. Algorithm 4 shows the process of confident temporal association rule discovery. When $k = 2$, we have rules such as $s_1 \Rightarrow s_2$. However, there is also a kind of Tri-SASP like $s_1 \Delta s_2$. They are similar but completely different in terms of both the semantics and metrics (see Eq. (10) vs. Eq. (13)).

Algorithm 4 Confident temporal association rule discovery.

Input: $\mathcal{P} = \{P | Sup(P) \geq \gamma\}, \lambda$;
Output: $R = \{r | c(r) \geq \lambda\}$;
Method: r.

1: $R = \emptyset$;
2: **for** (each $P = s_1 \Delta s_2 \dots s_k \in \mathcal{P}$ and $k \geq 2$) **do**
3: **for** $(i = k; i > 1; i - -)$ **do**
4: $P_1 = s_1 \Delta s_2 \dots s_{i-1}$ and $P_2 = s_i \Delta s_{i+1} \dots s_k$;
5: **if** $(c(r) = \frac{Sup(P_1 \otimes P_2)}{Sup(P_1)} \geq \lambda)$ **then**
6: $R = R \cup \{P_1 \rightarrow P_2\}$;
7: **end if**
8: **end for**
9: **end for**
10: **return** R;

Running Example

With Tables 1 and 3, we present the process of frequent Tri-SASP discovery. For brevity, we discuss only the process of mining 1-$\mathcal{P}$ and 2-$\mathcal{P}$ for horizontal and vertical support computing techniques. Initially, we let $\gamma = 0.02$, $\alpha = 0.2$ and $\Delta = [0, 6]$. Because $\gamma < \alpha$, 1-$\mathcal{P}$ = POS $\cup$ BND.

Horizontal Computation

First, the set of candidate Tri-SASPs of length 2, namely, 2-$\mathcal{C}$, is generated. On the basis of Table 3, there are $7 \times 7 = 49$ candidate Tri-SASPs, such as $P = s_2 \Delta s_8$.

Second, the algorithm scans Table 1 from t_1 to t_{10} to search the time positions matching state s_2. t_2 is the first position matching s_2. Then, the algorithm scans t_3 to t_9 with $\Delta = [0, 6]$ to determine all positions matching s_8. There is no position matching s_8.

Third, the algorithm determines that t_3 matches s_2. Then, the algorithm scans t_4 to t_{10} to determine all positions matching s_8. Position t_{10} matches s_8; however, there is more than one state in POS occurring between t_4 and t_{10},

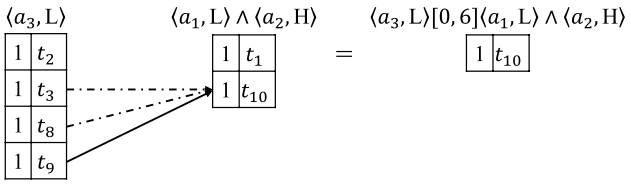


Fig. 5 An example of TPList production

such as s_5, s_2 and s_7. Once the algorithm determines that a state in POS happens in the current gap $[t_4, t_{10}]$, this iteration will terminate.

Fourth, t_8 is the third position matching s_2. When t_9 does not match s_8 and s_2 is found to occur in t_9, the iteration is terminated.

Fifth, t_9 is the fourth position matching s_2. t_{10} is found to match s_8, and there is no state in POS occurring in the gap $[t_{10}, t_{16}]$. Because the length of MTS is 10, namely, $|T| = 10$, the wildcard gap $[t_{10}, t_{16}]$ is equivalent to $[t_{10}, t_{10}]$.

Therefore, $\mathcal{I}_P = \{t_9 t_{10}\}, |\mathcal{I}_P| = 1$, and $Sup(P) = \frac{1}{10\times7} = \frac{1}{70} < \gamma = 0.02$. P is not a frequent Tri-SASP. The supports of the other candidate Tri-SASPs can be obtained in the same way.

Vertical Computation

Figure 5 shows an example of TP-List production for states s_2 and s_8. The only difference between the horizontal and vertical techniques is the method of support computing. First, the algorithm constructs a TP-List for each frequent Tri-SASP of length 1, namely, all elements of 1-$\mathcal{P}$. For example, the TP-List of s_2 is $\{(1, t_2), (1, t_3), (1, t_8), (1, t_9)\}$. The TP-List of s_8 is $\{(1, t_1), (1, t_{10})\}$.

Second, the TP-List of $P = s_2[0, 6]s_8$ is the cartesian product of those of s_2 and s_8. Treating t_1 as a termination, the wildcard gap is $[t_{-6}, t_0]$; then, no start position of s_2 occurs in it. Treating t_{10} as a termination, the wildcard gap is $[t_3, t_9]$; then, start positions t_3, t_8 and t_9 occur in it.

Third, we remove these invalid position sequences. We determine whether there is at least one state in POS occurring between the starting and terminating positions. For example, state s_5 happens at t_4, which occurs between t_3 and t_{10}. Hence, $t_3 t_{10}$ is not a matching position sequence.

