Mining Frequent Partite Episodes with Partwise Constraints

Takashi Katoh, Shin-ichiro Tago, Tatsuya Asai, Hiroaki Morikawa, Junichi Shigezumi, and Hiroya Inakoshi

Fujitsu Laboratories Ltd., Kawasaki, 211-8588, Japan
{ kato.takashi_01, s-tago, asai.tatsuya, h.morikawa, j.shigezumi, inakoshi.hiroya } @jp.fujitsu.com

Abstract. In this paper, we study the problem of efficiently mining frequent *partite episodes* that satisfy *partwise constraints* from an input event sequence. Through our constraints, we can extract episodes related to events and their precedent-subsequent relations, on which we focus, in a short time. This improves the efficiency of data mining using trial and error processes. A partite episode of length k is of the form $P = \langle P_1, \ldots, P_k \rangle$ for sets P_i $(1 \le i \le k)$ of events. We call P_i a *part* of P for every $1 \le i \le k$. We introduce the *partwise constraints* for partite episodes P, which consists of *shape and pattern constraints*. A shape constraint specifies the size of each part of P and the length of P. A pattern constraint specifies subsets of each part of P. We then present a backtracking algorithm that finds all of the frequent partite episodes satisfying a partwise constraint from an input event sequence. By theoretical analysis, we show that the algorithm runs in output polynomial time and polynomial space for the total input size. In the experiment, we show that our proposed algorithm is much faster than existing algorithms for mining partite episodes on artificial datasets.

1 Introduction

Episode Mining. One of the most important tasks in data mining is to discover frequent patterns from time-related data. Mannila *et al.* [6] introduced *episode mining* to discover frequent *episodes* in an event sequence. An episode is formulated as a labeled acyclic digraph in which labels corresponding to events and arcs represent a temporal precedent-subsequent relation in an event sequence.

For subclasses of episodes [6, 3], a number of efficient algorithms have been developed. In a previous work [3], we introduced the class of *partite episodes*, which are time-series patterns of the form $\langle P_1, \ldots, P_k \rangle$ for sets P_i $(1 \le i \le k)$ of events, which means that in an input event sequence, every event in P_{i+1} follows every event in P_i for every $1 \le i < k$. For the class of partite episode, we presented an algorithm that finds all of the frequent episodes from an input event sequence.

A partite episode is a richer representation of temporal relationship than a subsequence, which represents just a linearly ordered relation in sequential pattern mining [1, 9]. In particular, the partite episode can represent the precedent-subsequent relation between the sets of events that occur simultaneously (in any order) in a time span.

Main Problem. In the data analysis using trial and error processes, we often want to extract only episodes which include some events we focus on. For example, if we get an episode that means *buying a telescope before buying a PC*, then we will be interested in frequent patterns in the period between the two events.

In this paper, we introduce *partwise constraints* for partite episodes. A partwise constraint consists of a *shape constraint* and a *pattern constraint*. For a partite episode P, a shape constraint specifies the size of each part of P and the length of P. A pattern constraint specifies the subsets of each part of P. Through episode mining with a shape constraint that means the length of any extracted episode is at most three and both the size of the first and the third set are at most one, and a pattern constraint that means the first set includes the event *buying a PC* and the third set includes the event *buying a telescope*, we may obtain a knowledge: *some customers bought a digital camera and a photo printer after buying a telescope before buying a PC.*

We can select the episodes satisfying these constraints in post-processing steps after or during non-constraint episode mining. This approach, however, requires a very long execution time since mining algorithms often outputs a large number of frequent episodes. It is a better idea to use the constraint-based algorithm PPS introduced by Ma *et al.* [5]. PPS efficiently handles a *prefix anti-monotone constraint* [11], which means that if an episode satisfies the constraint, so does every prefix of the episode. Any shape constraint is prefix anti-monotonic, and PPS can extract partite episodes satisfying a given shape constraint in polynomial amortized time per output.

On the other hand, pattern constraints are not always prefix anti-monotonic. For example, a pattern constraint having *buying a telescope* in the first set and *buying a PC* in the third set, which is described above, is not prefix anti-monotonic one. This pattern constraint is not prefix anti-monotonic because there is an episode whose prefix does not satisfy it; It is obvious by considering the episode; *buying a telescope before buying a camera and then buying a PC* and it's prefix episodes.

If we allow an algorithm to add event to any part of the partite episode instead of adding only the tail of the episode, there is a possibility to enumerate episode satisfying any pattern constraint without generating candidate episodes. However, the efficient computation methods of episodes and their occurrence in that enumeration is non-trivial.

Main Results. Our goal is to present an algorithm that extracts frequent partite episodes satisfying a partwise constraint without post-processing steps. Then we show that our algorithm runs in $O(Nsc)$ time per partite episode and $O(Nsm)$ space, where N is the total size of an input sequence, s is the number of event types in the input sequence (alphabet size), c is a constant that depends only on a given partwise constraint, and m is the maximum size of episodes.

Finally, our experimental result shows that our proposed algorithm is 230 times faster than the straightforward algorithm which consists of an algorithm for non-constraint episodes and post-processing steps for partwise constraints.

