Mining Frequent Diamond Episodes from Event Sequences*

Takashi Katoh[1], Kouichi Hirata[2], and Masateru Harao[2]

[1] Graduate School of Computer Science and Systems Engineering
[2] Department of Artificial Intelligence
Kyushu Institute of Technology
Kawazu 680-4, Iizuka 820-8502, Japan
{f673024t,hirata,harao}@ai.kyutech.ac.jp
Tel: +81-948-29-7622, Fax: +81-948-29-7601

Abstract. In this paper, we introduce a *diamond episode* of the form $s_1 \mapsto E \mapsto s_2$, where s_1 and s_2 are events and E is a set of events. The diamond episode $s_1 \mapsto E \mapsto s_2$ means that every event of E follows an event s_1 and is followed by an event s_2. Then, by formulating the *support* of diamond episodes, in this paper, we design the algorithm FreqDmd to extract all of the *frequent diamond episodes* from a given event sequence. Finally, by applying the algorithm FreqDmd to bacterial culture data, we extract diamond episodes representing *replacement of bacteria*.

keywords: data mining in time-related data, episode mining, data mining in medicine

1 Introduction

The *sequential pattern mining* [3, 8, 9, 12, 13] is one of the data mining methods from time-related data. The purpose of sequential pattern mining is to discover frequent *subsequences* as patterns in a sequential database. On the other hand, the *episode mining* [4, 7] introduced by Mannila *et al.* [7] is known as another approach to discover frequent patterns from time-related data. The purpose of episode mining is to discover *frequent episodes*, not subsequences, that are a collection of events occurring frequently together in event sequences.

In episode mining, the frequency is formulated as the number of occurrences of episodes in every *window* that is a subsequence of event sequences under a fixed time span called the *width* of windows. Then, Mannila *et al.* [7] have introduced a *parallel episode* as a set of events and a *serial episode* as a sequence of events. By combining the above episodes, they have extended the forms of episodes as *directed acyclic graphs* of events of which edges specify the temporal precedent-subsequent relationship.

On the other hand, in order to focus on the direct *causality* in time-related data, Katoh *et al.* [5, 6] have introduced a *sectorial episode* of the form $E \mapsto e$,

* This work is partially supported by Grand-in-Aid for Scientific Research 17200011 from the Ministry of Education, Culture, Sports, Science and Technology, Japan.

where E is a parallel episode and e is an event. A sectorial episode is a direct precedent-subsequent relationship between events. Then, they have designed an efficient algorithm Sect to extract all of the sectorial episodes that are frequent and accurate, and apply the algorithm to bacterial culture data in order to extract sectorial episodes representing changes for drug resistance. However, it remains open to formulate an episode that represents an indirect precedent-subsequent relationship beyond a sectorial episode, and then to design an algorithm to extract such episodes.

In order to solve this problem, in this paper, we introduce a *diamond episode*, which is an extension of a sectorial episode, of the form $s_1 \mapsto E \mapsto s_2$, where s_1 and s_2 are events and E is a parallel episode. We call s_1 a *source* and s_2 a *sink*. The diamond episode $s_1 \mapsto E \mapsto s_2$ means that every event of E follows a source s_1 and is followed by a sink s_2, so we can regard every event in E as a *causation* through the change from a source s_1 to a sink s_2.

Then, we formulate the *support* $supp(s_1 \mapsto E \mapsto s_2)$ of a diamond episode $s_1 \mapsto E \mapsto s_2$ as the ratio of the number of k-windows (i.e., a window with width k) in which $s_1 \mapsto E \mapsto s_2$ occurs for the number of all k-windows. For the *minimum support* σ such that $0 < \sigma < 1$, we say that $s_1 \mapsto E \mapsto s_2$ is *frequent* if $supp(s_1 \mapsto E \mapsto s_2) \geq \sigma$.