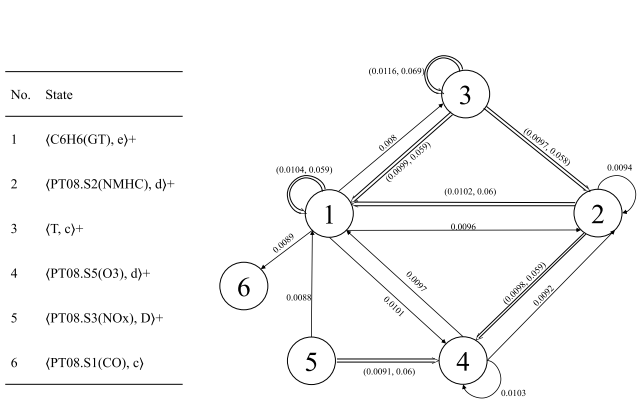
Fourth, the TP-List of P is $\{(1, t_{10})\}$, which means there is a matching position sequence ending with t_{10}. Hence, the occurrence of P is 1, and the support is $\frac{1}{70} < \gamma = 0.02$. P is not a frequent Tri-SASP. The TP-Lists of the other candidate Tri-SASPs can be obtained in the same way.

Tri-SASP Growth

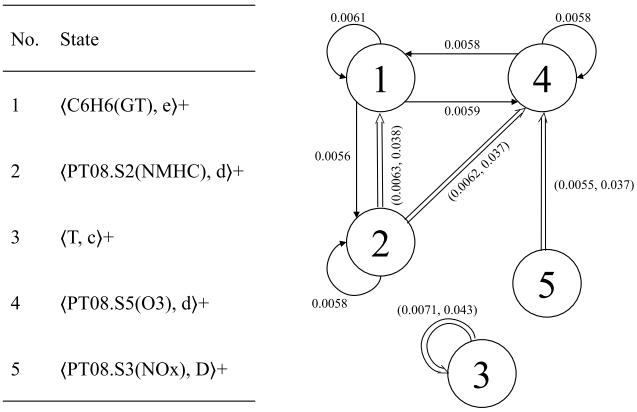
Because γ is set to 0.02, all frequent Tri-SASPs of length 2 are obtained by either the horizontal or vertical algorithm. The results of 2-$\mathcal{P}$ are listed as follows:

(a_3, L)[0,6](a_2, L)(0.0286), (a_3, L)[0,6](a_3, L)(0.0286), (a_1, H)[0,6](a_1, L)(0.0286), (a_1, L)[0,6](a_3, L)(0.0429) and (a_2, H)[0,6](a_3, L)(0.0286).

To determine all frequent Tri-SASPs of length 3, the set of candidate Tri-SASPs, namely, 3-$\mathcal{C}$ = 2-$\mathcal{P}\otimes$1-$\mathcal{P}$ = {(a_3, L)[0,6](a_2, L)[0, 6](a_2, L), ..., (a_2, H)[0,6](a_3, L)[0, 6](a_2, H)}, is determined. The cardinality of |3-$\mathcal{C}$| is $5 \times 7 = 35$. Through horizontal or vertical support computation, we have 3-$\mathcal{P} = \emptyset$. Therefore, the final output is 1-$\mathcal{P} \cup$ 2-$\mathcal{P}$.



(a) SP (α = 0 .1, β = 0 .15, γ = 0 .0088, λ = 0 .058)



(b) CP (α = 0 .1, γ = 0 .0055, λ = 0 .037)

Fig. 6 Visualization of Tri-SASPs and rules on Dataset I

Table 6 Basic information of the datasets

Dataset	Name	|T|	|A|	Area
I	AirQuality	9,358	15	Environments
II	CentralAirConditioning	88,840	51	Equipment
III	ESP-Sanding-Diagnose	3,893	22	Petroleum
IV	ESP-Working-Diagnose	33,001	14	Petroleum

Experiments

In this section, we focus on the problem of temporal association analysis, where the scalability of the algorithm and the readability of the patterns is the most important. Therefore, the following questions are answered by experiments:

(1) What interesting Tri-SASPs can we obtain from the real-world datasets?
(2) What is the difference between the discovered Tri-SASPs w.r.t. the SP and CP strategies?
(3) How good is the performance of the horizontal and vertical algorithms?