Related Works. In addition to Ma's research, there have been many studies on episode mining. Méger and Rigotti [7] introduced a different type of constraint called *gapmax* that represents the maximum time gap allowed between two events. Tatti *et al.* [13] and Zhou *et al.* [15] introduced *closed episode mining*. Closed episode mining is another approach to reduce the number of outputs and improve the mining efficiency. Closed episode mining algorithms extract only representative patterns called *closed episodes*, whereas constraint-based algorithms such as our algorithm and Méger's algorithm extract only episodes that satisfy some given conditions on which we focus. Seipel *et al.* [12] applied episode mining to system event logs for network management. Since their class of episodes was a subclass of partite episodes, their analysis would be improved by applying the proposed algorithm in this paper.

2 Partite Episodes

In this section, we introduce partite episodes and the related notions and lemmas necessary for a later discussion. We denote the sets of all natural numbers by $\mathbf{N}$. Then we define ∞ as the special largest number such that $a < \infty$ for all $a \in \mathbf{N}$. For a set $S = \{s_1, \ldots, s_n\}$ $(n \in \mathbf{N})$, we denote the cardinality n of S by $|S|$. For sets S and T, we denote the difference set $\{\, s \in S \mid s \notin T \,\}$ by $S \setminus T$. For a sequence $X = \langle x_1, \ldots, x_n \rangle$ $(n \in \mathbf{N})$, we denote the length n of X by $|X|$, the i-th element x_i of X by $X[i]$, and the consecutive subsequence $\langle x_i, \ldots, x_j \rangle$ of X by $X[i, j]$, for every $1 \leq i \leq j \leq n$, where we define $X[i, j] = \langle\rangle$ when $i > j$.

2.1 Input Event Sequence

Let $\Sigma = \{1, \ldots, m\}$ $(m \geq 1)$ be a finite alphabet with the total order $\leq$ over $\mathbf{N}$. Each element $e \in \Sigma$ is called an *event*[1]. An *input event sequence* (an *input sequence*, for short) $\mathcal{S}$ on Σ is a finite sequence $\langle S_1, \ldots, S_\ell \rangle \in (2^\Sigma)^*$ of sets of events of length ℓ. For an input sequence $\mathcal{S}$, we define the *total size* $||\mathcal{S}||$ of $\mathcal{S}$ by $\sum_{i=1}^{\ell} |S_i|$. Clearly, $||\mathcal{S}|| = O(|\Sigma|\ell)$. Without loss of generality, we can assume that every event in Σ appears at least once in $\mathcal{S}$.

[1] Mannila *et al.* [6] originally referred to each element $e \in \Sigma$ itself as an *event type* and an occurrence of e as an *event*. However, we simply refer to both of these as *events*.

2.2 Partite Episodes

Mannila *et al.* [6] and Katoh *et al.* [3] formulated an episode as a partially ordered set and a labeled acyclic digraph, respectively. In this paper, for a sub-class of episodes called *partite episodes*, we define an episode as a sequence of sets of events for simpler representations.

Definition 1. A *partite episode* P over Σ is a sequence $\langle P_1, \ldots, P_k \rangle \in (2^{\Sigma})^*$ of sets of events ($k \geq 0$), where $P_i \subseteq \Sigma$ is called the i-th *part* for every $1 \leq i \leq k$. For a partite episode $P = \langle P_1, \ldots, P_k \rangle$, we define the *total size* $||P||$ of P by $\sum_{i=1}^{k} |P_i|$. We call P *proper* when $P_i \neq \emptyset$ for every $1 \leq i \leq k$.

Definition 2. *Let* $P = \langle \{a_{(1,1)}, \ldots, a_{(1,m_1)}\}, \ldots, \{a_{(k,1)}, \ldots, a_{(k,m_k)}\} \rangle$ *be a partite episode of length* $k \geq 0$*, and* $Q = \langle \{b_{(1,1)}, \ldots, b_{(1,n_1)}\}, \ldots, \{b_{(l,1)}, \ldots, a_{(l,n_l)}\} \rangle$ *a partite episode of length* $l \geq 0$*, where* $m_i \geq 0$ *for every* $1 \leq i \leq k$ *and* $n_i \geq 0$ *for every* $1 \leq i \leq l$*. A partite episode* P *is a* sub-episode *of* Q*, denoted by* $P \sqsubseteq Q$*, if and only if there exists some injective mapping* $h : A \to B$ *satisfying* (i) $A = \bigcup_{i=1}^{k} \bigcup_{j=1}^{m_i} \{(i,j)\}$, $B = \bigcup_{i=1}^{l} \bigcup_{j=1}^{n_i} \{(i,j)\}$, (ii) *for every* $x \in A$, $a_x = b_{h(x)}$ *holds, and* (iii) *for every* $(p_1, q_1), (p_2, q_2) \in A$*, and values* $(p'_1, q'_1) = h((p_1, q_1))$ *and* $(p'_2, q'_2) = h((p_2, q_2))$*, if* $p_1 < p_2$ *holds, then* $p'_1 < p'_2$ *holds.*

Let $P = \langle P_1, \ldots, P_k \rangle$ and $Q = \langle Q_1, \ldots, Q_l \rangle$ be partite episodes of length $k \geq 0$ and $l \geq 0$, respectively. We denote the partite episode $\langle P_1, \ldots, P_k, Q_1, \ldots, Q_l \rangle$ by $P \mapsto Q$, the partite episode $\langle P_1 \cup Q_1, \ldots, P_{\max(k,l)} \cup Q_{\max(k,l)} \rangle$ by $P \circ Q$, and the partite episode $\langle P_1 \setminus Q_1, \ldots, P_k \setminus Q_k \rangle$ by $P \setminus Q$, where we assume that $P_i = \emptyset$ for any $i > k$ and $Q_j = \emptyset$ for any $j > l$. In Fig. 1, we show examples of an input event sequence S and partite episodes P^i ($1 \leq i \leq 7$).

