International Journal of Innovative Computing, Information and Control
Volume **8**, Number **1(A)**, January **2012**
 ISSN 1349-4198
pp. 453-470

MINING CONDENSED SETS OF FREQUENT EPISODES WITH MORE ACCURATE FREQUENCIES FROM COMPLEX SEQUENCES

Min Gan* and Honghua Dai

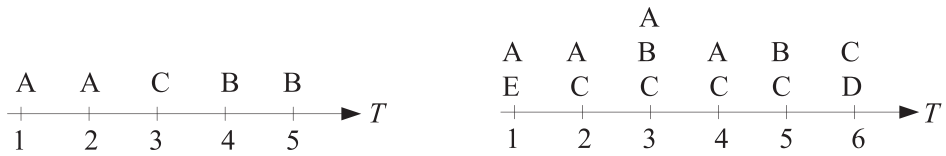
School of Information Technology
Deakin University
221 Burwood Highway, Melbourne, Victoria 3125, Australia
*Corresponding author: min.gan.au@gmail.com

Received August 2010; revised January 2011

Abstract. *Many previous approaches to frequent episode discovery only accept simple sequences. Although a recent approach has been able to find frequent episodes from complex sequences, the discovered sets are neither condensed nor accurate. This paper investigates the discovery of condensed sets of frequent episodes from complex sequences. We adopt a novel anti-monotonic frequency measure based on non-redundant occurrences, and define a condensed set, nDaCF (the set of non-derivable approximately closed frequent episodes) within a given maximal error bound of support. We then introduce a series of effective pruning strategies, and develop a method, nDaCF-Miner, for discovering nDaCF sets. Experimental results show that, when the error bound is somewhat high, the discovered nDaCF sets are two orders of magnitude smaller than complete sets, and nDaCF-miner is more efficient than previous mining approaches. In addition, the nDaCF sets are more accurate than the sets found by previous approaches.*
Keywords: Frequent episodes, Condensed sets, Sequence data mining

1. **Introduction.** Sequences are an important type of data. Large numbers of real-world data can be represented as sequences, such as DNA sequences [1], Web-click streams [2] and audio/video streams [3,4]. Sequences in data mining can be classified into simple sequences, i.e., single-item sequences as shown in Figure 1(a), and complex sequences in which multiple items may appear at one timestamp (see Figure 1(b)).

Episodes introduced by Mannila et al. [5] are important pattern for modelling the relative order of occurrences for different types of data elements over a single data sequence. For instance, the order 'A occurs before C' can be represented as a serial episode denoted as $\langle AC \rangle$. One fundamental and important problem in single sequence analysis is finding frequent episodes (FEs), i.e., the episodes with *supports* (frequencies) no less than a user specified threshold min $\_sup$, as FEs are able to capture 'common features' of the relative order of occurrence within a sequence. Since FEs were introduced [5], several typical approaches [5-11] have been proposed to discover FEs. Most approaches [5-9,11] only focus on FE mining from simple sequences. However, in the real world, complex sequences need to be considered when multiple events (items) may appear at each time slot. For example, in the instance of stock price analysis in [10], ten evens need to be considered in each time slot. Recently, Huang et al. [10] began to investigate FE mining in complex sequences, and proposed $EMMA$ as an approach to find all FEs in complex sequences. $EMMA$, however, suffers from three significant deficiencies. First, the frequency measure adopted in $EMMA$ violates anti-monotonicity, a common principle in frequent pattern mining [12]. The second deficiency, inaccurate resulting sets, is a consequence of non-anti-monotonic measures. The third deficiency with EMMA is that resulting sets are unable to be condensed because all FEs will be found. In the community of frequent pattern (FP) mining [13-16], it has been widely recognised that it is not necessary to find complete FP sets and only condensed FP sets need to be extracted. With large numbers of redundant patterns, complete FP sets not only bring inconvenience to users' comprehension and utilisation, but also result in high search complexity. Consequently, inaccuracy and non-condensation of the complete FE sets found by $EMMA$ hinder its real applications.



(a) A single-item sequence (b) A complex sequence

Figure 1. Two classes of sequences

In order to overcome the deficiencies of $EMMA$ and develop a more efficient and effective approach for FE discovery from complex sequences, we adopt a new frequency measure, $LMaxnR\text{-}O\text{-}freq$, which we proposed in our latest publication [17]. Based on this measure, we define a condensed set of FEs, nDaCF (the set of non-derivable approximately closed frequently episodes). We then introduce a series of pruning strategies and develop a method, nDaCF-miner for discovering nDaCF sets.

Table 1 shows a comparison between our work and previous studies in terms of input, frequency measure and output. For output, three aspects are compared: the outputted FE set, its condensation and accuracy. In Column 5, 'F' means false patterns may be included in the resulting set, and 'M' means frequent patterns may be missed. [1] The comparison in Table 1 demonstrates that, compared with previous studies, our work has three unique features and main advantages. These are a more appropriate anti-monotonic frequency measure, $LMaxnR\text{-}O\text{-}freq$ [17], more condensed resulting sets and more accurate results. These will be verified in the experimental evaluation.

Table 1. A comparison between our work and previous studies

Input	**Frequency measure**	**Output**			**Paper**
		FE set	Condensation	Accuracy	
A single-item sequence	fixed-win-freq	complete	no	(F,)	[5]
	mo-freq	complete	no	(, M)	[6,7]
	T-freq	maximal	high	(F, M)	[8]
	non-overlapped-freq	complete	no	(, M)	[9]
	mo-freq	closed	insufficient	(, M)	[11]
A complex sequence	distinct-bound-st-freq	complete	no	(F, M)	[10]
	LMaxnR-O-freq	nDaCF	high	(,)	this

The rest of this paper is organised as follows. In Section 2, we present preliminaries, frequency measure and problem definition. Section 3 proposes the mining method, and experimental results are presented in Section 4. Section 5 concludes the paper.

2. Preliminaries, Frequency Measure and Problem Definition.

[1]Please refer to [17,19] for the details of accuracy analysis.

2.1. **Preliminaries.** In this paper, we take complex sequences as input. Given a finite set of items $I = \{i_1, i_2, \ldots, i_{|I|}\}$ ($|I| \geq 1$), a complex sequence, S, over I is an ordered list of data elements, denoted as $S = \langle (e_1, t_1)(e_2, t_2) \ldots (e_n, t_n) \rangle$ ($n \geq 1$), where $e_j \subseteq I$ is a data element, $t_j \in \{1, 2, \ldots\}$ ($j = 1, \ldots, n$) is the occurrence time (timestamp) of e_j in S, and $t_j < t_{j+1}$ for all $j = 1, \ldots, n-1$. We assume that input sequences are consecutive, i.e., $t_j = j$. Sequence $\langle (e_1, 1)(e_2, 2) \ldots (e_n, n) \rangle$ is abbreviated as $\langle (e_1)_1 (e_2)_2 \ldots (e_n)_n \rangle$. For example, the sequences in Figures 1(a) and 1(b) can be represented as $\langle (A)_1 (A)_2 (C)_3 (B)_4 (B)_5 \rangle$ and $\langle (AE)_1 (AC)_2 (ABC)_3 (AC)_4 (BC)_5 (CD)_6 \rangle$ respectively.

Definition 2.1. *(Serial Episode) Given itemset $I = \{i_1, i_2, \ldots, i_{|I|}\}$ ($|I| \geq 1$), a serial episode*[2] *P over I is an ordered list of types of data elements, denoted as $P = \langle p_1 p_2 \ldots p_m \rangle$, where $p_j \subseteq I$ ($p_j \neq \emptyset$) ($j = 1, \ldots, m$) is called an episode element. The length of P, denoted as $P.L$, is defined as m, and the size of P, denoted as $P.size$, is defined as the number of items that P contains. Essentially episode P imposes a constraint on the relative order of occurrences of p_j, i.e., p_j always occurs before p_{j+1} for all $j = 1, 2, \ldots, m-1$.*

In this paper, all items in each episode element p_j are listed in alphabetical order in a pair of parentheses, and the parentheses are omitted when p_j contains only one item. For example, $P = \langle B(AC) \rangle$ is an episode which requires that B occurs before A and C, and A and C appear simultaneously.

Definition 2.2. *(Sub-episode) An episode $P = \langle p_1 p_2 \ldots p_m \rangle$ is a sub-episode of another episode $P' = \langle p'_1 p'_2 \ldots p'_n \rangle$, denoted by $P \sqsubseteq P'$ or $P' \sqsupseteq P$, ($P \sqsubset P'$ if $P \neq P'$) if there exists $1 \leq j_1 < j_2 < \ldots < j_m \leq n$, such that $p_k \subseteq p'_{j_k}$ for all $k = 1, \ldots, m$.*

Definition 2.3. *(Sliding Window) Given sequence $S = \langle (e_1)_1 (e_2)_2 \ldots (e_n)_n \rangle$, a sliding window in S from starting time st to ending time et, denoted as $win(S, st, et)$, is defined as $\langle (e_{st})_{st} (e_{st+1})_{st+1} \ldots (e_{et})_{et} \rangle$, where $1 \leq st < et \leq n$. The width of the window is defined as $et - st$* [8].