In this paper, we show that the diamond episode preserves *anti-monotonicity*, that is, for $E_1 \subseteq E_2$, if $s_1 \mapsto E_2 \mapsto s_2$ is frequent then so is $s_1 \mapsto E_1 \mapsto s_2$. Hence, under fixing a source s_1 and a sick s_2, we can deal with a diamond episode $s_1 \mapsto E \mapsto s_2$ as an itemset, by regarding a serial episode $s_1 \mapsto e \mapsto s_2$ with size 3 as an item for $e \in E$. Then, we design the algorithm FreqDmd to extract all of the *frequent diamond episodes* from a given event sequence by using the frequent itemset mining algorithm AprioriTid [1, 2].

Since our diamond episode is a combination of parallel and serial episodes, it is possible to extract all of the frequent diamond episodes by combining the algorithms designed in [7]. It is a main advantage for our algorithm FreqDmd to be efficient and appropriate for extracting frequent diamond episodes, because the algorithm FreqDmd scans a given event sequence just once and regards a serial episode as an itemset, in the extraction of frequent diamond episodes.

The algorithm FreqDmd contains the algorithm BitSerl, which is a main difference from the algorithm Sect [5]. The algorithm BitSerl constructs the set of bit vectors representing the occurrences of all serial episodes with size 3, by using the bit-wise conjunction and disjunction, and the shift operator.

The algorithm FreqDmd first constructs all of the serial episodes of the form $s_1 \mapsto e \mapsto s_2$ for an event e and their bit vectors by calling BitSerl. Then, it extracts the set of all frequent diamond episodes of the form $s_1 \mapsto E \mapsto s_2$ for a parallel episode E, by regarding a serial episode $s_1 \mapsto e \mapsto s_2$ as an item e and by calling the algorithm AprioriTid [1, 2].

Finally, we apply the algorithm FreqDmd to bacterial culture data. Note that, from the medical viewpoint, in order to extract frequent diamond episodes concerned with *replacement of bacteria*, it is necessary to extract them based on the same sample. Hence, in this paper, by fixing the sample, we connect data of

every patient with some span, and then extract diamond episodes representing the replacement of bacteria from them.

2 Diamond Episode

As similar as [7], we assume that an event has an associated time of occurrence as a natural number. Formally, let $\mathcal{E}$ be a set of *event types*. Then, a pair (e, t) is called an *event*, where $e \in \mathcal{E}$ and t is a natural number which is the (*occurrence*) *time* of the event. In the following, for a set $E \subseteq \mathcal{E}$ of event types, we denote $\{(e, t) \mid e \in \mathcal{E}\}$ by (E, t), and also call it by an *event* again. Furthermore, we denote a set $\{e_1, \ldots, e_m\} \subseteq \mathcal{E}$ of event types by a string $e_1 \cdots e_m$.

An *event sequence* $\mathcal{S}$ on $\mathcal{E}$ is a triple (S, T_s, T_e), where

$$S = \langle (E_1, t_1), \ldots, (E_n, t_n) \rangle$$

is an ordered sequence of events satisfying the following conditions.

1. $E_i \subseteq \mathcal{E}$ $(1 \leq i \leq n)$,
2. $t_i < t_{i+1}$ $(1 \leq i \leq n-1)$, and
3. $T_s \leq t_i < T_e$ $(1 \leq i \leq n)$.

In particular, T_s and T_e are called the *starting* time and the *ending* time of $\mathcal{S}$. We denote $T_e - T_s$ by $l_{\mathcal{S}}$.

A *window* in an event sequence $\mathcal{S} = (S, T_s, T_e)$ is an event sequence $W = (w, t_s, t_e)$ such that $t_s < T_e$, $t_e > T_s$ and w consists of all of the events (e, t) in S where $t_s \leq t < t_e$. The time span $t_e - t_s$ is called the *width* of the window. We call a window with width k in $\mathcal{S}$ a *k-window*, and denote the k-window $(w, t, t + k)$ of $\mathcal{S}$ starting from t by $w(\mathcal{S}, t, k)$.