Datasets

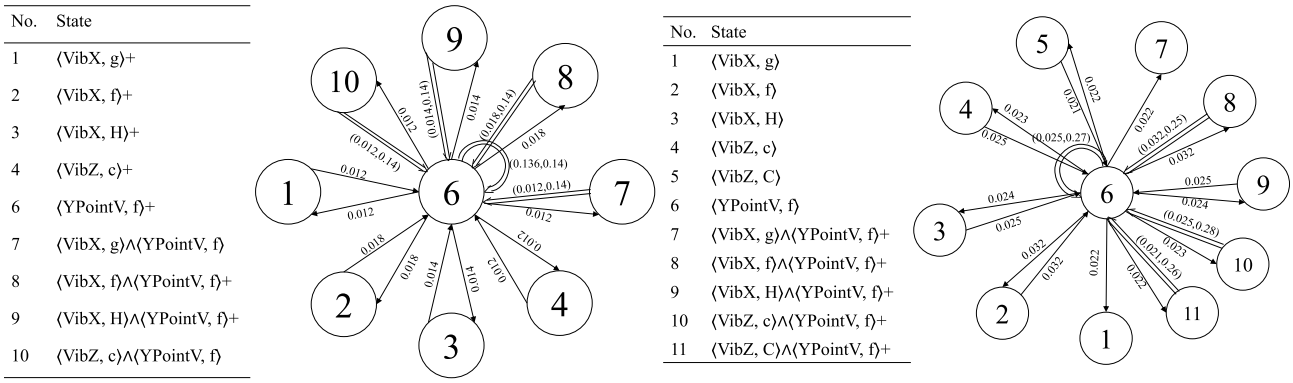
Table 6 shows the information of the temporal MTS datasets. The original AirQuality dataset was downloaded from UCI[1]. CentralAirConditioning is a semi-open-access dataset for the Chinese National Contest of Maths Models[2]. The last two oilwell maintenance datasets are provided by the China National Offshore Oil Corporation (CNOOC)[3].

Interpretability

Question (1) is answered here.

Figures 6 and 7 enhance human cognition about state associations by illustrating a number of nodes and directed edges. For brevity of visualization, we first adjust the thresholds β and γ to keep the number of frequent Tri-SASPs with length 2 in the range of 5 to 15. Although interesting Tri-SASPs with length no less than 3 cannot be observed directly, the graph still provides some clues. Second, the tri-wildcard gap Δ between any two states is set to [0, 6]. Third, there are two kinds of states. The ones in the POS region are marked with +, while another one belonging to BND has no tag. Fourth, two kinds of edges are introduced: (1) $s_1 \xrightarrow{Sup(P)} s_2$ denotes that $s_1 \Delta s_2$ is frequent but $r : s_1 \Rightarrow s_2$ is not confident; and (2) $s_1 \overset{(Sup(P),c(r))}{\Longrightarrow} s_2$ denotes that $s_1 \Delta s_2$ is frequent and $r : s_1 \Rightarrow s_2$ is confident. Finally, we present only two state transition charts from datasets I and IV, namely, AirQuality and ESP-Working-Diagnose.

Figure 6 shows the visualization of the Tri-SASPs and their temporal association rules on Dataset I. There are 6 states/nodes and 5 nodes in Fig. 6a, b, respectively. In Fig. 6a, the associations among nodes ①, ②, ③ and ④ are stronger than that between nodes ⑤ and ⑥. Rules such as ③ $\overset{(0.0102,0.06)}{\Longrightarrow}$ ② have two interpretations: (1) the probability of pattern ③[0, 6]② occurring is 0.0102; and (2) if state ② occurs, the probability of state ② occurring is 0.06. Patterns such as ① $\xrightarrow{0.008}$ ③ can be interpreted as indicating that the probability of pattern ①[0, 6]③ occurring is 0.008. In Fig. 6b, node ③ seems independent of the other nodes because there is no pattern or rule among them. However, the hidden temporal associations around node ③ can be observed through decreasing γ or λ.



(a) SP (α = 0 .05, β = 0 .091, γ = 0 .012, λ = 0 .14)

(b) CP (α = 0 .05, γ = 0 .02, λ = 0 .25)

Fig. 7 Visualization of Tri-SASPs and rules on Dataset IV

[1] http://archive.ics.uci.edu/ml/datasets/Air+Quality

[2] http://www.tipdm.org/bdrace/index.html

[3] http://www.cnooc.com.cn/en/

Table 7 The number of different Tri-SASPs between the SP and CP strategies on Dataset I

(a) $\beta = 0.08, \gamma = 0.0018$									
α	0.015	0.02	0.025	0.03	0.035	0.04	0.045	0.05	0.055
$\lvert\mathcal{P}^{SP} - \mathcal{P}^{CP}\rvert$	194	184	186	182	171	159	143	114	94
$\lvert\mathcal{P}^{CP} - \mathcal{P}^{SP}\rvert$	39	8	4	1	3	3	4	14	9
(b) $\alpha = 0.05, \beta = 0.08$									
γ	0.0017	0.0018	0.0019	0.002	0.0021	0.0022	0.0023	0.0024	0.0025
$\lvert\mathcal{P}^{SP} - \mathcal{P}^{CP}\rvert$	123	114	81	92	82	76	69	66	66
$\lvert\mathcal{P}^{CP} - \mathcal{P}^{SP}\rvert$	7	14	4	4	1	2	5	4	4