Definition 2.4. *(Occurrence, Minimal Occurrence) Given $S = \langle (e_1)_1 (e_2)_2 \ldots (e_n)_n \rangle$ and episode $P = \langle p_1 p_2 \ldots p_m \rangle$, if there exists $1 \leq j_1 < j_2 < \ldots < j_m \leq n$ s.t. $p_k \subseteq e_{j_k}$ for all $k = 1, 2, \ldots, m$, then we say that P occurs in S, and $o = \langle (e_{j_1})_{j_1} (e_{j_2})_{j_2} \ldots (e_{j_m})_{j_m} \rangle$ is an occurrence of P in S. For simplicity, we use a timestamp list $\langle j_1, j_2, \ldots, j_m \rangle$ to represent o, and use $o[k]$ to denote j_k ($k = 1, 2, \ldots, m$). Furthermore, if $st \leq j_1 < j_m \leq et$, then we say that window $win(S, st, et)$ contains P. The width of occurrence $o = \langle j_1, j_2, \ldots, j_m \rangle$ is defined as $j_m - j_1$. An occurrence of P in S, $o = \langle j_1, j_2, \ldots, j_m \rangle$, is minimal if P has no other occurrence in S, $o' = \langle j'_1, j'_2, \ldots, j'_m \rangle$, s.t. $(j_1 < j'_1 \wedge j'_m \leq j_m) \vee (j_1 \leq j'_1 \wedge j'_m < j_m)$ (i.e., $[j'_1, j'_m] \subset [j_1, j_m]$).*

The set of all (minimal) occurrences of P in S is denoted as $O(S, P)$ ($MO(S, P)$). The occurrences in any set are ordered by their first timestamps in descending order. For instance, given S in Figure 1(a) and $P = \langle AB \rangle$, $O(S, P) = \{\langle 1, 4 \rangle, \langle 2, 4 \rangle, \langle 1, 5 \rangle, \langle 2, 5 \rangle\}$ and $MO(S, P) = \{\langle 2, 4 \rangle\}$.

An episode P of size $P.size$ can be extended to any super-episode P' of size $P.size + 1$ by adding an item α in two ways: vertical extension and horizontal extension. Vertical extension, also called element extension, involves inserting an item into an episode element. Horizontal extension, also called sequence extension, involves inserting an item (as an episode element) into an episode. According to the position of insertion, episode extensions can be classified into rightmost extensions and non-rightmost extensions. Episode extensions are formally defined as follows.

[2] In this paper, we only consider serial episodes, and do not consider parallel episodes or composite episodes [5].

Definition 2.5. ***(Episode Extension '$\diamond$')*** *Episode $P = \langle p_1 p_2 \ldots p_m \rangle$ can be extended with item $\alpha \in I$ in two ways.*

1. *Vertical Extension '$\diamond_V$'*
   (a) *Rightmost Vertical Extension '$\diamond_V^R$'*
   Episode $P' = \langle p_1 p_2 \ldots p_{m-1} p'_m \rangle$ is called a rightmost vertical extension of P with item α, denoted by $P \diamond_V^R \alpha$, if $p'_m = p_m \cup \{\alpha\}$, and $\forall \beta \in p_m$, $\alpha < \beta$ [3]*.*
   (b) *Non-rightmost Vertical Extension on the jth Element '$\diamond_V^j$'*
   Episode $P' = \langle p_1 p_2 \ldots p_{j-1} p'_j p_{j+1} \ldots p_m \rangle$ ($1 \leq j \leq m$) is called a non-rightmost element extension of P on the jth element with item α, denoted by $P \diamond_V^j \alpha$, if (1) $p'_j = p_j \cup \{\alpha\}$ and (2) $\forall \beta \in p_j$, $\alpha > \beta$ if $j = m$.
2. *Horizontal Extension '$\diamond_H$'*
   (a) *Rightmost Horizontal Extension '$\diamond_H^R$'*
   Episode $P' = \langle p_1 p_2 \ldots p_m \alpha \rangle$ is called a rightmost horizontal extension of P with item α, denoted by $P \diamond_H^R \alpha$.
   (b) *Non-rightmost Horizontal Extension before the jth Element '$\diamond_H^j$'*
   Episode $P' = p_1 p_2 \ldots p_{j-1} \alpha p_j \ldots p_m$ ($1 \leq j \leq m$ and $\alpha \neq p_j$) is called a non-rightmost horizontal extension of P before the jth element with item α, denoted by $P \diamond_H^j \alpha$.

For example, $\langle A(BC) \rangle = \langle AB \rangle \diamond_V^R C$, $\langle A(BC)D \rangle = \langle ABD \rangle \diamond_V^2 C$, $\langle AB \rangle = \langle A \rangle \diamond_H^R B$ and $\langle ABD \rangle = \langle AD \rangle \diamond_H^2 B$. Note that $\langle A(BC) \rangle = \langle AC \rangle \diamond_V^2 B \neq \langle AC \rangle \diamond_V^R B$ since $B > C$, $\langle ACC \rangle = \langle AC \rangle \diamond_H^R C \neq \langle AC \rangle \diamond_H^2 C$ since $C = C$, and $\langle ACCD \rangle = \langle ACD \rangle \diamond_H^3 C \neq \langle ACD \rangle \diamond_H^2 C$ since $C = p_2$.

Definition 2.6. ***(Episode Concatenation '$\oplus$')*** *Given prefix episode $P = \langle p_1 \ldots p_m \rangle$ ($m \geq 1$) and suffix episode $Q = \langle q_1 \ldots q_n \rangle$ ($n \geq 0$), these can be concatenated in two ways.*

1. *Horizontal Concatenation '$\oplus_H$'*
   $P \oplus_H Q = \langle p_1 \ldots p_m q_1 \ldots q_n \rangle$.
2. *Vertical Concatenation '$\oplus_V$'*
   $P \oplus_V Q = \langle p_1 \ldots p_{m-1} p'_m q_2 \ldots q_n \rangle$, *where $p'_m = p_m \cup q_1$.*

For example, $\langle AB \rangle \oplus_H \langle CD \rangle = \langle ABCD \rangle$ and $\langle AB \rangle \oplus_V \langle CD \rangle = \langle A(BC)D \rangle$. Any episode with P as the prefix (P-prefixed episode) can be represented by $P \oplus Q$, where $\oplus \in \{\oplus_H, \oplus_V\}$.

2.2. **Frequency measure.** We adopt a new frequency measure, $LMaxnR\text{-}O\text{-}freq$ [17] to measure frequency of episodes in this paper. This measure was adapted from 'repetitive support' in [18]. This section reviews the definitions briefly.

Definition 2.7. *(Non-redundant Sets of Occurrences) Given sequence S and episode P, a set of occurrences of P in S is non-redundant, if for any two occurrences, $o = \langle j_1, j_2, \ldots, j_m \rangle$ and $o' = \langle j'_1, j'_2, \ldots, j'_m \rangle$ ($o \neq o'$) in this set, $\neg \exists k \in \{1, 2, \ldots, m\}$, s.t. $j_k = j'_k$.*

The set of all non-redundant sets of occurrences of P in S is denoted as $nR\text{-}O(S, P)$. In $nR\text{-}O(S, P)$, the sets with maximal cardinality are included in $MaxnR\text{-}O(S, P)$. For instance, given S in Figure 1(a) and $P = \langle AB \rangle$, $MaxnR\text{-}O(S, P) = \{\{\langle 1, 4 \rangle, \langle 2, 5 \rangle\}, \{\langle 2, 4 \rangle, \langle 1, 5 \rangle\}\}$.

Based on the maximal non-redundant set of occurrences, a naive frequency measure can be defined as:

$$MaxnR\text{-}O\text{-}freq(S, P) = \max_{\forall OS \in nR\text{-}O(S,P)} (|OS|) \tag{1}$$

[3] $\alpha < \beta$ means α is after β alphabetically, e.g., $B < A$.

However, $MaxnR$-O-$freq$ is time expensive to compute. To achieve a more efficient computation, for $MaxnR$-$O(S,P)$, we chose a special set called the leftmost maximal non-redundant set, denoted as $LMaxnR$-$O(S,P)$. To define the set, the occurrences in any non-redundant set are ordered by the ending timestamp in ascending order, i.e., in a sorted set $\{o_1, o_2, \ldots, 0_r\} \in nR$-$O(S,P)$, $o_l[m] < o_{l+1}[m]$ $(m = P.L)$ holds for all $l = 1, 2, \ldots, r-1$.

Definition 2.8. *(LMaxnR-O) The leftmost maximal non-redundant set of occurrences of P in S, $LMaxnR$-$O(S,P)$, is defined as the occurrence set, $OS = \{o_1, \ldots, o_r\}$, that satisfies (1) $OS \in MaxnR$-$O(S,P)$ and (2) $\forall OS' = \{o'_1, \ldots, o'_r\} \in MaxnR$-$O(S,P)$ $(OS \neq OS')$, $o_l[k] \leq o'_l[k]$ holds for all $l = 1, \ldots, r$ and $k = 1, \ldots, P.L$.*

For instance, in $MaxnR$-$O(S,P) = \{\{\langle 1,4\rangle, \langle 2,5\rangle\}, \{\langle 2,4\rangle, \langle 1,5\rangle\}\}$, $LMaxnR$-$O(S,P)$ $= \{\langle 1,4\rangle, \langle 2,5\rangle\}$.

Definition 2.9. *(LMax-nR-O-freq) The measure based on the leftmost maximal non-redundant set of occurrences is defined as $LMaxnR$-O-$freq(S,P) = |LMaxnR$-$O(S,P)|$.*

Given S and min_sup, P is frequent in S if $sup(S,P) = LMaxnR$-O-$freq(S,P) \geq$ min_sup. In the rest of the paper, S is omitted when it is apparent. For example, $sup(S,P) = sup(P)$, $O(S,P) = O(P)$ and $LMaxnR$-O-$freq(S,P) = LMaxnR$-O-$freq(P)$.

2.3. **Problem definition.** This section defines the condensed set, nDaCF and formalises the mining problem.