Note that we can regard a set $E = e_1 \cdots e_m$ of event types as a *parallel episode* and a sequence $e_1 \mapsto \cdots \mapsto e_m$ of event types as a *serial episode* [7]. In this paper, we newly introduce the following *diamond episode*, as an extension of a sectorial episode [5, 6].

Definition 1. Let s_1 and s_2 be event types and $E \subseteq \mathcal{E}$ be a parallel episode. Then, a *diamond episode* is of the following form.

$$s_1 \mapsto E \mapsto s_2.$$

We call s_1 a *source* and s_2 a *sink*.

We call a parallel, a serial and a diamond episodes *episodes* simply.

Let $\mathcal{S}$ be an event sequence $\mathcal{S} = (S, T_s, T_e)$ and e an event type. Then, we say that e *occurs* in $\mathcal{S}$ if there exists an event $(E, t) \in S$ such that $e \in E$. We denote $\{t \mid (E, t) \in S \wedge e \in E\}$ by $T(e, \mathcal{S})$. Also we denote $st(e, \mathcal{S}) = \min\{t \mid t \in T(e, \mathcal{S})\}$ and $et(e, \mathcal{S}) = \max\{t \mid t \in T(e, \mathcal{S})\}$.

Definition 2. We say that a parallel episode E *occurs* in $\mathcal{S}$ if every $e \in E$ occurs in $\mathcal{S}$. Also we say that a serial episode $e_1 \mapsto \cdots \mapsto e_m$ *occurs* in $\mathcal{S}$ if every e_i occurs in $\mathcal{S}$ $(1 \leq i \leq m)$ and $st(e_i, \mathcal{S}) < et(e_{i+1}, \mathcal{S})$ $(1 \leq i \leq m-1)$.

We say that a diamond episode $s_1 \mapsto E \mapsto s_2$ *occurs* in $\mathcal{S}$ if s and t occur in $\mathcal{S}$, and, for every $e \in E$, e occurs in $\mathcal{S}$, $st(s_1, \mathcal{S}) < et(e, \mathcal{S})$ and $st(e, \mathcal{S}) < et(s_2, \mathcal{S})$.

Let $\mathcal{S}$ be an event sequence and k a natural number. Then, we denote the set of all k-windows by $W(\mathcal{S}, k)$. Also, for an episode X, we denote the set of all k-windows such that X occurs in $\mathcal{S}$ by $W(X, \mathcal{S}, k)$.

Note that we can number all k-windows in $W(\mathcal{S}, k)$ from $T_s - k$ to T_e. We call such a number i $(T_s - k < i < T_e)$ the *label* of the i-th k-window. For an event sequence $\mathcal{S}$ and an episode X, we identify $W(X, \mathcal{S}, k)$ with the set of all labels of k-windows in which X occurs in $\mathcal{S}$.

Example 1. Let $\mathcal{E} = \{a, b, c\}$. Then, Figure 1 describes an event sequence $\mathcal{S} = (S, 4, 10)$ on $\mathcal{E}$ where:

$$S = \langle (abc, 4), (ab, 5), (a, 6), (ab, 7), (abc, 8), (ab, 9) \rangle.$$

Also an event sequence $w = (\langle (ab, 5), (a, 6), (ab, 7), (abc, 8), (ab, 9) \rangle, 5, 9)$ is a 5-window of $\mathcal{S}$ starting from 5, that is, $w = w(\mathcal{S}, 5, 4)$.

For the above event sequence $\mathcal{S}$, it holds that $T(c, \mathcal{S}) = \{4, 8\}$, $st(c, \mathcal{S}) = 4$ and $et(c, \mathcal{S}) = 8$. Also there exist 10 5-windows, of which starting time is from 1 to 9. Furthermore, for the above window w, diamond episodes $a \mapsto ab \mapsto c$ and $b \mapsto abc \mapsto b$ occur in w, for example.

4	5	6	7	8	9
a	a	a	a	a	a
b	b		b	b	b
c				c	

Fig. 1. An event sequence S in Example 1.