Figure 7 shows the visualization of Tri-SASPs and their temporal association rules on Dataset IV. There are 9 states/nodes in Fig. 7a and 11 nodes in Fig. 7b. State ⑥ is the center of both Fig. 7a, b. However, state ⑥ belongs to POS in the SP strategy, while it belongs to BND in the CP strategy. In terms of the SP strategy, POS = {①, ②, ③, ④, ⑧, ⑨} and BND = {⑦, ⑩}. In terms of the CP strategy, POS = {⑦, ⑧, ⑨, ⑩, ⑪} and BND = {①, ②, ③, ④, ⑤, ⑥}. The transition probabilities between the same two pairs of states are different. In Fig. 7b, $Sup(⑥[0, 6]④) = 0.023$, while $Sup(④[0, 6]⑥) = 0.025$. Interestingly, the supports of the symmetrical Tri-SASPs in Fig. 7a are the same. For example, $Sup(⑥[0, 6]③) = Sup(③[0, 6]⑥) = 0.014$, $Sup(⑥[0, 6]①) = Sup(①[0, 6]⑥) = 0.012$, and $Sup(⑥[0, 6]⑧) = Sup(⑧[0, 6]⑥) = 0.018$. This is because state nodes ①, ②, ③, ④, ⑦, ⑧, ⑨, ⑩ and ⑥ appear alternately in Dataset IV. It is not useful for experts to further distinguish the causalities between events. Fortunately, the temporal rules obtained here are very informative.

Diversity

Question (2) is answered here.

Tables 7 and 8 show the diversity between the SP and CP strategies on Datasets I and IV, respectively. Diversity is defined as

$$|\mathcal{P}^{SP} - \mathcal{P}^{CP}| + |\mathcal{P}^{CP} - \mathcal{P}^{SP}|.$$

In Table 7, it can be observed that (1) the SP strategy finds more Tri-SASPs than the CP one; and (2) the diversity increases with lower α and γ. In Table 7 (a), when $\alpha = 0.015$, the diversity between the SP and CP is the maximal value, $194 + 39 = 233$. When $\alpha = 0.055$, the diversity between the SP and CP is the minimal value, $94 + 9 = 103$. In Table 7 (b), when $\gamma = 0.0017$, the diversity between the SP and CP is the maximal value, 130. When $\gamma = 0.0025$, the diversity between the SP and CP is the minimal value, 100.

Table 8 shows the number of different Tri-SASPs between the SP and CP strategies on Dataset IV. With Table 8, it can be observed that (1) the CP strategy finds more Tri-SASPs than the SP strategy; and (2) the diversity between the SP and CP decreases with larger α and γ. In Table 8 (a), when $\alpha = 0.007$, the diversity between the SP and CP is the maximal value, 1,458. When $\alpha = 0.009$, the diversity between the SP and CP is the minimal value, 114. In Table 8 (b), when $\gamma = 0.006$, the diversity between the SP and CP is the maximal value, 664. When $\gamma = 0.0034$, the diversity between the SP and CP is the minimal value, 51.

In short, the SP and CP strategies cannot replace each other in general. Given an arbitrary MTS, both of them are worth trying. Various tri-partition state alphabets specified by users can bring many different Tri-SASPs.

Table 8 The number of different Tri-SASPs between the SP and CP strategies on Dataset IV

(a) $\beta = 0.006, \gamma = 0.0055$									
α	0.002	0.003	0.004	0.005	0.006	0.007	0.008	0.009	0.01
$\lvert\mathcal{P}^{SP} - \mathcal{P}^{CP}\rvert$	21	21	21	21	21	20	19	16	16
$\lvert\mathcal{P}^{CP} - \mathcal{P}^{SP}\rvert$	1,137	677	731	761	1,043	1,438	374	98	99
(b) $\alpha = 0.005, \beta = 0.01$									
γ	0.006	0.01	0.014	0.018	0.022	0.026	0.03	0.034	0.038
$\lvert\mathcal{P}^{SP} - \mathcal{P}^{CP}\rvert$	21	6	5	5	5	5	4	2	3
$\lvert\mathcal{P}^{CP} - \mathcal{P}^{SP}\rvert$	643	340	217	135	112	96	68	49	65