A pattern P is closed if there exist no supper-pattern P' of P such that $sup(P) = sup(P')$ [12]. Closed patterns [14] are a concise representation that is widely used in frequent pattern mining. However, Pei et al. [15] point out that sets of closed frequent patterns might still be too large since a pattern P can only be pruned if there is a supper-pattern P' with the same support. Even if $sup(P')$ deviates from $sup(P)$ slightly, e.g., $sup(P) = 100$ and $sup(P') = 99$, P cannot be pruned and represented by P'. In this paper, a small deviation between $sup(P')$ and $sup(P)$ is tolerable. Episode P is pruned as a non-approximately-closed episode if there exists supper-episode P' such that the deviation between $sup(P')$ and $sup(P)$ is within a specified maximal error bound θ. The conventional 'closed' is called 'exactly closed' in this paper. Exactly closed episodes and approximately closed episodes are defined as follows.

Definition 2.10. ***((Non) Exactly Closed Episode)*** *Given S, P is exactly closed (eC) if $\neg\exists P' \sqsupset P$, such that $sup(P) = sup(P')$, else P is non-exactly-closed (neC).*

Definition 2.11. ***((Non)Approximately-closed Episode)*** *Given S, P and maximal error bound θ, P is approximately-closed (aC) if $\neg\exists\ P' \sqsupset P$ such that $J(sup(P), sup(P')) = |sup(P') - sup(P)|/sup(P) \leq \theta$, where $J(sup(P), sup(P'))$ represents the deviation between $sup(P')$ and $sup(P)$. Otherwise, we say P is non-approximately-closed (naC).*

Non-exactly-closed episode P can be represented by its supper-episode P' since $sup(P)$ can be exactly derived from $sup(P')$, i.e., given $sup(P')$, it can be deduced that $sup(P) = sup(P')$. Non-approximately-closed episode P can be represented by its supper-episode P' since $sup(P)$ can be approximately derived from $sup(P')$, i.e., given $sup(P')$, it can be deduced that $J(sup(P), sup(P')) \leq \theta$. The derivation relationship is defined as follows.

Definition 2.12. ***(Derivation Relationship)*** *Given S, θ, P and $P' \sqsupset P$, assume $sup(P')$ has been obtained. we say P can be exactly derived from P', denoted as $eD(P,P')$, if $sup(P) = sup(P')$ can be deduced. We say P can be approximately derived from P', denoted as $aD(P,P')$, if it can be deduced that $J(sup(P), sup(P')) \leq \theta$.*

The eD relationship satisfies the transitivity property, i.e., if $eD(P, P')$ and $eD(P', P'')$, then $eD(P, P'')$. However, aD does not satisfy the transitivity property.

Definition 2.13. ***(The nDaCF Set)*** *Given S, $\min\_sup$ and θ, a set of frequent episodes discovered from S, $\mathbb{P}$ is called a nDaCF set, if (1) for $\forall P \in \mathbb{P}$, P cannot be derived from $\mathbb{P}$, i.e., $\neg\exists P' \in \mathbb{P}$ such that $eD(P, P')$ or $aD(P, P')$, and (2) for $\forall P \notin \mathbb{P}$, if $sup(P) \geq \min\_sup$, then P can be derived from $\mathbb{P}$, i.e., $\exists P' \in \mathbb{P}$ such that $eD(P, P')$ or $aD(P, P')$.*

Intuitively, any episode in the nDaCF set is non-derivable, and any frequent episode outside of it can be derived from it. Given a found nDaCF set $\mathbb{P}$, for any frequent episode P, we have

$$sup(P) \begin{cases} = sup(P) & \text{if } P \in \mathbb{P} \\ \in [sup(P'), (1+\theta)sup(P')] & \text{if } P \notin \mathbb{P} \end{cases} \tag{2}$$

where $P' \in \mathbb{P}$ and $P' \supset P$. That is to say we can use $sup(P')$ to approximate $sup(P)$ with guaranteed maximal error bound θ for each frequent episode P outside of $\mathbb{P}$.

Definition 2.14. ***(The Mining Problem)*** *Given S, $\min\_sup$ and θ, the problem is discovering a nDaCF set $\mathbb{P}$ from a given complex sequence S.*

3. **Mining the nDaCF Sets.** The basic idea of the mining process is to enumerate each candidate episode in a prefix tree level by level. In the enumeration procedure, a series of effective strategies is introduced to prune derivable non-closed episodes, and infrequent episodes are pruned. Unpruned frequent episodes are inserted into the nDaCF set. All frequent episodes outside the nDaCF set can be derived from it. Example 3.1 is used to illustrate the checking and pruning operations along with the execution of the proposed method.

Example 3.1. *Given a sequence as shown in Figure 1(b), $\min\_sup = 3$ and $\theta = 1/3$, the proposed method is used to discover a nDaCF set from the sequence.*

3.1. **Computation of support.** The essence of computing $LMaxnR\text{-}O\text{-}freq(P)$ is to construct $LMaxnR\text{-}O(P)$. Given S and $P = \langle p_1 p_2 \dots p_m \rangle$, $LMaxnR\text{-}O(P)$ can be constructed recursively as follows. If $m = 1$, let $LMaxnR\text{-}O(P) = O(P)$. If $m > 1$, $LMaxnR\text{-}O(P) = Join(LMaxnR\text{-}O(pre(P)), LMaxnR\text{-}O(tail(P)))$, where $pre(P) = \langle p_1\ p_2 \dots p_{m-1} \rangle$ and $tail(P) = \langle p_m \rangle$. As shown in Figure 2, the basic idea of the $Join$ procedure is, for each occurrence o'_j in $LMaxnR\text{-}O(pre(P))$, to find the leftmost occurrence o^*_r in $LMaxnR\text{-}O(tail(P))$ that comes after the last timestamp of o'_j, and to insert $o_j = o'_j \circ o^*_r$ into $LMaxnR\text{-}O(P)$. For example, given sequence S shown in Figure 1(b) and $P = \langle A(AC) \rangle$, we have $pre(P) = \langle A \rangle$, $tail(P) = \langle AC \rangle$, $LMaxnR\text{-}O(pre(P)) = \{\langle 1 \rangle, \langle 2 \rangle, \langle 3 \rangle, \langle 4 \rangle, \langle 6 \rangle\}$, $LMaxnR\text{-}O(tail(P)) = \{\langle 2 \rangle, \langle 3 \rangle, \langle 4 \rangle, \langle 6 \rangle\}$, $LMaxnR\text{-}O(S, P) = Join(\{\langle 1 \rangle, \langle 2 \rangle, \langle 3 \rangle, \langle 4 \rangle, \langle 6 \rangle\}, \{\langle 2 \rangle, \langle 3 \rangle, \langle 4 \rangle, \langle 6 \rangle\}) = \{\langle 1, 2 \rangle, \langle 2, 3 \rangle, \langle 3, 4 \rangle, \langle 4, 6 \rangle\}$. Note that in implementation, it is not necessary to record the whole $LMaxnR\text{-}O(S, P)$ if $P.L > 1$. We only need to keep the last timestamp of every occurrence since the former timestamps can be obtained from $pre(P)$, e.g., $LMaxnR\text{-}O(S, P)$ is recorded as $\{\langle 2 \rangle, \langle 3 \rangle, \langle 4 \rangle, \langle 6 \rangle\}$.

3.2. **Search strategy.** Breadth-first search strategy is adopted to generate episodes, so that non-closed frequent episodes can be pruned as early as possible. To begin with, the input sequence is scanned to find frequent size-1 episodes (e.g., in Example 3.1, the set is $\{A, C\}$), and then for each frequent size-1 episode, conduct $\diamond^R_V$ and $\diamond^R_S$ to generate size-2 episodes. The episodes are extended level by level until larger frequent episodes can no longer be generated. The episodes are stored in an episode enumeration tree. Figure 3 shows the complete episode enumeration tree for Example 3.1 (infrequent episodes are excluded in the tree). Each episode P is stored in a node (nodes and episodes are used

Procedure 1: Join($LMaxnR\text{-}O(pre(P)), LMaxnR\text{-}O(tail(P))$)

Input : $LMaxnR\text{-}O(pre(P))$ and $LMaxnR\text{-}O(tail(P))$
Output: $LMaxnR\text{-}O(P)$

$start \leftarrow 1$;
for each j=1 to $|LMaxnR\text{-}O(pre(P))|$ **do**
    for each r=start to $|LMaxnR\text{-}O(tail(P))|$ **do**
        if $o^*_r[1] > o'_j[m-1]$ **then**
            $o_j \leftarrow o'_j \circ o^*_r = \langle o'_j[1], o'_j[2], ..., o'_j[m-1], o^*_r[1] \rangle$;
            Insert o_j into $LMaxnR\text{-}O(P)$;
            $start \leftarrow r+1$;
return $LMaxnR\text{-}O(P)$

FIGURE 2. The Join procedure

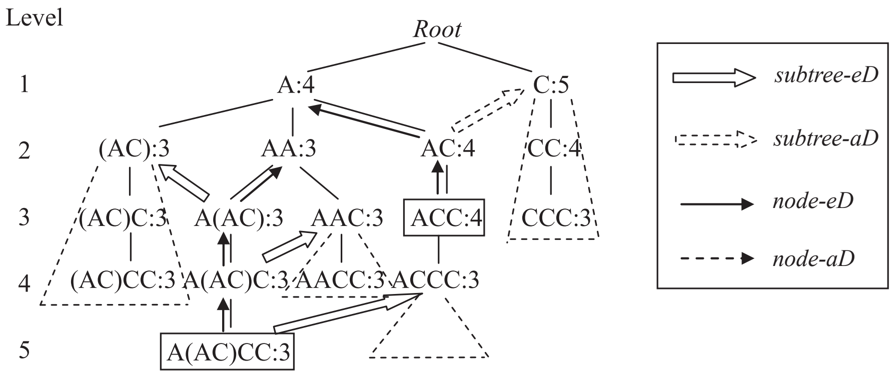


FIGURE 3. Episode enumeration tree and pruned nodes

interchangeably in the rest of this paper), and all P-prefixed episodes are included in the P-rooted subtree. Any node P is extended as follows: (1) for each possible item α, do $P \diamond^R_V \alpha$ (if $\alpha > \beta$, do the extension with α first), and (2) for each possible item α, do $P \diamond^R_H \alpha$ (if $\alpha > \beta$, do the extension with α first). The extension strategy guarantees that no episodes are missed and no duplicate episodes are generated.