Let $\mathcal{S}$ be an event sequence, X an episode and k a natural number. Then, the *frequency* $freq_{\mathcal{S},k}(X)$ and the *support* $supp_{\mathcal{S},k}(X)$ of X in $\mathcal{S}$ w.r.t. k are defined as follows.

$$freq_{\mathcal{S},k}(X) = |W(X, \mathcal{S}, k)|,\ supp_{\mathcal{S},k}(X) = \frac{freq_{\mathcal{S},k}(X)}{|W(\mathcal{S}, k)|}.$$

Definition 3. Let σ be the *minimum support* such that $0 < \sigma < 1$. Then, we say that an episode X is *frequent* if $supp(X) \geq \sigma$.

Lemma 1 (Anti-monotonicity for diamond episodes). *Let E_1 and E_2 be parallel episodes such that $E_1 \subseteq E_2$, If $s_1 \mapsto E_2 \mapsto s_2$ is frequent, then so is $s_1 \mapsto E_1 \mapsto s_2$.*

Proof. It is sufficient to show that $W(s_1 \mapsto E_2 \mapsto s_2, \mathcal{S}, k) \subseteq W(s_1 \mapsto E_1 \mapsto s_2, \mathcal{S}, k)$. Suppose that $l \in W(s_1 \mapsto E_2 \mapsto s_2, \mathcal{S}, k)$ and let W_l be the l-th k-window in $\mathcal{S}$. Then, it holds that $W_l = w(W(s_1 \mapsto E_2 \mapsto s_2, \mathcal{S}, k), l, k)$. For every $e \in E_2$, it holds that $st(s_1, W_l) < et(e, W_l)$ and $st(e, W_l) < et(s_2, W_l)$. Since $E_1 \subseteq E_2$, it holds that $st(s_1, W_l) < et(e', W_l)$ and $st(e', W_l) < et(s_2, W_l)$ for every $e' \in E_1$, so $s_1 \mapsto E_1 \mapsto s_2$ occurs in W_l. Hence, it holds that $l \in W(s_1 \mapsto E_1 \mapsto s_2, \mathcal{S}, k)$. □

3 Algorithm to Extract Diamond Episode

In this section, we design the algorithm to extract all of the frequent diamond episodes from an event sequence $\mathcal{S}$, the minimum support σ and the width k of windows.

The algorithm BitSerl, described as Figure 2, computes a bit vector of the occurrences of all serial episodes with size 3, by scanning an event sequence. Here, for a bit vector $v \in \{0,1\}^*$, $|v|$ denotes the length of v, and v_i denotes the i-th bit of v for $1 \leq i \leq |v|$, that is, $v = v_1 \cdots v_{|v|}$. Then, $\mathit{sft}(v)$ is $v_2 \cdots v_{|v|}0$. For $v, w \in \{0,1\}^*$, $v \circ w$ denotes the concatenation of v and w. Furthermore, for $v, w \in \{0,1\}^*$ such that $|v| = |w|$, $v \wedge w$ and $v \vee w$ are bit-wise logical conjunction and bit-wise logical disjunction, respectively. In particular, $0(k)$ denotes a bit vector $\underbrace{0 \cdots 0}_{k}$.