2.3 Occurrences

Next, we introduce occurrences of episodes in an input sequence. An *interval* in an input sequence $\mathcal{S}$ is a pair of integer $(s,t) \in \mathbf{N}^2$ satisfying $s \leq t$. Let P be a partite episode, and $x = (s,t)$ be an interval in an input sequence $\mathcal{S} = \langle S_1, \ldots, S_\ell \rangle$ ($\ell \geq 0$). A partite episode P *occurs in* an interval x, if and only if $P \sqsubseteq \mathcal{S}[s, t-1]$, where we define $\mathcal{S}[i]$ for any index such that $i < 1$ and $i > |\mathcal{S}|$ as $\mathcal{S}[i] = \emptyset$.

For a partite episode P that occurs in an interval $x = (s,t)$ in an input sequence $\mathcal{S}$, the interval x is a *minimum occurrence* if and only if P does not occur in both intervals $(s+1,t)$ and $(s,t-1)$. Moreover, a *minimum occurrence set* of P on $\mathcal{S}$ is a set $\{\, (s,t) \mid P \sqsubseteq S[s,t-1], P \not\sqsubseteq S[s,t-2], P \not\sqsubseteq S[s+1,t-1] \,\} \subseteq \mathbf{N}^2$ of all minimum occurrences of P on $\mathcal{S}$. For a minimum occurrence set X of P, a *minimum occurrence list* (*mo-list*, for short) of P, denoted by $mo(P)$, is a sequence $\langle (s_1,t_1), \ldots, (s_n,t_n) \rangle \in (\mathbf{N}^2)^*$ of minimum occurrences such that $n = |X|$, $s_i < s_{i+1}$ for every $1 \leq i < n$.

A *window width* is a fixed positive integer $w \geq 1$. For an input sequence $\mathcal{S}$ and any $1 - w < i < |\mathcal{S}|$, the interval $(i, i+w)$ is a *window* of width w on $\mathcal{S}$. Then the frequency $freq_{\mathcal{S},w}(P)$ of a partite episode P is defined by the number of

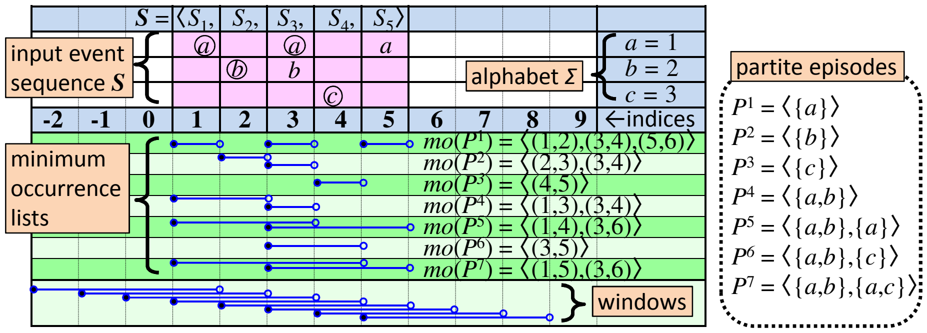

Fig. 1. An input event sequence $\mathcal{S}$ of length 5 (left top), partite episodes P^i $(1 \leq i \leq 7)$ (right), their minimum occurrence lists (left middle), and windows of width 4 on $\mathcal{S}$ (left bottom). In the input sequence $\mathcal{S}$, we indicate an occurrence of P^7 in the fourth window $(1, 5)$ in circles. See Example 1 for details.

windows (of width w on $\mathcal{S}$) in which P occurs. A *minimum frequency threshold* is any positive integer $\sigma \geq 1$. An episode P is *σ-frequent in $\mathcal{S}$* if $freq_{\mathcal{S},w}(P) \geq \sigma$. For a minimum frequency threshold σ and a window width w, the *minimum support threshold* $\acute{\sigma}$ is the relative value $\acute{\sigma} = \sigma/(|\mathcal{S}| + w - 1)$. By the definition of the frequency, we can show the next lemma.

Lemma 1. *Let P and Q be partite episodes, $\mathcal{S}$ an input sequence, and w a window width. If $P \sqsubseteq Q$ holds, then $freq_{\mathcal{S},w}(P) \geq freq_{\mathcal{S},w}(Q)$ holds.*

Our definition of frequency is identical to the one given by Mannila [6]. With this definition, an occurrence of episode could be counted more than once, and more than two occurrences could be counted only once. Lemma 1 holds even with another definition that does not have the problems above, although we do not explain why because of the limited space.