3.3. **Checking and pruning strategies.** Since frequency measure $LMaxnR\text{-}O\text{-}freq$ is anti-monotonic [17], infrequent episodes can be safely pruned by the downward pruning strategy [12]. This means an episode is not extended if it is infrequent, and an episode is infrequent if any of its sub-episodes is infrequent. For frequent episodes, a series of strategies are introduced to check and prune derivable episodes. The derivable episodes are divided into four categories.

1. Subtree-eD episodes: episode P is a subtree-eD episode if there exists $P' = P \diamond \alpha$, such that any P-prefixed episode, $P \oplus Q$, can be exactly derived from the supper-episode $P' \oplus Q$, i.e., $eD(P \oplus Q, P' \oplus Q)$.
2. Subtree-aD episodes: episode P is a subtree-aD episode if there exists $P' = P \diamond \alpha$, such that any P-prefixed episode, $P \oplus Q$, can be approximately derived from the supper-episode $P' \oplus Q$, i.e., $aD(P \oplus Q, P' \oplus Q)$.
3. Node-eD episodes: episode P is a node-eD episode if there exists a supper-episode $P' = P \diamond \alpha$ such that $eD(P, P')$.
4. Node-aD episodes: pattern P is a node-aD pattern if there exists a supper-pattern $P' = P \diamond \alpha$ such that $aD(P, P')$.

Any subtree with subtree-eD episode P as the root can be pruned since all P-prefixed episodes are neC. A node-aD episode indicates that only itself is naC and can be pruned. The subtrees of a subtree-aD episode and a node-aD episode can be pruned if they satisfy additional conditions. For the above four classes of episodes, four corresponding checking and pruning strategies are introduced.

Theorem 3.1. ***(Subtree-eD Checking)*** *Given S and P, let $LMaxnR\text{-}O(P) = \{o_l\}$ $(1 \le l \le sup(P))$, $P' = P \diamond \alpha$, and $LMaxnR\text{-}O(P') = \{o'_l\}$ $(1 \le l \le sup(P'))$. For any P-prefixed episode, $P \oplus Q$, $eD(P \oplus Q, P' \oplus Q)$ holds, i.e., $P \oplus Q$ is non-exactly-closed, in one of the following three cases for any Q (Note: Subtree-eD-Checking(P) returns true if such a P' exists).*

1. *If $\exists P' = P \diamond_H^j \alpha$ s.t. (1) $sup(P) = sup(P')$; and (2) $o'_l[P'.L] \le o_l[P.L]$ for all $l = 1, \ldots, sup(P)$.*
2. *If $P.L = 1$, $\exists P' = P \diamond_V^1 \alpha$, s.t. $sup(P) = sup(P')$.*
3. *If $P.L > 1$, $\exists P' = P \diamond_V^j \alpha$, s.t. (1) $j \ne P.L$; (2) $sup(P) = sup(P')$ and (3) $o'_l[j] = o_l[j]$ for all $l = 1, \ldots, sup(P)$.*

For instance, as shown in Figure 3, in Example 3.1, $P = \langle (AC) \rangle$ is a *subtree-eD* episode, because there exists $P' = P \diamond_H^1 A = \langle A(AC) \rangle$ that satisfies the conditions in Case 1 of Theorem 3.1. Similarly, $\langle AAC \rangle$ and $\langle ACCC \rangle$ are subtree-eD episodes.

Theorem 3.2. ***(Subtree-aD Checking)*** *Given S, P, $\min\_sup$ and θ, any P-prefixed episode, $P \oplus Q$, must be non-approximately-closed (since $J(sup(P \oplus Q), sup(P' \oplus Q)) \le \theta$), if $\exists P' = P \diamond_H^j \alpha$ such that (1) $sup(P) - sup(P') \le \theta \times \min\_sup$, and (2) for $\forall$ $o'_l \in LMaxnR\text{-}O(P')$ and the corresponding $o_l \in LMaxnR\text{-}O(P)$, $o'_l[P'.L] \le o_l[P.L]$ (Note: Subtree-aD-Checking(P) returns true if such a P' exists).*

For instance, as shown in Figure 3, in Example 3.1, $P = \langle C \rangle$ is a subtree-aD episode because there exists $P' = \langle AC \rangle$ such that the conditions in Theorem 3.2 are satisfied. Please refer to the appendix for the proofs of Theorems 3.1 and 3.2.

Definition 3.1. ***(Node-eD Checking)*** *If $P' = P \diamond \alpha$ exists such that $sup(P) = sup(P')$, then $eD(P, P')$ holds, i.e., P is neC (return true), else P is eC (return false).*

Definition 3.2. ***(Node-aD Checking)*** *Given S, P and θ, it returns true if $\exists P' = P \diamond \alpha$, such that $J(sup(P), sup(P')) \le \theta$.*

Four pruning strategies are introduced for the four classes of derivable episodes.

Definition 3.3. ***(Subtree-eD Pruning)*** *If Subtree-eD-Checking(P) returns true, then P is called a subtree-eD episode, i.e., the subtrees of P can be pruned since any P-prefixed episodes, $P \oplus Q$, can be derived exactly from a corresponding P'-prefixed episodes, $P' \oplus Q$.*

For instance, the subtrees of $\langle (AC) \rangle$ can be pruned because $\langle (AC) \rangle$ is a subtree-eD episode.

Definition 3.4. ***(Node-eD Pruning)*** *Episode P is pruned if there exists P' such that Node-eD-Checking(P) = true.*

Exactly derivable episodes can be pruned directly since eD satisfies transitivity. In contrast, not all approximately derivable episodes can be pruned straightaway, since aD does not satisfy the transitivity property. Actually, P can be pruned if there exists P' such that not only $aD(P, P')$ but also $J(P.\max\text{-}aD\text{-}sup, sup(P')) \le \theta$, where, $P.\max\text{-}aD\text{-}sup$ denotes the maximal support of the episodes that need to be approximately derived from P, *P.aD-episodes*.

Two aD pruning strategies are defined as follows.

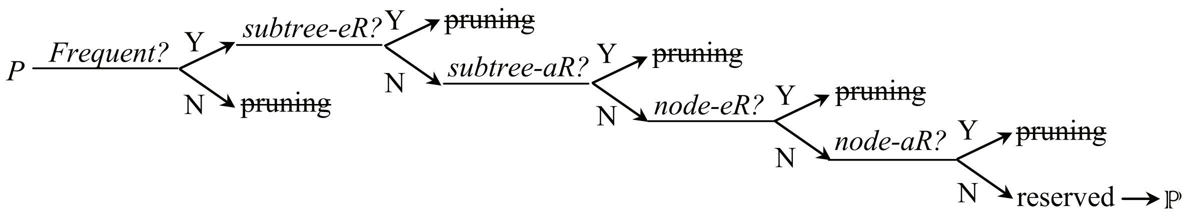


Figure 4. Checking and pruning process

Definition 3.5. ***(Subtree-aD Pruning)*** *If* $\exists P' = P \diamond_H^j \alpha$, *such that (1) Subtree-aD-Checking(P)=true and (2)* $J(P.\max\text{-}aD\text{-}sup,\ sup(P')) \leq \theta$ *if* $P.\max\text{-}aD\text{-}sup > 0$, *or (3)* $P.\max\text{-}aD\text{-}sup = 0$, *then P is called subtree-aD node, i.e., P and its subtrees can be pruned since any P-prefixed episode* $P \oplus Q$ *can be approximately derived from* $P' \oplus Q$, *and any episode in P.aD-episodes can be approximately derived.*

For instance, in Figure 3, for $P = \langle C \rangle$, there exists $P' = \langle AC \rangle$ such that Subtree-aD-Checking(P) = *true*. Consequently, the subtrees of $\langle C \rangle$ can be pruned. Since $\langle AC \rangle$ is chosen to approximately represent the subtrees of $\langle C \rangle$, $\langle AC \rangle.\max\text{-}aD\text{-}sup = sup(C) = 5$.

Definition 3.6. ***(node-aD Pruning)*** *Episode P is pruned if there exists P′ such that (1) Node-aD-Checking(P) = true, and (2)* $J(P.\max\text{-}aD\text{-}sup,\ sup(P')) \leq \theta$ *when* $P.\max\text{-}aD\text{-}sup > 0$ *or (3)* $P.\max\text{-}aD\text{-}sup = 0$.

For instance, in Example 3.1, for $P = \langle AC \rangle$, there exists $P' = \langle AAC \rangle$ such that node-aD-checking(P)=true. However, $\langle AAC \rangle$ cannot be used to approximately represent $\langle AC \rangle$, because $J(\langle AC \rangle.\max\text{-}aD\text{-}sup,\ sup(\langle AAC \rangle)) = (5-3)/5 = 0.4 > 1/3 = \theta$.