---

procedure BitSerl$(S, T_s, T_e, k, \mathcal{E})$
/* $\langle S, T_s, T_e \rangle$: event sequence, k: the width of windows, $\mathcal{E}$: event types */
/* initialization, where $B[e] = B[e][T_s] \cdots B[e][T_e - 1]$ */
for $t = T_s$ **to** $T_e - 1$ **do**
  foreach $e \in \mathcal{E}$ **do** $B[e][t] \leftarrow 0$;
/* transforming an event sequence to a set of bit vectors */
for $t = T_s$ **to** $T_e - 1$ **do**
  foreach $e \in E$ such that $(E, t) \in S$ **do** $B[e][t] \leftarrow 1$;
/* constructing a bit vector of a serial episode $a \mapsto b \mapsto c$ */
$l \leftarrow T_e - T_s + k - 1$; $T \leftarrow \emptyset$;
foreach $(a, b, c) \in \mathcal{E} \times \mathcal{E} \times \mathcal{E}$ **do begin** /* $|V[\cdot]| = |W[\cdot]| = l$ */
  $V[a] \leftarrow 0(k-1) \circ B[a]$; $V[b] \leftarrow 0(k-1) \circ B[b]$; $V[c] \leftarrow 0(k-1) \circ B[c]$;
  $W[c] \leftarrow V[c]$; $W[b \mapsto c] \leftarrow 0(l)$; $W[a \mapsto b \mapsto c] \leftarrow 0(l)$;
  for $d = 1$ **to** $k - 1$ **do begin**
    $W[a \mapsto b \mapsto c] \leftarrow \mathit{sft}(W[a \mapsto b \mapsto c]) \vee (V[a] \wedge \mathit{sft}(W[b \mapsto c]))$;
    $W[b \mapsto c] \leftarrow \mathit{sft}(W[b \mapsto c]) \vee (V[b] \wedge \mathit{sft}(W[c]))$;
    $W[c] \leftarrow \mathit{sft}(W[c]) \vee V[c]$;
  end /* for */
  $T \leftarrow T \cup \{W[a \mapsto b \mapsto c]\}$;
end /* foreach */
return T;

Fig. 2. The algorithm BitSerl.

The following lemma guarantees that the construction of $W[a \mapsto b \mapsto c]$ in the algorithm BitSerl is correct.

Lemma 2. *Let $\mathcal{S}$ be an event sequence $\langle S, T_s, T_e \rangle$. Then, a serial episode $e_1 \mapsto \cdots \mapsto e_n$ occurs in a window $w(\mathcal{S}, t, k)$ if and only if one of the following two statements holds.*

1. *A serial episode* $e_1 \mapsto \cdots \mapsto e_n$ *occurs in a window* $w(\mathcal{S}, t+1, k-1)$.
2. *An event* e_1 *occurs in a window* $w(\mathcal{S}, t, 1)$ *and a serial episode* $e_2 \mapsto \cdots \mapsto e_n$ *occurs in a window* $w(\mathcal{S}, t+1, k-1)$.

Proof. First, we show the if direction. If the statement 1 holds, then there exists a sequence $(a_1, t_1) \ldots, (a_n, t_n) \in S$ of events such that $e_i = a_i$ $(1 \leq i \leq n)$ and $t+1 \leq t_1 < \cdots < t_n < t+k$. Since $t \leq t+1$, the statement holds. On the other hand, if the statement 2 holds, then there exists a sequence $(a_1, t_1) \ldots, (a_n, t_n) \in S$ of events such that $e_i = a_i$ $(1 \leq i \leq n)$, $t \leq t_1 < t+1$, and $t+1 \leq t_2 < \cdots < t_n < t+k$. Since $t \leq t_1 < t+1 \leq t_2 < \cdots < t_n < t+k$, the statement holds.

Next, we show the only-if direction by contraposition, that is, if both of the following statements hold, then a serial episode $e_1 \mapsto \cdots \mapsto e_n$ $(n \geq 2)$ does not occur in $w(\mathcal{S}, t, k)$.

1. (A) A serial episode $e_1 \mapsto \cdots \mapsto e_n$ does not occur in $w(\mathcal{S}, t+1, k-1)$.
2. (B) An event e_1 does not occur in $w(\mathcal{S}, t, 1)$ or (C) a serial episode $e_2 \mapsto \cdots \mapsto e_n$ does not occur in $w(\mathcal{S}, t+1, k-1)$.

Suppose that both of the above statements hold but a serial episode $e_1 \mapsto \cdots \mapsto e_n$ occurs in $w(\mathcal{S}, t, k)$. Then, there exists a sequence $(a_1, t_1), \ldots, (a_n, t_n) \in S$ of events such that $e_i = a_i$ $(1 \leq i \leq n)$ and $t \leq t_1 < \cdots < t_n < t+k$.