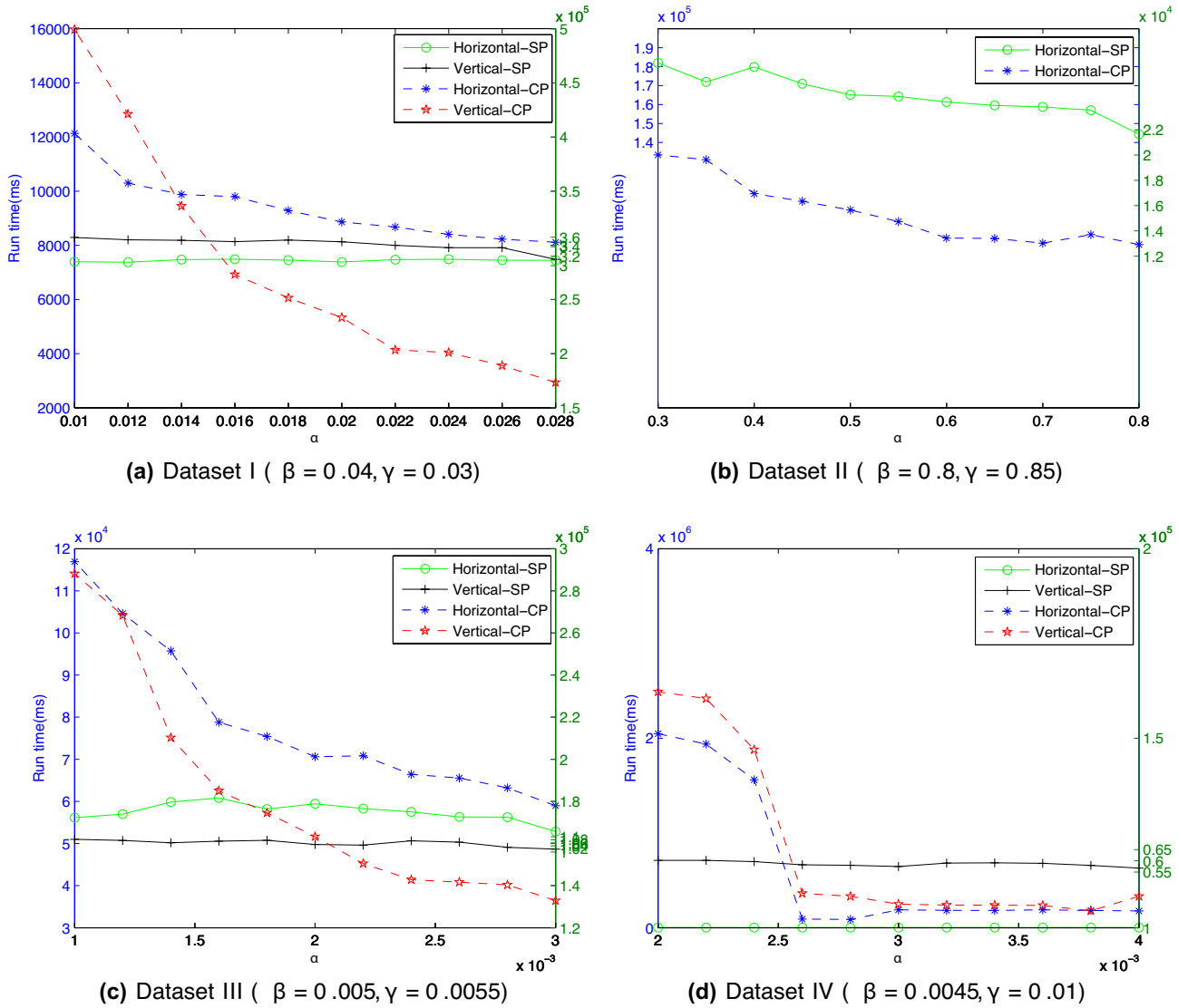


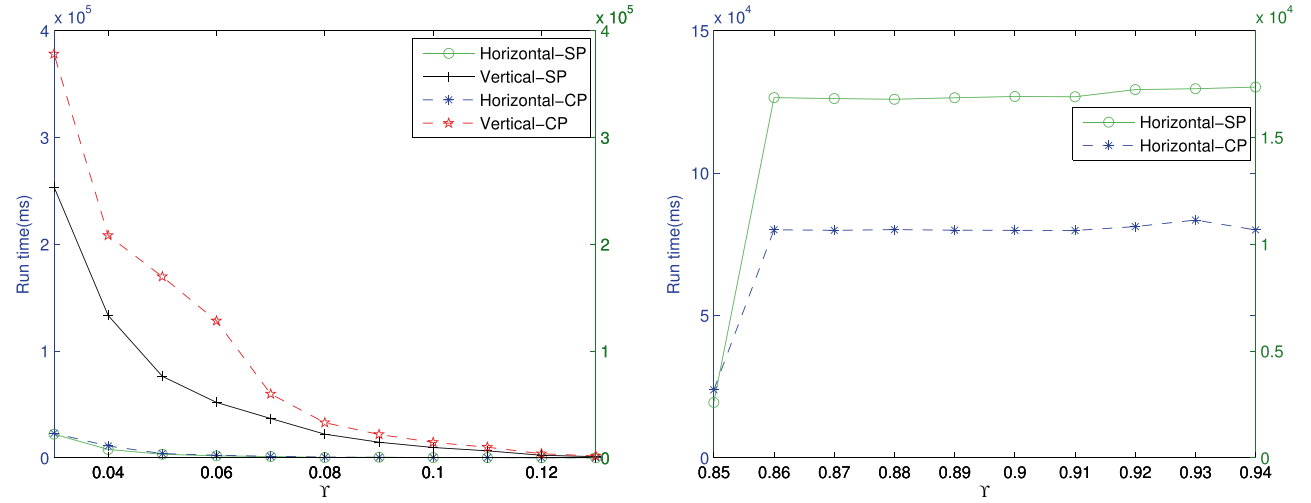
Fig. 8 The runtime of the techniques w.r.t. α. The SP (solid line) corresponds to the Y axis on the left, and the CP (dotted line) corresponds to the Y axis on the right

Scalability

Question (3) is answered here.