Example 1. In Fig. 1, we show an input sequence $\mathcal{S} = \langle\{a\}, \{b\}, \{a, b\}, \{c\}, \{a\}\rangle$ of length $\ell = 5$ over an alphabet $\Sigma = \{a, b, c\}$ of events, partite episodes $P^1 = \langle\{a\}\rangle$, $P^2 = \langle\{b\}\rangle$, $P^3 = \langle\{c\}\rangle$, $P^4 = P^1 \circ P^2 = \langle\{a, b\}\rangle$, $P^5 = P^4 \mapsto P^1 = \langle\{a, b\}, \{a\}\rangle$, $P^6 = P^4 \mapsto P^3 = \langle\{a, b\}, \{c\}\rangle$, and $P^7 = P^5 \circ P^6 = \langle\{a, b\}, \{a, c\}\rangle$, and their minimum occurrence lists $mo(P^1) = \langle(1, 2), (3, 4), (5, 6)\rangle$, $mo(P^2) = \langle(2, 3), (3, 4)\rangle$, $mo(P^3) = \langle(4, 5)\rangle$, $mo(P^4) = \langle(1, 3), (3, 4)\rangle$, $mo(P^5) = \langle(1, 4), (3, 6)\rangle$, $mo(P^6) = \langle(3, 5)\rangle$, and $mo(P^7) = \langle(1, 5), (3, 6)\rangle$, respectively, where $a = 1$, $b = 2$, and $c = 3$ are events. The input sequence $\mathcal{S}$ has eight windows with width 4 from $(-2, 2)$ to $(5, 9)$. Among these, the partite episode P^5 occurs in $(0, 4)$, $(1, 5)$, $(2, 6)$, and $(3, 7)$. Therefore, the frequency of P^5 is 4. Furthermore, the partite episode P^7 occurs in $(1, 5)$, $(2, 6)$, and $(3, 7)$. Therefore, the frequency of P^7 is 3. We see that $P^5 \sqsubseteq P^7$ and $freq_{\mathcal{S},4}(P^5) \geq freq_{\mathcal{S},4}(P^7)$.

3 Partwise Constraints

In this section, we introduce *partwise constraints* for partite episodes which consist of the *shape and pattern constraints*. A *shape constraint* for a partite episode P is a sequence $C = \langle c_1, \ldots, c_k \rangle \in \mathbf{N}^*$ $(k \geq 0)$ of natural numbers. A *pattern constraint* for a partite episode P is a sequence $D = \langle D_1, \ldots, D_k \rangle \in (2^{\Sigma})^*$ $(k \geq 0)$ of sets of events. We consider a pattern constraint as a partite episode.

Definition 3. A partite episode P *satisfies* the shape constraint C if $|P| \leq |C|$ and $|P[i]| \leq C[i]$ for every $1 \leq i \leq |C|$. A partite episode P *satisfies* the pattern constraint D if $|P| \geq |D|$ and $P[i] \supseteq D[i]$ for every $1 \leq i \leq |D|$.

By this definition, we see that a shape constraint is *anti-monotone* [10] and a pattern constraint is *monotone* [10].

Lemma 2. *Let P and Q be partite episodes such that $P \sqsubseteq Q$. For a shape constraint C and a pattern constraint D,* (i) *if Q is satisfying C then P is so, and* (ii) *if P is satisfying D then Q is so.*

We denote the classes of partite episodes (over Σ) by $\mathcal{PE}$. Moreover, we denote the classes of partite episodes satisfying a pattern constraint D by $\mathcal{PE}(D)$.

Definition 4. Partite Episode Mining with a Partwise Constraint: Given an input sequence $\mathcal{S} \in (2^{\Sigma})^*$, a window width $w \geq 1$, a minimum frequency threshold $\sigma \geq 1$, a shape constraint C, and a pattern constraint D, the task is to find all of the σ-frequent partite episodes satisfying C and D that occur in $\mathcal{S}$ with a window width w without duplication.

Our goal is to design an algorithm for the frequent partite episode mining problem with a partwise constraint, which we will show in the next section, in the framework of enumeration algorithms [2]. Let N be the total input size and M the number of solutions. An enumeration algorithm $\mathcal{A}$ is of *output-polynomial time*, if $\mathcal{A}$ finds all solutions in total polynomial time both in N and M. Moreover, $\mathcal{A}$ is of *polynomial delay*, if the *delay*, which is the maximum computation time between two consecutive outputs, is bounded by a polynomial in N alone.

4 Algorithm

In this section, we present an output-polynomial time and a polynomial-space algorithm PartiteCD for extracting all the frequent partite episodes satisfying partwise constraints in an input event sequence. Throughout this section, let $\mathcal{S} = \langle S_1, \ldots, S_\ell \rangle \in (2^{\Sigma})^*$ be an input event sequence over an alphabet Σ, $w \geq 1$ a window width, and $\sigma \geq 1$ the minimum frequency threshold. Furthermore, let M be the number of all solutions of our algorithm, and m the maximum size of output episode P, that is, $m = \|P\| + |P|$. Then we define the length of the input sequence $\ell = |\mathcal{S}|$, the total size of the input sequence $N = ||\mathcal{S}|| + \ell$, the total size of the constraints $c = |C| + \|D\| + |D|$, and the alphabet size $s = |\Sigma|$ for analysis of time and space complexity of our algorithm, where we assume $O(s) = O(N)$ and $O(m) = O(N)$.

4.1 Family Tree

The main idea of our algorithm is to enumerate all of the frequent partite episodes satisfying partwise constraints by searching the whole search space from general to specific by using depth-first search. First, we define the search space. For a partite episode P such that $|P| \geq 1$, the *tail pattern* $tail(P)$ of P is defined by the partite episode Q such that $|Q| = \max\{\, 1 \leq i \leq |P| \mid P[i] \neq \emptyset \,\}$, $Q[i] = \emptyset$ for every $1 \leq i < |Q|$, and $Q[i] = \{\max P[i]\}$ for $i = |Q|$, where $\max P[i]$ is the maximum integer in the i-th part of P. Then, for a pattern constraint D, we introduce the parent-child relationship between partite episodes satisfying D.