The above checking and pruning strategies are conducted as shown in Figure 4. The subtree-eD-pruning and subtree-aD-pruning are conducted first to prune derivable episodes as early as possible.

3.4. **nDaCF-Miner.** The proposed nDaCF-Miner is described in Figure 5. The basic idea is to enumerate episodes level by level, and prune infrequent episodes and derivable non-closed episodes. To begin with, frequent items are found as the basis for episode enumeration. In Line 2, frequent length-1 episodes are found as the $Tail$ set to be used in the Join procedure. From Line 4 to Line 33, episodes are enumerated level by level and the five pruning strategies are conducted. Episode extension is embedded in the checking process. For new episodes with a length of more than 1, the Join procedure is conducted to compute their supports with infrequent episodes pruned straightaway. The episode extension in lines 23 and 30 only creates the episodes that have not been generated in the previous checking process. In lines 12-15, 18-19 and 25-28, $P.\max\text{-}aD\text{-}sup$ is transmitted to P' when P' is selected to approximately derive P.

Example 3.1 is used to illustrate the execution of nDaCF-Miner. First, find frequent item set, $\{\langle A \rangle : 5, \langle C \rangle : 5\}$ and $Tail = \{\langle A \rangle : 5, \langle C \rangle : 5, \langle (AC) \rangle : 3\}$, where the number attached to each episode is its *support*.

As shown in Figure 3, when $l = 1$, there exists $P' = \langle AC \rangle$ so that $P = \langle C \rangle$ is a subtree-aD episode, and the subtree of P can be pruned according to Definition 3.5. Therefore, $\langle AC \rangle$ is selected to approximately represent the subtree of $\langle C \rangle$, and $\langle AC \rangle.\max\text{-}aD\text{-}sup = sup(\langle C \rangle) = 5$. Episode $\langle A \rangle$ can be exactly derived from $\langle AC \rangle$.

When $l = 2$, there exists $P' = \langle A(AC) \rangle$ so that $P = \langle (AC) \rangle$ is a subtree-eD episode, and the subtree of P can be pruned. Episode $\langle AA \rangle$ can be exactly derived from $\langle A(AC) \rangle$. Episode $\langle ACC \rangle$ is selected to exactly represent $\langle AC \rangle$. Since $\langle AC \rangle.\max\text{-}aD\text{-}sup = 5 > 0$, it is transmitted to $\langle ACC \rangle$, i.e., $\langle ACC \rangle.\max\text{-}aD\text{-}sup = 5$.

Algorithm 1: nDaCF-Miner(S, min_sup, θ, Tr, $\mathbb{P}$)

```
Input  : S, min_sup, θ
Output: Pattern set ℙ, and search tree Tr
Scan S to find frequent items and insert them into the 1st level of Tr;
Find frequent length-1 patterns as the Tail set;
l ← 1;
while Level l is not empty do
    for each pattern P in Level l do
        if Subtree-eD-checking(P)=true then
            P.type ← subtree-eD; Set a subtree-eD link from P′ to P;
            P′.max-aD-sup ← P.max-aD-sup; Prune the subtrees of P;
        else if the conditions in Theorem 3.2 are satisfied then
            P.type ← subtree-aD; Set a subtree-aD link from P′ to P;
            Prune the subtrees of P;
            if P.max-aD-sup > 0 then
                P′.max-aD-sup=P.max-aD-sup;
            else
                P′.max-aD-sup=sup(P);
        else if Node-eD-Checking(P)=true then
            P.type ← node-eD; Extend pattern P;
            if P.max-aD-sup > 0 then
                P′.max-aD-sup=P.max-aD-sup;
                Set a node-eD link from P′ to P;
            Delete node P;
        else if the conditions in Definition 3.2 are satisfied then
            P.type ← node-aD; Extend pattern P;
            Select a P′ of the maximal support, and set a node-aD link from P′ to P;
            if P.max-aD-sup > 0 then
                P′.max-aD-sup=P.max-aD-sup;
            else
                P′.max-aD-sup=sup(P);
        else
            P.type ← reserved; ℙ ← ℙ ∪ {P}; Extend pattern P;
            if P.max-aD-sup = 0 and P.type ≠ subtree-aD then
                Delete node P;
    l ← l + 1;
return ℙ;
```

FIGURE 5. nDaCF-Miner

When $l = 3$, $\langle A(AC)\rangle$ can be exactly derived from $\langle A(AC)C\rangle$. There exists $P' = \langle A(AC)C\rangle$ such that $P = \langle AAC\rangle$ is a subtree-eD episode, and the subtree of P can be pruned. For $P = \langle ACC\rangle$, according to Definition 3.6, there exist no P' at Level 4 such that Conditions (1) and (2) are satisfied. Therefore, $\langle ACC\rangle$ is kept in the resulting set.

When $l = 4$, $\langle A(AC)C\rangle$ can be exactly derived from $\langle A(AC)CC\rangle$. For $P = \langle ACCC\rangle$, there exists $P' = \langle A(AC)CC\rangle$ such that P is a subtree-eD episode.

When $l = 5$, there is only one episode $\langle A(AC)CC\rangle$, and no larger FEs can be generated. Consequently, $\langle A(AC)CC\rangle$ is inserted into the resulting set, and the extension terminates.

Finally, the found nDaCF set is $\mathbb{P} = \{\langle ACC\rangle : 4, \langle A(AC)CC\rangle : 3\}$, which consists of only two episodes. In contrast, the complete set contains 16 episodes as shown in Figure 3. It shows that the found nDaCF set is highly condensed.

Based on the nDaCF set and the derivation relationship in the tree in Figure 3, 14 FEs outside $\mathbb{P}$ can be derived from Level 5 to Level 1. When $l = 5$, from $\langle A(AC)CC\rangle : 3$, it can be derived that $\langle A(AC)C\rangle : 3$ and $\langle ACCC\rangle : 3$. When $l = 4$, from $\langle A(AC)C\rangle : 3$, $\langle A(AC)\rangle : 3$ can be derived; from $\langle A(AC)CC\rangle : 3$ and the subtree-eD link from $\langle A(AC)C\rangle : 3$ to $\langle AAC\rangle : 3$, $\langle AACC\rangle : 3$ can be derived. When $l = 3$, from $\langle A(AC)\rangle : 3$ (plus $\langle A(AC)C\rangle : 3$ and $\langle A(AC)CC\rangle : 3$), it can be derived that $\langle (AC)\rangle : 3$, $\langle (AC)C\rangle : 3$ and $\langle (AC)CC\rangle : 3$. Episode $\langle AA\rangle : 3$ can de derived from $\langle A(AC)\rangle : 3$, and $\langle AC\rangle : 4$ can be derived from $\langle ACC\rangle : 4$. When $l = 2$, $\langle A\rangle : 4$ can be derived from $\langle AC\rangle : 4$; from $\langle AC\rangle : 4$, it can be derived that $\langle C\rangle : 5$ (since it is kept in the tree); from $\langle ACC\rangle : 4$, it can be derived that $sup(\langle CC\rangle) \in [sup(\langle ACC\rangle), (1 + 1/3)sup(\langle ACC\rangle)] = [4, 5.33]$. Similarly, we have $sup(\langle CCC\rangle) \in [sup(\langle ACCC\rangle), (1 + 1/3)sup(\langle ACCC\rangle)] = [3, 4]$.

4. **Experimental Evaluation.** To evaluate the performance of nDaCF-Miner, we generated a series of synthetic sequences and performed nDaCF-Miner on the sequences. The performance of nDaCF-Miner was compared with that of the typical mining approach $MINEPI$ [7] and two of the latest ones, $EMMA$ [10] and *Clo_episode* [11]. The algorithms were implemented in Java, and were performed on a computer with an Intel processor at 1.86 Ghz and a RAM of 2 Gb, running Windows XP.

We designed a synthetic sequence generator. The generator accepts four major parameters: $|S|$ (the number of data elements contained in S), $|I|$ (the number of distinct items), E (average number of items contained in a data element) and w (average window width). Since $MINEPI$ and *Clo_episode* can only process single-item sequences, we generated two groups of sequences; one contains five complex sequences, denoted as S10I500E1W10, and the other contains five single-item sequences, denoted as S10I500E6W10, with each number behind a parameter equaling the value of the parameter. Each experiment was conducted on five sequences to obtain an average value. In the experiments, we evaluated and compared three major components: accuracy, compactness and efficiency.

4.1. **Accuracy.** We use $\mathbb{P}_0$ to denote the set discovered from a single-item sequence under the measurement $LMaxnR$-O-$freq$, and use $\mathbb{P}_1$, $\mathbb{P}_2$ and $\mathbb{P}_3$ to denote the sets discovered from the same sequence under three typical frequency measures: $fixed$-win-$freq$ [5], mo-$freq$ [6] and $distinct$-$bound$-st-$freq$ [10]. The restriction of fixed/max window width is specified as $w = 10$. Taking $\mathbb{P}_0$ as the standard result, the accuracy of $\mathbb{P}_k$ ($1 \leq k \leq 3$) w.r.t. $\mathbb{P}_0$ is evaluated by the difference between $\mathbb{P}_k$ and $\mathbb{P}_0$. The difference is captured by three classes of episodes [17].

1. Missed (M) episodes — episodes missed by $\mathbb{P}_k$, i.e., the episodes in $\mathbb{P}_0$, but not in $\mathbb{P}_k$.
2. False (F) episodes — episodes in $\mathbb{P}_k$, but not in $\mathbb{P}_0$.
3. Inaccurate (I) episodes — episodes in $\mathbb{P}_k \cap \mathbb{P}_0$ with different frequencies in $\mathbb{P}_k$ and $\mathbb{P}_0$.