Suppose that the statement (B) holds. Then, it holds that either $t_1 < t$ or $t+1 \leq t_1$. Since $t \leq t_1$, it holds that $t+1 \leq t_1$. Hence, it holds that $t+1 \leq t_1 < \cdots < t_n < t+k$, which contradicts the statement (A).

On the other hand, suppose that the statement (C) holds. If $t+1 \leq t_2$, then it holds that $t+1 \leq t_2 < \cdots < t_n < t+k$, which contradicts the statement (C), so it holds that $t_2 < t+1$. Since $t \leq t_1$, and t_i is a natural number, it holds that $t_1 \leq t_2 < t+1 \leq t_1 + 1$, that is, $t_1 = t_2$, which contradicts that $t_1 < t_2$.

Hence, a serial episode $e_1 \mapsto \cdots \mapsto e_n$ does not occur in $w(\mathcal{S}, t, k)$. □

Lemma 3. *The algorithm* BitSerl *is correct.*

Proof. For a window W and an episode E, let *occ* be a function that $occ(W, E) = 1$ if E occurs in W; 0 otherwise. Since $V[\cdot]$ and $W[\cdot]$ are bit vectors with length l, we denote $V[\cdot][i] = (V[\cdot])_i$ and $W[\cdot][i] = (W[\cdot])_i$ for $1 \leq i \leq l$. Then, before the for-loop in the algorithm BitSerl, the following statements hold:

$$\begin{aligned} V[E][i] &= occ(w(\mathcal{S}, i-(k-1), 1), E), \\ W[a \mapsto b \mapsto c][i] &= occ(w(\mathcal{S}, i-(k-1), 1), a \mapsto b \mapsto c), \\ W[b \mapsto c][i] &= occ(w(\mathcal{S}, i-(k-1), 1), b \mapsto c), \\ W[c][i] &= occ(w(\mathcal{S}, i-(k-1), 1), c). \end{aligned}$$

Suppose that, before the d-th iteration of the for-loop, the following statements hold:

$$\begin{aligned} W[a \mapsto b \mapsto c][i] &= occ(w(\mathcal{S}, i-(k-1), d), a \mapsto b \mapsto c), \\ W[b \mapsto c][i] &= occ(w(\mathcal{S}, i-(k-1), d), b \mapsto c), \\ W[c][i] &= occ(w(\mathcal{S}, i-(k-1), d), c). \end{aligned}$$

By Lemma 2, it holds that $occ(w(\mathcal{S}, i-(k-1), d+1), a \mapsto b \mapsto c) = occ(w(\mathcal{S}, i-(k-1+1), d+1-1), a \mapsto b \mapsto c) \vee (occ(w(\mathcal{S}, i-(k-1), 1), a) \wedge occ(w(\mathcal{S}, i-(k-1+1), d+1-1), b \mapsto c)) = W[a \mapsto b \mapsto c][i+1] \vee (V[a][i] \wedge W[b \mapsto c][i+1])$. Similarly, it holds that $occ(w(\mathcal{S}, i-(k-1), d+1), b \mapsto c) = W[b \mapsto c][i+1] \vee (V[b][i] \wedge W[c][i+1])$ and $occ(w(\mathcal{S}, i-(k-1), d+1), c) = W[c][i+1] \vee V[c][i]$. Here, note that $W[\cdot][i+1]$ is corresponding to the shift operation *sft*. Hence, after the d-th iteration of the for-loop, the following statements hold:

$$\begin{aligned} W[a \mapsto b \mapsto c][i] &= occ(w(\mathcal{S}, i-(k-1), d+1), a \mapsto b \mapsto c), \\ W[b \mapsto c][i] &= occ(w(\mathcal{S}, i-(k-1), d+1), b \mapsto c), \\ W[c][i] &= occ(w(\mathcal{S}, i-(k-1), d+1), c). \end{aligned}$$

Hence, by repeating the above procedure in k times, we can obtain the bit vector representing the occurrences of a serial episode $a \mapsto b \mapsto c$. □

After constructing the set of bit vectors representing the occurrences of all serial episodes with size 3 by the algorithm BitSerl, we design the algorithm FreqDmd to extract all of the frequent diamond episodes under the minimum support σ as Figure 3.