Definition 5. The partite episode $\bot = D$ is the *root*. The *parent* of the partite episode $P = \langle P_1, \ldots, P_k \rangle$ is defined by:

$$parent_D(P) = \begin{cases} \langle P_1, \ldots, P_{k-1} \rangle, & \text{if } |P_k| \leq 1 \text{ and } |P| > |D|, \\ P \setminus tail(P \setminus D), & \text{otherwise.} \end{cases}$$

We define the set of all children of P by $children_D(P) = \{\, Q \mid parent_D(Q) = P \,\}$. Then we define the *family tree* for $\mathcal{PE}(D)$ by a rooted digraph $\mathcal{T}(\mathcal{PE}(D)) = (V, E, \bot)$ with the root $\bot$, where the root $\bot = D$, the vertex set $V = \mathcal{PE}(D)$, and the edge set $E = \{\, (P, Q) \mid P = parent_D(Q), Q \neq D \,\}$.

Lemma 3. *The family tree $\mathcal{T}(\mathcal{PE}(D)) = (V, E, D)$ for $\mathcal{PE}(D)$ is a rooted tree with the root D.*

For any episode Q on $\mathcal{T}(\mathcal{PE}(D))$, the parent $parent_D(Q)$ is a subepisode of Q. Therefore, we can show the next lemma by Lemma 1 and Lemma 2.

Lemma 4. *Let C be a shape constraint, D a pattern constraint, and P and Q partite episodes such that $P = parent_D(Q)$. If Q is frequent then so is P. If Q is satisfying C then so is P.*

By Lemma 2 and Lemma 3, we know that a family tree $\mathcal{T}(\mathcal{PE}(D))$ contains only episodes satisfying a pattern constraint D. Thus, we can make a search tree containing only episodes that are frequent and satisfy the shape and pattern constraints by pruning episodes that do not satisfy the conditions.

Example 2. We describe the part of family trees $\mathcal{T}(\mathcal{PE}(\langle\rangle))$ and $\mathcal{T}(\mathcal{PE}(\langle\{\}, \{b\}\rangle))$ that forms the spanning trees for all partite episodes of $\mathcal{PE}(\langle\rangle)$ and $\mathcal{PE}(\langle\{\}, \{b\}\rangle)$ on an alphabet $\Sigma = \{a, b\}$ in Fig. 2. For a pattern constraint $D = \langle\{\}, \{b\}\rangle$, the parent of $P^1 = \langle\{a, b\}, \{a, b\}\rangle$ is $P^1 \setminus tail(P^1 \setminus D) = P^1 \setminus tail(\langle\{a, b\}, \{a\}\rangle) = P^1 \setminus \langle\{\}, \{a\}\rangle = \langle\{a, b\}, \{b\}\rangle = P^2$.

4.2 Pattern Expansion and Incremental Computation

Secondly, we discuss how to enumerate all children of a parent. For a pattern constraint D, a partite episode Q, and its parent P, we define the *index of the expanded part* of Q by $iex(Q) = |Q|$ if $|Q| > |P|$, and $iex(Q) = |tail(Q \setminus P)|$ otherwise. Additionally, we define $iex(Q) = 0$ for the root $Q = D$.

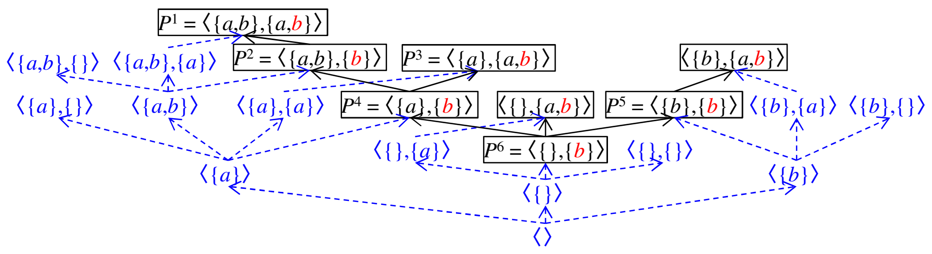
$P^1 = \langle\{a,b\},\{a,b\}\rangle$
$\langle\{a,b\},\{\}\rangle$
$\langle\{a,b\},\{a\}\rangle$
$P^2 = \langle\{a,b\},\{b\}\rangle$
$P^3 = \langle\{a\},\{a,b\}\rangle$
$\langle\{b\},\{a,b\}\rangle$
$\langle\{a\},\{\}\rangle$
$\langle\{a,b\}\rangle$
$\langle\{a\},\{a\}\rangle$
$P^4 = \langle\{a\},\{b\}\rangle$
$\langle\{\},\{a,b\}\rangle$
$P^5 = \langle\{b\},\{b\}\rangle$
$\langle\{b\},\{a\}\rangle$
$\langle\{b\},\{\}\rangle$
$\langle\{\},\{a\}\rangle$
$P^6 = \langle\{\},\{b\}\rangle$
$\langle\{\},\{\}\rangle$
$\langle\{a\}\rangle$
$\langle\{\}\rangle$
$\langle\{b\}\rangle$
$\langle\rangle$

Fig. 2. The parent-child relationships on an alphabet $\Sigma = \{a, b\}$ for pattern constraints $\langle\rangle$ (dashed arrows) and $\langle\{\}, \{b\}\rangle$ (solid arrows), where $a = 1$ and $b = 2$ are events.