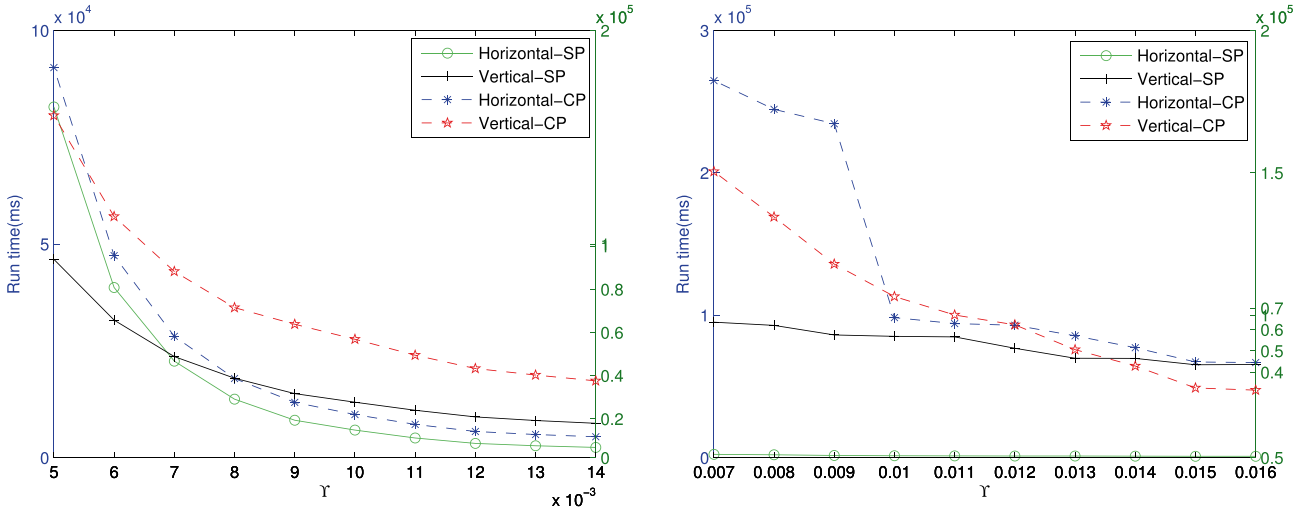
Two sets of experiments are undertaken to investigate the runtime with respect to thresholds α and γ. Figure 8 shows the results of the first set of experiments. The left Y axis corresponds to the two SP-strategy-based techniques, and the right one corresponds to the two CP-strategy-based techniques. Figure 8b shows only two folding lines for the horizontal techniques. This is because the runtimes of the two vertical ones are greater than 24 hours. Moreover, the performance of the vertical ones becomes very poor when the TPLists of the Tri-SASPs are very large. Otherwise, the vertical techniques will be very efficient; see Fig. 8a, c.

Figure 9 shows the results of the second set of experiments. Namely, the runtimes of the threshold and compressed partition strategies based on the horizontal and vertical techniques with respect to γ are shown. The left Y axis corresponds to the two SP-strategy-based techniques, and the right one corresponds to the two CP strategy-based techniques. Similarly, the runtime of each technique becomes very long with smaller γ. In Fig. 9b, the two vertical techniques are also too slow to obtain patterns. In Fig. 9c, d, the two CP-strategy-based techniques are much faster than the two SP-strategy-based ones. This is because the CP strategy generates much smaller POS



(a) Dataset I ($\alpha = 0.03, \beta = 0.072$)

Dataset II ($\alpha = 0.85, \beta = 0.95$)



(c) Dataset III ($\alpha = 0.003, \beta = 0.007$)

(d) Dataset IV ($\alpha = 0.0025, \beta = 0.0045$)

Fig. 9 The runtime of the techniques w.r.t. γ. The SP (solid line) corresponds to the Y axis on the left, and the CP (dotted line) corresponds to the Y axis on the right

and BND regions than the SP one. The value of β does not effect the runtime of the SP-strategy-based techniques because POS ∪ BND remains unchanged when α is specified.

Conclusion

Given an MTS, we define frequent Tri-SASPs and confident temporal association rules to help human experts make better decisions. The SP strategy gives more frequent states greater importance, while the CP strategy gives more special (longer) states greater importance. For an MTS under an approximately uniform distribution, the horizontal algorithm is faster than the vertical one; otherwise, the vertical algorithm is faster Moreover, numerous insignificant states containing "Normal (N)" are ignored to focus on more interesting Tri-SASPs.

The following research topics deserve further investigation:

- Obtaining frequent Tri-SASPs in a depth-first, parallel, incremental, distributed or hybrid way.
- Using a one-off or non-overlapping condition instead of a general one.
- Considering fuzzy, uncertain, weighted and utility variants of Tri-SASPs.
- Extending Tri-SASPs to sequence databases, which are more complex and more widespread in reality.

Funding This study was funded by the National Natural Science Foundation of China (grant numbers 41604114, 62006200); the Sichuan Science and Technology Program (grant numbers 2019YFG0216, 2020YFG0307); the Scientific Research and Innovation Team of Sichuan Tourism University (grant number 18SCTUTD06); and the Scientific Research Project of Sichuan Tourism University (grant number 2020SCTU14).