The inaccuracy of $\mathbb{P}_k$ w.r.t. $\mathbb{P}_0$ is evaluated by three ratios below.

$$R_M^k = \frac{|\mathbb{P}_0 \backslash \mathbb{P}_k|}{|\mathbb{P}_0|} \tag{3}$$

$$R_F^k = \frac{|\mathbb{P}_k \backslash \mathbb{P}_0|}{|\mathbb{P}_0|} \tag{4}$$

$$R_I^k = \frac{|\{P | P \in \mathbb{P}_k \cap \mathbb{P}_0, sup_k(P) \neq sup_0(P)\}|}{|\mathbb{P}_0|} \tag{5}$$

where $1 \leq k \leq 3$, and $sup_k(P)$ ($sup_0(P)$) represents the frequency of episode P in $\mathbb{P}_k$ ($\mathbb{P}_0$). Let total inaccuracy $TI = R_M^k + R_F^k + R_I^k$.

TABLE 2. A comparison between P_k and P_0 for frequent episodes

	R_M^k	R_F^k	R_I^k	TI^k
$k = 1$	0.003	0.124	0.313	0.440
$k = 2$	0.082	0.000	0.164	0.246
$k = 3$	0.002	0.062	0.212	0.276

To evaluate the inaccuracies of the results discovered under different frequency measures, we performed the three previous approaches and the proposed method on the group of single-item sequences, S10I500E1W10. For *fixed-win-freq* and *distinct-bound-st-freq* adopted by *MINEPI* [6] and *EMMA* [10], both the window-width restrictions *fixed-win* and max-*win* were specified as 10. The parameter min_*sup* was specified as 500. The inaccuracies are evaluated by the three ratios. Table 2 shows the average ratios for inaccuracies, with each value being an average obtained from the five tests. The results in Column 2 show that $\mathbb{P}_k$ $(k = 1, 2, 3)$ missed some frequent episodes because some non-redundant occurrences were missed in the counting. In Column 3, $R_F^2 = 0$ indicates that no false (infrequent) episodes were included in the discovered sets under *mo-freq* since no redundant occurrences are over-counted. Whereas R_F^1 and R_F^3 are neither zero as redundant occurrences are over-counted under *fixed-win-freq* and *distinct-bound-st-freq*. Column 5 in Table 2 demonstrates that the found sets under *fixed-win-freq*, *mo-freq* and *distinct-bound-st-freq* include inaccuracies.

4.2. **Compactness.** To evaluate the compactness of found sets, three types of sets, *all*, *closed* and nDaCF, are discovered from simple sequences and complex sequences, with comparison between their sizes.

Figures 6(a) and 6(b) show the comparisons of compactness among the three sets discovered from single-item sequences S10I500E1W10 and complex sequences S10I500E6W10 respectively when θ varies from 0 to 0.1. In Figure 6(a), the *all* set is discovered by *MINEPI* [7], and the *closed* set is discovered by *clo_episode* [11]. In Figure 6(b), the *all* set is discovered by *EMMA* [10], and the *closed* set is discovered by *nDaCF-Miner-NP* (the nDaCF-Miner without pruning strategies) with $\theta = 0$. The results demonstrate that *nDaCF* is much more condensed than *all* and *closed* (*all* has a magnitude of 10^4, *closed* has a magnitude of 10^3, and *nDaCF* has a magnitude of 10^2 when θ is somewhat high). In addition, *nDaCF* achieves higher compactness when θ rises. It should be noted that different frequency measures are adopted by different mining approaches. The *clo_episode* adopts *mo-freq*, *nDaCF-Miner* uses *LMaxnR-O-freq*, and *EMMA* uses *distinct-bound-st-freq*. With more restricted constraints, *mo-freq*(P) is normally less than *LMaxnR-O-freq*(P). In general, *distinct-bound-st-freq*(P) is greater than *LMaxnR-O-freq*(P). Therefore, when $\theta = 0$, *nDaCF* is slightly larger than *closed* in Figure 6(a), and *nDaCF* is smaller than *closed* in Figure 6(b).

Figures 7(a) and 7(b) show the comparisons of compactness on single-item sequences S10I500E1W10 and complex sequences S10I500E6W10 respectively when min_*sup* varies from 400 to 700 and $\theta = 0.6$. Note that y-axis uses a logarithmic scale in Figures 6 and 7.

4.3. **Efficiency.** For single-item sequences, the efficiency of nDaCF-Miner is compared with that of *MINEPI* [7] and *Clo_episode* [11] when θ and min_*sup* vary. For complex sequences, the efficiency is compared between nDaCF-Miner and *EMMA* [10] when θ and min_*sup* vary. Figures 8(a) and 8(b) show the runtime comparison on single-item sequences S10I500E1W10 when min_*sup* and θ vary respectively. Figures 9(a) and 9(b)

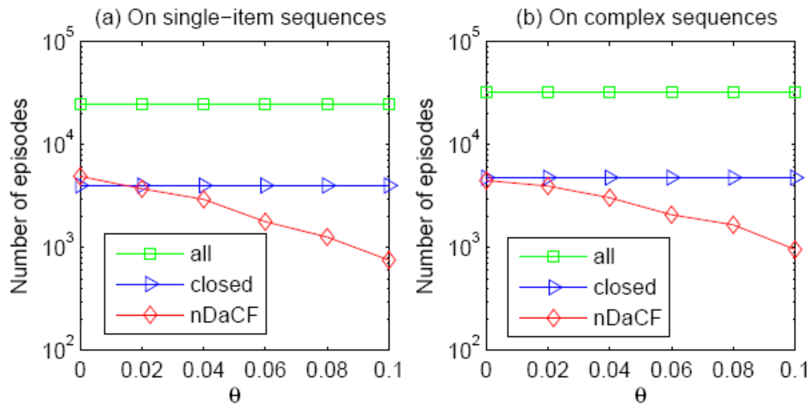


FIGURE 6. Compactness comparison with varying θ

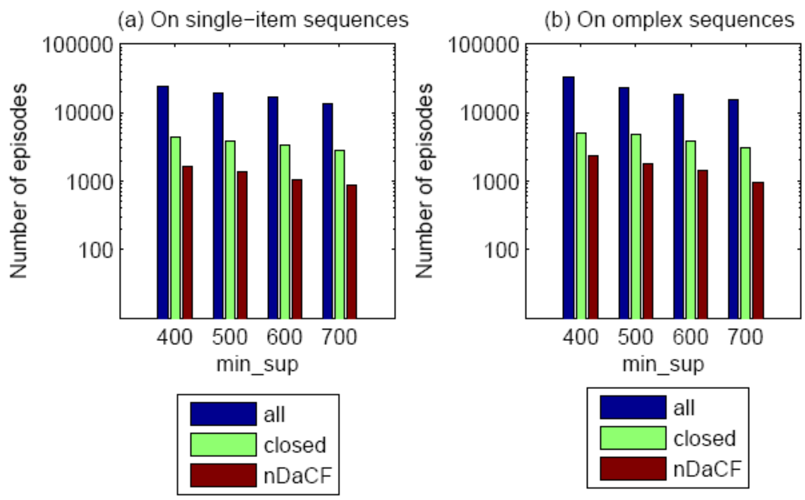


FIGURE 7. Compactness comparison with varying min $\_sup$

show the runtime comparison on complex sequences S10I500E6W10 when min $\_sup$ and θ vary respectively. In Figure 8(a) and Figure 9(a), $\theta = 0.06$, and in Figure 8(b) and Figure 9(b), min $\_sup = 600$. It can be seen from Figures 8 and 9 that (1) nDaCF-Miner is more efficient than $MINEPI$, (2) nDaCF-Miner is competitive with $EMMA$ and $Clo\_episode$ in terms of efficiency when θ is low, and (3) nDaCF-Miner outperforms $EMMA$ and $Clo\_episode$ when θ is high to some degree.

The high efficiency of nDaCF-Miner benefits from one scan of the sequence and also from effective pruning strategies. $MINEPI$ needs n scans of the sequence and generates large numbers of candidate episodes, e.g., 54836 candidates are generated when min $\_sup =$ 600. Consequently, $MINEPI$ is the lest efficient. $Clo\_episode$ also needs n scans of the sequence. $Clo\_episode$ is faster than $MINEPI$ since it generates fewer candidates when non-closed episodes are pruned. In contrast, nDaCF-Miner needs only one scan of the sequence and prunes large numbers of candidates. The effectiveness of pruning heavily depends on the error bound θ. When θ is high to some degree, nDaCF-Miner is faster than either $clo\_episode$ or $EMMA$.