Let $h = iex(P)$ be the index of the expanded part of a parent episode P. We define *type-i children* of P by $children_D(P, i) = \{\, Q \in children_D(P) \mid iex(Q) = i, \|Q\| > \|P\| \,\}$ for an index $i \geq h$. By Definition 5, we can make any type-i child Q of P by adding an event $a \in \Sigma \setminus (P[i] \cup D[i])$ at $P[i]$. Moreover, by Definition 5, we see $children_D(P) = (\bigcup_{i=m}^{n} children_D(P, i)) \cup \{P \mapsto \langle\emptyset\rangle\}$, where $m = \max(h, 1)$, and $n = \max(h, |D|) + 1$.

Furthermore, for partite episodes P, Q, and S such that $P = parent_D(Q)$, $S = parent_D(P)$, $iex(Q) \geq 1$, and $|(Q \setminus D)[iex(Q)]| \geq 2$, we define the *uncle* R of Q by $uncle_D(Q) = S \circ tail(Q \setminus D)$.

Example 3. In Fig. 2, for a pattern constraint $D = \langle\{\}, \{b\}\rangle$, an episode $P^4 = \langle\{a\}, \{b\}\rangle$ is the parent of an episode $P^2 = \langle\{a, b\}, \{b\}\rangle$. The index $iex(P^2)$ of the expanded part of P^2 is $|tail(P^2 \setminus P^4)| = |tail(\langle\{b\}, \{\}\rangle)| = |\langle\{b\}\rangle| = 1$. Therefore, P^2 is a type-1 child of P^4. On the other hand, P^3 is a type-2 child of P^4 because $iex(P^3)$ is $|tail(\langle\{\}, \{b\}\rangle)| = 2$. Since the parent of P^4 is P^6, the uncle of P^2 is $P^6 \circ tail(P^2 \setminus D) = P^6 \circ \langle\{a\}\rangle = \langle\{a\}, \{b\}\rangle = P^5$.

To compute a type-i child Q of P and its mo-list incrementally from the parent P, we make Q as follows. Let $h = iex(P)$ be the index of the expanded part of P; (i) we make Q by $P \circ R$, and the subepisode $Q_{sub} = Q[1, i]$ of Q by $P_{sub} \circ R_{sub}$ when $i = h$, where $R = uncle_D(Q)$, $P_{sub} = P[1, i]$, and $R_{sub} = R[1, i]$; (ii) we make Q by $P_{sub} \mapsto D_H \mapsto A \mapsto D_T$ and the subepisode $Q_{sub} = Q[1, i]$ of Q by $P_{sub} \mapsto D_H \mapsto A$ when $i > h$, where $P_{sub} = P[1, h]$, $D_H = D[h + 1, i - 1]$, $A = D[i] \circ \langle\{a\}\rangle$, and $D_T = D[i + 1, |D|]$. Since the length of mo-list I of an episode P is less than the length $|\mathcal{S}|$ of the input sequence $\mathcal{S}$, we can incrementally compute the mo-list in $O(\|\mathcal{S}\|) = O(N)$ time [3, 4, 8].

4.3 Depth-first Enumeration

In Fig. 3, we describe the algorithm PartiteCD, and its subprocedure RecCD for extracting frequent partite episodes satisfying a partwise constraint. For a pattern constraint D, the algorithm is a backtracking algorithm that traverses the spanning tree $\mathcal{T}(\mathcal{PE}(D))$ based on a depth-first search starting from the root $P = D$ using the parent-child relationships over $\mathcal{PE}(D)$.

```
algorithm PartiteCD($\mathcal{S}, \Sigma, w, \sigma, C, D$)
input: input sequence $\mathcal{S} \in (2^{\Sigma})^*$, alphabet of events $\Sigma$, window width $w > 0$, minimum
frequency $1 \leq \sigma$, shape and pattern constraints $C$ and $D$ ,respectively;
output: frequent partite episodes satisfying the shape and pattern constraints;
01 compute the mo-list $mo(a)$ for each event $a$ in $\Sigma$
       by scanning imput sequence $\mathcal{S}$ at once and store to $\Sigma_{\mathcal{S}}$;
02 make the root episode $P = D$ and compute its mo-list $mo(P)$ from $\Sigma_{\mathcal{S}}$;
03 if ($P$ is frequent and satisfying $C$) RecCD($P, 1, \emptyset, \Sigma_{\mathcal{S}}, w, \sigma, C, D$);

procedure RecCD($P, i, U, \Sigma_{\mathcal{S}}, w, \sigma, C, D$)
output: all frequent partite episodes that are descendants of $P$;
04 output $P$;
05 while ($i \leq \max(h, |D|) + 1$) do // where $h = iex(P)$.
06    $V := \emptyset$;
07    foreach ($a \in \Sigma \setminus (P[i] \cup D[i])$) do
08       make a type-i child $Q$ of $P$ by adding the event $a$ at $P[i]$;
09       compute the mo-lists of $Q$ and the subepisode $Q_{sub} = Q[1, i]$ by using $U$;
10       if ($Q$ is frequent and satisfying $C$) store $(Q, mo(Q), mo(Q_{sub}))$ to $V$;
11    end foreach
12    foreach (a child $Q$ of $P$ stored in $V$) RecCD($Q, i, V, \Sigma_{\mathcal{S}}, w, \sigma, C, D$);
13    $i := i + 1$;
14 end while
15 if ($P \mapsto \langle\emptyset\rangle$ is frequent and satisfying $C$) RecCD($P \mapsto \langle\emptyset\rangle, i, \emptyset, \Sigma_{\mathcal{S}}, w, \sigma, C, D$);
```

Fig. 3. The main algorithm PartiteCD and a recursive subprocedure RecCD for mining frequent partite episodes satisfying shape and pattern constraints.