Declarations

Ethical Approval This article does not contain any studies with animals performed by any of the authors.

Conflict of Interest The authors declare that there are no conflicts of interest regarding the publication of this paper.

References

1. Yang B, Li JH. Complex network analysis of three-way decision researches. Int J Mach Learn Cybern. 2020:973–87.
2. Yao YY. Three-way decision: An interpretation of rules in rough set theory. In: International Conference on Rough Sets and Knowledge Technology. 2009:642–49.
3. Pawlak Z. Rough sets. Int J Comput Inform Sci. 1982;11(5):341–56.
4. Campagner A, Cabitza F, Ciucci D. Three-way decision for handling uncertainty in machine learning: a narrative review. In: International Joint Conference on Rough Sets. Springer. 2020:137–52.
5. Yao YY. Three-way decisions and cognitive computing. Cogn Comput. 2016;8(4):543–54.
6. Yao YY. The geometry of three-way decision. Appl Intell. 2021. https://doi.org/10.1007/s10489-020-02142-z.
7. Li JH, Huang CC, Qi JJ, Qian YH, Liu WQ. Three-way cognitive concept learning via multi-granularity. Inf Sci. 2017;378:244–63.
8. Mao H, Zhao SF, Yang LZ. Relationships between three-way concepts and classical concepts. J Intell Fuzzy Syst. 2018;35(1):1063–75.
9. Deng XF, Yao YY. Decision-theoretic three-way approximations of fuzzy sets. Inf Sci. 2014;279:702–15.
10. Yao YY. Interval sets and three-way concept analysis in incomplete contexts. Int J Mach Learn Cybern. 2017;8(1):3–20.
11. Fang Y, Min F. Cost-sensitive approximate attribute reduction with three-way decisions. Int J Approx Reason. 2019;104:148–65.
12. Min F, Liu FL, Wen LY, Zhang ZH. Tri-partition cost-sensitive active learning through knn. Soft Comput. 2019;23(5):1557–72.
13. Ye X, Liu D. An interpretable sequential three-way recommendation based on collaborative topic regression. Expert Syst Appl. 2021;168:114454. https://doi.org/10.1016/j.eswa.2020.114454.
14. Zhang HR, Min F, Shi B. Regression-based three-way recommendation. Inf Sci. 2017;378:444–61.
15. Min F, Zhang SM, Ciucci D, Wang M. Three-way active learning through clustering selection. Int J Mach Learn Cybern. 2020;11(5):1033–46.
16. Yue XD, Chen YF, Miao DQ, Qian J. Tri-partition neighborhood covering reduction for robust classification. Int J Approx Reason. 2016;83:371–84.
17. Yu H, Wang XC, Wang GY, Zeng XH. An active three-way clustering method via low-rank matrices for multi-view data. Inf Sci. 2020;507:823–39.
18. Min F, Zhang ZH, Zhai WJ, Shen RP. Frequent pattern discovery with tri-partition alphabets. Inf Sci. 2020;507(1):715–32.
19. Li HX, Zhang LB, Huang B, Zhou XZ. Sequential three-way decision and granulation for cost-sensitive face recognition. Knowl-Based Syst. 2016;91(C):241–51.
20. Ren RS, Wei L. The attribute reductions of three-way concept lattices. Knowl-Based Syst. 2016;99:92–102.
21. Zhou B, Yao YY, Luo JG. Cost-sensitive three-way email spam filtering. J Intell Inf Syst. 2014;42(1):19–45.
22. Saira Q, Hasan M, Hammad M, Omer BM. Relationship identification between conversational agents using emotion analysis. Cogn Comput. 2021:1–15.
23. Wang GY, Yu H. Multi-granularity cognitive computing-a new model for big data intelligent computing. Frontiers of Data and Domputing. 2020;1(2):75–85.
24. Agrawal R, Srikant R. Mining sequential patterns. Proceedings of the International Conference on Data Engineering. 1995;95:3–14.
25. Gan WS, Lin JCW, Fournier-Viger P, Chao HC, Yu PS. A survey of parallel sequential pattern mining. ACM Trans Knowl Discov Data. 2019;13(3):1–34.
26. Sakai H, Nakata M. Rough set-based rule generation and apriori-based rule generation from table data sets: a survey and a combination. CAAI Transactions on Intelligence Technology. 2019;4(4):203–13.
27. Fournier-Viger P, Lin JCW, Kiran RU, Koh YS, Thomas R. A survey of sequential pattern mining. Data Science and Pattern Recognition. 2017;1(1):54–77.
28. Srikant R, Agrawal R. Mining sequential patterns: Generalizations and performance improvements. In: International Conference on Extending Database Technology. Springer. 1996:1–17.
29. Agrawal R, Srikant R. Fast algorithms for mining association rules. In: Proceedings of the 20th Very Large Data Bases Conference. 1994:487–99.
30. Zaki MJ. Spade: An efficient algorithm for mining frequent sequences. Mach Learn. 2001;42(1):31–60.
31. Ayres J, Flannick J, Gehrke J, Yiu T. Sequential pattern mining using a bitmap representation. In: Proceedings of the eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. 2002:429–35.
32. Yang ZL, Kitsuregawa M. Lapin-spam: An improved algorithm for mining sequential pattern. In: 21st International Conference on Data Engineering Workshops. 2005:1222–25.
33. Pei J, Han JW, Mortazavi-Asl B, Wang JY, Pinto H, Chen QM, Dayal U, Hsu MC. Mining sequential patterns by pattern-growth: the prefixspan approach. IEEE Trans Knowl Data Eng. 2004;16(11):1424–40.
34. Zaki MJ. Scalable algorithms for association mining. IEEE Trans Knowl Data Eng. 2000;12(3):372–90.
35. Wang JY, Han JW, Li C. Frequent closed sequence mining without candidate maintenance. IEEE Trans Knowl Data Eng. 2007;19(8):1042–56.
36. Luo CN, Chung SM. Efficient mining of maximal sequential patterns using multiple samples. In: SIAM International Conference on Data Mining. 2005:415–26.
37. Lo D, Khoo SC, Li JY. Mining and ranking generators of sequential patterns. In: SIAM International Conference on Data Mining. 2008:553–64.
38. Chang JH. Mining weighted sequential patterns in a sequence database with a time-interval weight. Knowl-Based Syst. 2011;24(1):1–9.
39. Lan GC, Hong TP, Tseng VS, Wang SL. Applying the maximum utility measure in high utility sequential pattern mining. Expert Syst Appl. 2014;41(11):5071–81.
40. Muzammal M, Raman R. Mining sequential patterns from probabilistic databases. Knowl Inf Syst. 2015;44(2):325–58.
41. Fiot C, Laurent A, Teisseire M. From crispness to fuzziness: Three algorithms for soft sequential pattern mining. IEEE Trans Fuzzy Syst. 2007;15(6):1263–77.
42. Zhuang DEH, Li GCL, Wong AK. Discovery of temporal associations in multivariate time series. IEEE Trans Knowl Data Eng. 2014;26(12):2969–82.