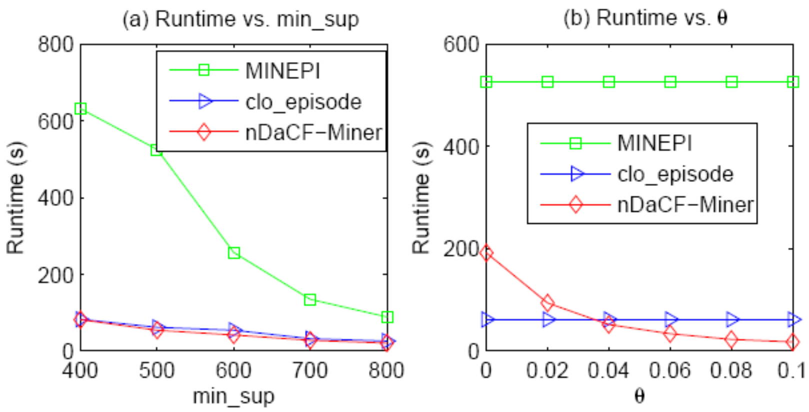


FIGURE 8. Runtime comparison on single-item sequences

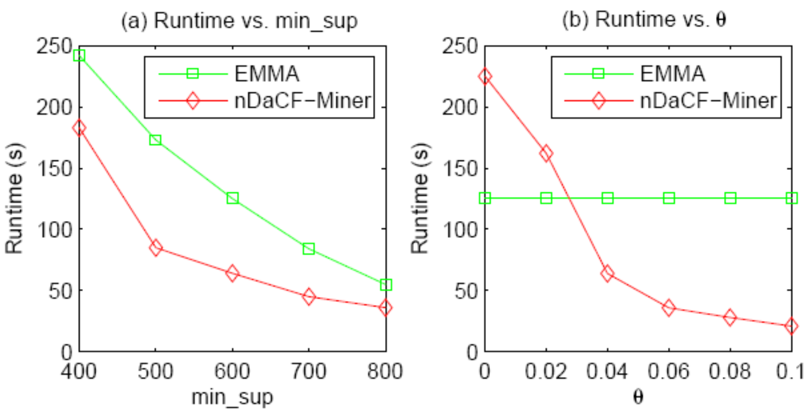


FIGURE 9. Runtime comparison on complex sequences

5. **Conclusion and Discussion.** In this paper, based on a new frequency measure, *LMaxnR-O-freq* [17], we defined a condensed set of episodes, $nDaCF$, with more accurate frequencies. We further introduced a series of pruning strategies, and developed nDaCF-Miner for discovering nDaCF sets from complex sequences. The experimental results showed that when error bound θ is high to some degree, the found $nDaCF$ sets are able to compress the complete sets by a magnitude of 100 and compress the closed sets by a magnitude of 10. This demonstrates that nDaCF-Miner is more efficient than previous mining approaches such as $MINEPI$ [6], $EMMA$ [10] and *Clo_episode* [11].

Both Table 1 and the experimental results have shown that the proposed method outperforms previous approaches. On the one hand, compared with approaches intended for simple sequences [5-9,11], nDaCF-Miner has wider applications because it accepts both complex sequences and simple sequences, and can find more accurate and condensed results. On the other hand, nDaCF-Miner has overcome three deficiencies of $EMMA$ by adopting a new frequency measure and obtaining more accurate and condensed FE sets. In addition, it is faster than previous approaches when the error bound is somewhat high.

This research is of both theoretical and practical significance. In practical terms, the proposed method can be used to discover FEs from various real-world sequences more efficiently and more effectively. Furthermore, accurate and significant episode associations can be generated from the found nDaCF sets directly [21]. These episode associations represent the relationships among different elements in real-world sequences, such as DNA sequences, Web-click sequences and stock sequences, and thus, the techniques can be applied to protein analysis, Web usage analysis, Web intrusion detection, and stock price analysis and forecasting. In theoretical terms, more effective and more accurate discriminative episodes, sequence clustering and sequence classifiers (different from traditional clustering [20] and classification in transactional data) could be constructed based on the nDaCF sets of higher accuracy and compactness [21].

However, the proposed method has some deficiencies which need to be explored in future work. Firstly, it is not applicable to data streams. Discovery of nDaCF sets over streams is a more challenging problem with stricter requirements such as one-pass, overtime feedback and limited space expense. A key to this problem is updating the found nDaCF set adaptively and quickly when a new data element arrives. Secondly, the method should be adapted to processing time series data [22], where sequence elements may be numerical values instead of items. One possible way is to discretise numerical values into different levels which can be represented by items.

REFERENCES

[1] Q. Zhang, R. Zhang and B. Wang, DNA sequence sets design by particle swarm optimization algorithm, *International Journal of Innovative Computing, Information and Control*, vol.5, no.8, pp.2249-2255, 2009.

[2] G. T. Raju, P. S. Satyanarayana and L. M. Patnaik, Knowledge discovery from web usage data: Extraction and applications of sequential and clustering patterns – A survey, *International Journal of Innovative Computing, Information and Control*, vol.4, no.2, pp.381-389, 2008.

[3] G. Fang, J. Lin, K. Chin and C. Lee, Software integration for applications with audio/video stream, *International Journal of Innovative Computing, Information and Control*, vol.6, no.3(B), pp.1421-1433, 2010.

[4] D. H. Kim, Packet scheduling algorithm for realistic traffic model of real-time video streaming service in ofdma system with integrated traffic scenario, *International Journal of Innovative Computing, Information and Control*, vol.6, no.11, pp.4797-4811, 2010.

[5] H. Mannila, H. Toivonen and A. Verkamo, Discovering frequent episodes in sequences, *Proc. of the 1st Int. Conf. Knowledge Discovery and Data Mining*, pp.210-215, 1995.

[6] H. Mannila and H. Toivonen, Discovering generalized episodes using minimal occurrences, *Proc. of the 2nd Int. Conf. Knowledge Discovery and Data Mining*, pp.146-151, 1996.

[7] H. Mannila, H. Toivonen and A. I. Verkamo, Discovery of frequent episodes in event sequences, *Data Mining and Knowledge Discovery*, vol.1, no.3, pp.259-289, 1997.

[8] K. Iwanuma, R. Ishihara, Y. Takano and H. Nabeshima, Extracting frequent subsequences from a single long data sequence: A novel anti-monotonic measure and a simple on-line algorithm, *Proc. of the 3rd IEEE Int. Conf. Data Mining*, pp.186-193, 2005.

[9] S. Laxman, P. Sastry and K. Unnikrishnan, A fast algorithm for finding frequent episodes in event streams, *Proc. of the 13th Int. Conf. Knowledge Discovery and Data Mining*, pp.410-419, 2007.

[10] K. Huang and C. Chang, Efficient mining of frequent episodes from complex sequences, *Information Systems*, vol.33, no.1, pp.96-114, 2008.

[11] W. Zhou, H. Liu and H. Cheng, Mining closed episodes from event sequences efficiently, *Proc. of the 14th Pacific-Asia Conf. Knowledge Discovery and Data Mining*, pp.310-318, 2010.

[12] R. Agrawal, T. Imielinski and A. Swami, Mining association rules between sets of items in large databases, *Proc. of ACM-SIGMOD Int. Conf. Management of Data*, pp.207-216, 1993.

[13] R. J. Bayardo, Efficiently mining long patterns from databases, *Proc. of ACMSIGMOD Int. Conf. Management of Data*, pp.85-93, 1998.

[14] N. Pasquier, R. Bastide, R. Taouil and L. Lakhal, Discovering frequent closed itemsets for association rules, *Proc. of the 7th Int. Conf. Database Theory*, pp.398-416, 1999.

[15] J. Pei, G. Dong, W. Zou and J. Han, On computing condensed frequent pattern bases, *Proc. of IEEE Int. Conf. Data Mining*, pp.378-385, 2002.

[16] R. Shettar and G. T. Shobha, Finding frequent structures in semistructured data, *ICIC Express Letters*, vol.3, no.2, pp.135-140, 2009.

[17] M. Gan and H. Dai, A study on the accuracy of frequency measures and its impact on knowledge discovery in single sequences, *Proc. of Workshops at IEEE the 10th Int. Conf. on Data Mining*, Sydney, Australia, 2010.
[18] B. Ding, D. Lo, J. Han and S. C. Khoo, Efficient mining of closed repetitive gapped subsequences from a sequence database, *Proc. of Int. Conf. Data Engineering*, pp.1024-1035, 2009.
[19] M. Gan and H. Dai, Obtaining accurate frequencies of sequential patterns over a single sequence, *ICIC Express Letters*, vol.5, no.4(B), pp.1461-1466, 2011.
[20] P. Jiang, F. Ren and N. Zheng, A new approach to data clustering using a computational visual attention model, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(A), pp.4597-4606, 2009.
[21] G. Dong and J. Pei, *Sequence Data Mining*, Springer, 2007.
[22] Y. Matsumoto and J. Watada, Knowledge acquisition from time series data through rough sets analysis, *International Journal of Innovative Computing, Information and Control*, vol.5, no.12(B), pp.4885-4898, 2009.

Appendix A. **Proof of Theorem 3.1.**

Proof: Any P-prefixed pattern can be represented as $P \oplus_H Q$ or $P \oplus_V Q$. The main idea of the proof is: for any P-prefixed pattern $P \oplus_H Q$ ($P \oplus_V Q$), we replace P with P' to get $P' \oplus_H Q$ ($P' \oplus_V Q$). Then, we prove $sup(P' \oplus_H Q) = sup(P \oplus_H Q)$ ($sup(P' \oplus_V Q) = sup(P \oplus_V Q)$) to complete the proof. In the proof, assume $P.L = m$ and $Q.L = n$.

Case 1 $P' = P \diamond_H^j \alpha$.