First, for an alphabet Σ, PartiteCD computes mo-lists $mo(\langle\{a\}\rangle)$ of partite episodes of size 1 for every event $a \in \Sigma$. Then PartiteCD makes the root episode $P = D$ and calls the recursive subprocedure RecCD. Finally, for an episode P the recursive subprocedure RecCD enumerates all frequent episodes that are descendants of P and satisfy the shape constraint C. Algorithm PartiteCD finds all of the frequent partite episodes satisfying the partwise constraints occurring in an input event sequence until all recursive calls are finished.

Theorem 1. *Algorithm* PartiteCD *runs in $O(Nsc)$ amortized time per output and in $O(Nsm)$ space.*

Finally, we describe the possible improvement for PartiteCD. By using the alternating output technique [14], that is, we output episodes after Line 15 (the end of RecCD) instead of Line 4 if and only if the depth of recursions is even, our algorithm runs in $O(Nsc)$ delay.

5 Experimental Results

In this section, we show that the algorithm described in Sect. 4 has a significantly better performance than a straightforward algorithm which consists of an algorithm for non-constraint episodes and post-processing steps for partwise constraints by experiments for an artificial data set.

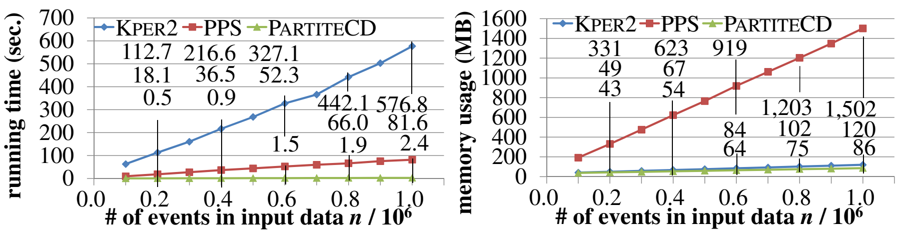

Fig. 4. Running time (left) and memory usage (right) for the number of events in input data.

5.1 Data and Method

An artificial data set consisted of the randomly generated event sequence $\mathcal{S} = \langle S_1, \ldots, S_\ell \rangle$ ($\ell \geq 1$) over an alphabet $\Sigma = \{1, \ldots, s\}$ ($s \geq 1$) as described below. Let $1 \leq i \leq \ell$ be any index. For every event $a \in \Sigma$, we add a into S_i independently with a probability of $p = 0.1/a$. By repeating this process for every S_i, we made a skewed data set that emulates real-world data sets having long tails such as point-of-sale data sets.

We implemented the following algorithms to compare their performance. Kper2 is a straightforward algorithm which consists of our previous algorithm [3] for non-constraint episodes and post-processing steps for partwise constraints. PPS is an algorithm which consists of Ma's algorithm [5] handling prefix anti-monotone constraints and post-processing steps for partwise constraints which are not prefix anti-monotonic. PPS efficiently handles shape constraints and a part of pattern constraints which have prefix anti-monotonic property, that is, for a partite episode P, a pattern constraint D, and an index $1 \leq i \leq |P|$, if $P[1, i]$ does not satisfy $D[1, i]$, then PPS does not generate P. PartiteCD is our algorithm that handles partwise constraints presented in Sect. 4.

All the experiments were run on a Xeon X5690 3.47GHz machine with 180 GB of RAM running Linux (CentOS 6.3). If not explicitly stated otherwise, we assume that the number of event in the input sequence S is $n = \|\mathcal{S}\| = 1,000,000$, the alphabet size is $s = |\Sigma| = 1,000$, the window width is $w = 20$, the minimum support threshold is $\acute{\sigma} = 0.1\%$, the shape constraint $C = \langle +\infty, 1, +\infty \rangle$, and the pattern constraint $D = \langle \{10\}, \emptyset, \{10\} \rangle$.

5.2 Experiments

Figure 4 shows the running time (left), and the amount of memory usage (right) of the algorithms Kper, PPS, and PartiteCD for the number of events n in input data. With respect to the total running time, we see that PartiteCD is 230 times as fast as Kper2 and 35 times as fast as PPS in this case. This difference comes from the number of episode generated by each algorithm before post-processing steps.

For the amount of memory usage, we see that PPS uses more memory than both Kpar2 and PartiteCD. The reason is that PPS is based on breadth-first

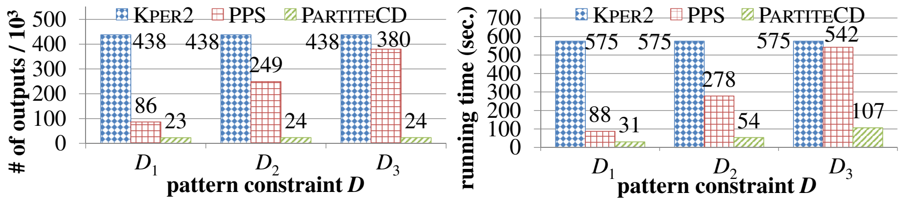

Fig. 5. The number of outputs before post-prossesing steps (left) and running time (rigth) for pattern constraints D_i $(1 \leq i \leq 3)$, where $D_1 = \langle\{10\}, \emptyset, \emptyset\rangle$, $D_2 = \langle\emptyset, \{10\}, \emptyset\rangle$, and $D_3 = \langle\emptyset, \emptyset, \{10\}\rangle$.