43. Tatavarty G, Bhatnagar R, Young B. Discovery of temporal dependencies between frequent patterns in multivariate time series. In: Computational Intelligence and Data Mining. IEEE Symposium on. 2007:688–96.
44. Zhang ZH, Min F. Frequent state transition patterns of multivariate time series. IEEE Access. 2019;7:142934–46.
45. Segura-Delgado A, Gacto MJ, Alcalá R, Alcalá-Fdez J. Temporal association rule mining: An overview considering the time variable as an integral or implied component. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2020;10(4):e1367.
46. Wu YX, Tong Y, Zhu XQ, Wu XD. Nosep: Nonoverlapping sequence pattern mining with gap constraints. IEEE Transactions on Cybernetics. 2017;48(10):2809–22.
47. Wu XD, Zhu XQ, He Y, Arslan AN. Pmbc: Pattern mining from biological sequences with wildcard constraints. Comput Biol Med. 2013;43(5):481–92.
48. Min F, Wu YX, Wu XD. The apriori property of sequence pattern mining with wildcard gaps. International Journal of Functional Informatics and Personalised Medicine. 2012;4(1):15–31.
49. Yao YY, Wong SK. A decision theoretic framework for approximating concepts. Int J Man Mach Stud. 1992;37(6):793–809.
50. Ziarko W. Variable precision rough set model. J Comput Syst Sci. 1993;46(1):39–59.
51. Sang BB, Guo YT, Shi DR, Xu WH. Decision-theoretic rough set model of multi-source decision systems. Int J Mach Learn Cybern. 2017:1–14.
52. Hu BQ. Three-way decisions space and three-way decisions. Inf Sci. 2014;281(281):21–52.
53. Li XN, Yi HJ, She YH, Sun BZ. Generalized three-way decision models based on subset evaluation. Int J Approx Reason. 2017;83(C):142–59.
54. Liu D, Liang DC, Wang CC. A novel three-way decision model based on incomplete information system. Knowl-Based Syst. 2016;91:32–45.
55. Xu WH, Li MM, Wang XZ. Information fusion based on information entropy in fuzzy multi-source incomplete information system. Int J Fuzzy Syst. 2017;19(4):1200–16.
56. Wang M, Min F, Zhang ZH, Wu YX. Active learning through density clustering. Expert Syst Appl. 2017;85:305–17.
57. Sun L, Zhang XY, Qian YH, Xu JC, Zhang SG, Tian Y. Joint neighborhood entropy-based gene selection method with fisher score for tumor classification. Appl Intell. 2019;49(4):1245–59.
58. Wang CZ, Huang Y, Shao MW, Hu QH, Chen DG. Feature selection based on neighborhood self-information. IEEE Transactions on Cybernetics. 2019;50(9):4031–42.
59. Tan CD, Min F, Wang M, Zhang HR, Zhang ZH. Discovering patterns with weak-wildcard gaps. IEEE Access. 2016;4:4922–32.