Case 1.1 $P \oplus_H Q$. Let $LMaxnR\text{-}O(P \oplus_H Q) = \{o''_l\}$. For each occurrence $o''_l = \langle o''_l[1], \ldots, o''_l[m+n]\rangle \in LMaxnR\text{-}O(P \oplus_H Q)$ ($l = 1, \ldots, sup(P \oplus_H Q)$), the left part $LP = \langle o''_l[1], \ldots, o''_l[m]\rangle = \langle o_l[1], \ldots, o_l[m]\rangle \in LMaxnR\text{-}O(P)$. In o''_l replacing LP with $\langle o'_l[1], \ldots, o'_l[m+1]\rangle$, we obtain $o^*_l = \langle o'_l[1], \ldots, o'_l[m+1], o''_l[m+1], \ldots, o''_l[m+n]\rangle$. The o^*_l is an occurrence of $P' \oplus_H Q$ in S since $o'_l[m+1] \le o_l[m]$ (Condition (1) in Theorem 3.1) $= o''_l[m] < o''_l[m+1]$. Inserting each o^*_l constructed above into a set, we obtain a set of occurrences of $P' \oplus_H Q$, $O^* = \{o^*_l\}$. According to the way in which o^*_l is generated, O^* is a non-redundant set. Therefore, $sup(P \oplus_H Q) = |LMaxnR\text{-}O(P \oplus_H Q)| = |O^*| \le sup(P' \oplus_H Q)$, and thus, $sup(P \oplus_H Q) \le sup(P' \oplus_H Q)$. On the other hand, according to the anti-monotonicity of support, we have $sup(P \oplus_H Q) \ge sup(P' \oplus_H Q)$ as $P \sqsubset P'$. Consequently, $sup(P \oplus_H Q) = sup(P' \oplus_H Q)$.

Case 1.2 $P \oplus_V Q$. The theorem can be proven in a similar way as in Case 1.1 (the proof is omitted).

The theorem can be proved similarly in cases 2 and 3 (the proofs are omitted).

Appendix B. **Proof of Theorem 3.2.** In the proof, we use the notations: $P' = P \diamond_H^j \alpha$, $P^* = P \oplus \langle\beta\rangle$, $P^{**} = P' \oplus \langle\beta\rangle$, $LMaxnR\text{-}O(P) = \{o_l\}$, $LMaxnR\text{-}O(P') = \{o'_l\}$, $LMaxnR\text{-}O(P^*) = \{o^*_l\}$, $LMaxnR\text{-}O(tail(P^*)) = \{o''_l\}$. In order to prove Theorem 3.2, we introduce Definition B.1 and Lemma B.1.

Definition B.1. *Let* $P^* = P \oplus \langle\beta\rangle$ ($\oplus \in \{\oplus_H, \oplus_V\}$ *and* $\beta \in I$) *and*

$$\begin{aligned} sup(P) - sup(P \oplus \langle\beta\rangle) &= sup(P) - sup(P^*) \\ &= |LMaxnR\text{-}O(S, P)| - |LMaxnR\text{-}O(S, P^*)| \\ &= |O^{(P_{-\beta})}| \end{aligned} \tag{6}$$

where $O^{(P_{-\beta})}$ *denotes the set of occurrences of* P *that cannot be used to construct the corresponding occurrences in* $LMaxnR\text{-}O(S, P^*)$*. According to the combinations of different* $\oplus$ *and* $P.L$, $O^{(P_{-\beta})}$ *is formally defined in three cases.*

1. *(When* $P^* = P \oplus_V \langle\beta\rangle$ *and* $P.L = 1$*)*
$O^{(P_{-\beta})} = \{o_l |\, o_l \in LMaxnR\text{-}O(P) \wedge o_l \notin LMaxnR\text{-}O(\langle\beta\rangle)\}$ ($l = 1, 2, \ldots, sup(P)$).

2. *(When $P^* = P \oplus_V \langle\beta\rangle$ and $P.L > 1$) $pre(P^*) = pre(P)$, $tail(P^*) = tail(P) \oplus_V \langle\beta\rangle$, and $O^{(P_{-\beta})} = \{o_l | o_l \in LMaxnR\text{-}O(P) \wedge \neg\exists o''_r \in LMaxnR\text{-}O(tail(P^*))$ s.t. $o''_r[1] \geq o_l[P.L] \wedge (o''_r[1] > o^*_{l-1}[P^*.L] \vee l = 1)\}$.*
3. *(When $P^* = P \oplus_H \langle\beta\rangle$) $pre(P^*) = P$, $tail(P^*) = \langle\beta\rangle$, and $O^{(P_{-\beta})} = \{o_l | o_l \in LMaxnR\text{-}O(P) \wedge \neg\exists o''_r \in LMaxnR\text{-}O(\langle\beta\rangle)$ s.t. $o''_r[1] \geq o_l[P.L] \wedge (o''_r[1] > o^*_{l-1}[P^*.L] \vee l = 1)\}$.*

Based on Definition B.1, we have the following lemma.

Lemma B.1. *Given P, Q, and an item β, if $\exists P' = P \diamond^j_H \alpha$, such that, for $\forall\ o'_l \in LMaxnR\text{-}O(P')$ and the corresponding $o_l \in LMaxnR\text{-}O(P)$, $o'_l[P'.L] \leq o_l[P.L]$, then the following inequations hold.*

$$sup(P') - sup(P' \oplus \langle\beta\rangle) \leq sup(P) - sup(P \oplus \langle\beta\rangle) \tag{7}$$

$$sup(P') - sup(P' \oplus Q) \leq sup(P) - sup(P \oplus Q) \tag{8}$$

where $\oplus \in \{\oplus_H, \oplus_V\}$, and (7) is a special case of (8).

B.1. **Proof of Lemma B.1.**

Proof: First, we prove (7) is true in three cases of Definition B.1. The main idea is to show that in each case, for every $o'_l \in O^{(P'_{-\beta})}$, an occurrence, o_l, can be constructed such that $o_l \in O^{(P_{-\beta})}$. Thus, we have $|O^{(P'_{-\beta})}| \leq |O^{(P_{-\beta})}|$, i.e., (7). In the following, we only prove the lemma in Case 3. The proofs for Case 1 and 2 are omitted.

Case 3 (When $P^* = P \oplus_H \langle\beta\rangle$) $pre(P^*) = P$, $tail(P^*) = \langle\beta\rangle$. For each $o'_l \in O^{(P'_{-\beta})} = \{o'_l | o'_l \in LMaxnR\text{-}O(P') \wedge \neg\exists o''_r \in LMaxnR\text{-}O(\langle\beta\rangle)$ *s.t.* $o''_r[1] \geq o'_l[P'.L] \wedge (o''_r[1] > o^{**}_{l-1}[P^{**}.L] \vee l = 1)\}$, we construct $o_l = \langle o'_l[1], \ldots, o'_l[j-1], o'_l[j+1], \ldots, o'_l[P'.L]\rangle$ by deleting $o'_l[j]$ from o'_l. Since $P' = P \diamond^j_H \alpha$, o_l constructed above is an occurrence of P. Furthermore, $o_l \in LMaxnR\text{-}O(P)$ since $o'_l[P'.L] \leq o_l[P.L]$. In addition, if there exists no o''_r for o'_l in $O^{(P'_{-\beta})}$, then there exist no o''_r for o_l in $O^{(P_{-\beta})}$. Therefore, we have $|O^{(P'_{-\beta})}| \leq |O^{(P_{-\beta})}|$, i.e., (7).

Inequation (8) can be proved using (7) iteratively. Any $P \oplus Q$ can be represented as $(\ldots(P \oplus \beta_1) \oplus \beta_2) \ldots \oplus \beta_n$, where $\beta_1 \ldots \beta_n$ are the series of items in Q. Using (7) iteratively, we have:

1) $sup(P') - sup(P' \oplus \beta_1 \leq sup(P) - sup(P \oplus \beta_1)$
2) $sup(P' \oplus \beta_1) - sup((P' \oplus \beta_1) \oplus \beta_2) \leq sup(P \oplus \beta_1) - sup((P \oplus \beta_1) \oplus \beta_2)$
$\vdots$
n) $sup((\ldots(P' \oplus \beta_1) \ldots \oplus \beta_{n-1}) - sup((\ldots(P' \oplus \beta_1) \ldots \oplus \beta_n) \leq sup((\ldots(P \oplus \beta_1) \ldots \oplus \beta_{n-1}) - sup((\ldots(P \oplus \beta_1) \ldots \oplus \beta_n)$.

Adding all left sides from 1) to n) and adding all right sides from 1) to n), we obtain $sup(P') - sup((\ldots(P' \oplus \beta_1) \ldots \oplus \beta_n \leq sup(P) - sup((\ldots(P \oplus \beta_1) \ldots \oplus \beta_n) = sup(P') - sup(P' \oplus Q) \leq sup(P) - sup(\oplus Q)$ (Note: the value ($\oplus_H$ or $\oplus_V$) before β_j depends on how the prefix and β_j are concatenated in Q).

B.2. **Proof of Theorem 3.2.**

Proof: The proposition "Any P-prefixed pattern, $R = P \oplus Q$, is non-approximately-closed" $\Longleftrightarrow \exists\ R^* = (P \oplus Q) \diamond \alpha$ s.t. $J(sup(R), sup(R^*)) \leq \theta$.

The main idea of the proof is: we first choose $P' = P \diamond^j_H \alpha$ given in the assumption of this theorem, to construct a particular $R^* = P' \oplus Q = (P \diamond^j_H \alpha) \oplus Q$, then complete the proof by showing that $J(sup(R), sup(R^*)) \leq \theta$.

For $P' = P \diamond^j_H \alpha$, due to Condition (2) given in Theorem 3.2, we have $sup(P \oplus Q) - sup(P' \oplus Q) \leq sup(P) - sup(P')$ ((8) in Lemma B.1). Therefore, $J(sup(P \oplus Q), sup(P' \oplus$

$Q)) = \frac{sup(P \oplus Q) - sup(P' \oplus Q)}{sup(P \oplus Q)} \leq \frac{sup(P) - sup(P')}{sup(P \oplus Q)} \leq \frac{sup(P) - sup(P')}{\min\_sup} \leq \frac{\theta \times \min\_sup}{\min\_sup}$ (Condition (1) given in Theorem 3.2) $= \theta$. Thus, the proof is completed.