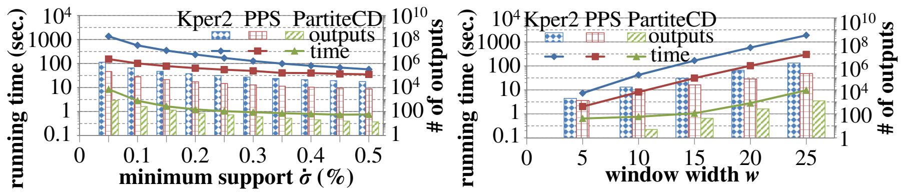

Fig. 6. Running time and number of outputs against minimum supports (left) and window width (right).

search in a search space [5], whereas both KPAR2 and PARTITECD are based on depth-first search. We also see that, both the running time and the memory usage of these algorithms seem to be almost linear to the input size and must therefore scale well on large datasets.

Figure 5 shows the number of outputs before the post-processing steps (left) and the total running time (right) for pattern constraints D_i $(1 \leq i \leq 3)$. In this experiment, we used a shape constraint $C = \langle+\infty, +\infty, +\infty\rangle$. We see that the number of outputs before post-processing steps of PPS is large and the running time is long if a part $\{10\}$ of the pattern constraint D appears at the tail side of D. The reason is that since PPS reduces the number of outputs by pruning based on prefix anti-monotonic property, PPS generates a large number of episodes before pruning by the constraint that appears at the tail side. On the other hand, the number of outputs of PARTITECD is almost constant for every constraint D, because PARTITECD generates only episodes that satisfy the given constraints.

Figure 6 shows the total running time and the number of outputs before the post-processing steps against minimum support $\dot{\sigma}$ (left) and window width w (right). We see that PARTITECD outperforms other algorithms both on the number of outputs and the running time for every parameter set.

Overall, we conclude that the partwise constraint reduces the number of outputs and the proposed algorithm handles the constraint much more efficiently than the straightforward algorithm using post-processing steps. In other words, by using our algorithm with our constraint, we can extract partite episodes on which we focused with lower frequency thresholds for larger input data in the same running time.

6 Conclusion

This paper studied the problem of frequent partite episode mining with partwise constraints. We presented an algorithm that finds all frequent partite episodes satisfying a partwise constraint in an input sequence. Then, we showed that our algorithm runs in the output polynomial time and polynomial space. We reported the experimental results that compare the performance of our algorithm and other straightforward algorithms. Both the theoretical and experimental results showed that our algorithm efficiently extracts episodes on which we focused. Thus, we conclude that our study improved the efficiency of data mining.

A possible future path of study will be the extension of PartiteCD for the class of proper partite episodes to reduce redundant outputs by traversing a search tree containing only proper partite episodes satisfying a pattern constraint. Our future work will also include the application of our algorithm to real-world data sets.

References

1. Arimura, H., Uno, T.: A polynomial space and polynomial delay algorithm for enumeration of maximal motifs in a sequence. In: ISAAC, LNCS 3827. pp. 724–737. Springer-Verlag (2005)
2. Avis, D., Fukuda, K.: Reverse search for enumeration. Discrete Applied Mathematics 65, 21–46 (1996)
3. Katoh, T., Arimura, H., Hirata, K.: Mining frequent k-partite episodes from event sequences. In: LLLL, LNAI 6284. pp. 331–344. Springer-Verlag (2010)
4. Katoh, T., Hirata, K.: A simple characterization on serially constructible episodes. In: PAKDD, LNAI 5012. pp. 600–607. Springer-Verlag (2008)
5. Ma, X., Pang, H., Tan, K.L.: Finding constrained frequent episodes using minimal occurrences. In: ICDM. pp. 471–474 (2004)
6. Mannila, H., Toivonen, H., Verkamo, A.I.: Discovery of frequent episodes in event sequences. Data Mining and Knowledge Discovery 1(3), 259–289 (1997)
7. Méger, N., Rigotti, C.: Constraint-based mining of episode rules and optimal window sizes. In: PKDD. pp. 313–324 (2004)
8. Ohtani, H., Kida, T., Uno, T., Arimura, H.: Efficient serial episode mining with minimal occurrences. In: ICUIMC. pp. 457–464 (2009)
9. Pei, J., Han, J., Mortazavi-Asi, B., Wang, J.: Mining sequential patterns by pattern-growth: The PrefixSpan approach. IEEE Transactions on Knowledge and Data Engineering 16(11), 1–17 (2004)
10. Pei, J., Han, J.: Can we push more constraints into frequent pattern mining? In: KDD. pp. 350–354 (2000)
11. Pei, J., Han, J., Wang, W.: Mining sequential patterns with constraints in large databases. In: CIKM. pp. 18–25. ACM (2002)
12. Seipel, D., Köhler, S., Neubeck, P., Atzmueller, M.: Mining complex event patterns in computer networks. In: NFMCP, LNAI 7765. Springer Verlag (2012)
13. Tatti, N., Cule, B.: Mining closed strict episodes. Data Mining and Knowledge Discovery 25(1), 34–66 (2012)
14. Uno, T.: Two general methods to reduce delay and change of enumeration algorithms. Tech. rep., National Institute of Informatics (2003)
15. Zhou, W., Liu, H., Cheng, H.: Mining closed episodes from event sequences efficiently. In: PAKDD. pp. 310–318 (2010)