---

procedure FreqDmd($S, T_s, T_e, k, \mathcal{E}, \sigma$) /* $\langle S, T_s, T_e \rangle$: event sequence */
/* k: the width of windows, $\mathcal{E}$: event types, σ: the minimum support */
$T \leftarrow$ BitSerl($S, T_s, T_e, \mathcal{E}, k$); $D \leftarrow \emptyset$;
foreach $(s_1, s_2) \in \mathcal{E}$ **do begin**
  $I \leftarrow \emptyset$;
  foreach $W[s_1 \mapsto e \mapsto s_2] \in T$ **do**
    $U[e] \leftarrow W[s_1 \mapsto e \mapsto s_2]$; $I \leftarrow I \cup \{U[e]\}$;
    /* e is regarded as an item */
  $F \leftarrow$ AprioriTid(I, σ);
  /* call AprioriTid [1, 2], where F is the set of frequent itemsets */
  foreach $E \in F$ **do** $D \leftarrow D \cup \{s_1 \mapsto E \mapsto s_2\}$;
end /* foreach */
return D;

Fig. 3. The algorithm FreqDmd.

The algorithm FreqDmd calls the frequent itemset mining algorithm AprioriTid, introduced by Agrawal and Srikant [2], because we can regard a serial episode $s_1 \mapsto e \mapsto s_2$ as an item under fixing a source s_1 and a sink s_2, by Lemma 1. Then, in the algorithm FreqDmd, a serial episode $s_1 \mapsto e \mapsto s_2$ is referred as an event e, and extracts the set of all frequent itemsets by using the set I of bit vectors of the occurrences of $s_1 \mapsto e \mapsto s_2$, which is corresponding to L_1 in the algorithm AprioriTid [2].

Theorem 1. *The algorithm* FreqDmd *extracts all of the frequent diamond episodes from an event sequence by scanning it just once.*

Proof. By Lemma 3, the algorithm BitSerl is correct. Then, by Lemma 1 and the correctness of AprioriTid, the algorithm FreqDmd is correct. Furthermore, the algorithm FreqDmd scans an event sequence just once in the algorithm BitSerl. □

Example 2. Consider the event sequence $\mathcal{S}$ in Example 1 (Figure 1).

Figure 4 describes the set I of bit vectors in the algorithm FreqDmd obtained by applying the algorithm BitSerl to $\mathcal{S}$ under the window width is 5. Here, (s_1, s_2) is a pair of a source s_1 and a sink s_2 and D is a serial episode $s_1 \mapsto e \mapsto s_2$ corresponding to $U[e] = W[s_1 \mapsto e \mapsto s_2]$.

(s_1, s_2)	D	0	1	2	3	4	5	6	7	8	9
(a, a)	$a \mapsto a \mapsto a$	0	0	1	1	1	1	1	1	0	0
	$a \mapsto b \mapsto a$	0	0	1	1	1	1	1	1	0	0
	$a \mapsto c \mapsto a$	0	0	0	0	0	1	1	1	0	0
(b, a)	$b \mapsto a \mapsto a$	0	0	1	1	1	1	1	1	0	0
	$b \mapsto b \mapsto a$	0	0	1	1	1	1	1	1	0	0
	$b \mapsto c \mapsto a$	0	0	0	0	0	1	1	1	0	0
(c, a)	$c \mapsto a \mapsto a$	0	0	1	1	1	0	0	0	0	0
	$c \mapsto b \mapsto a$	0	0	1	1	1	0	0	0	0	0
	$c \mapsto c \mapsto a$	0	0	0	0	0	0	0	0	0	0
(a, b)	$a \mapsto a \mapsto b$	0	0	0	1	1	1	1	1	0	0
	$a \mapsto b \mapsto b$	0	0	0	1	1	1	1	1	0	0
	$a \mapsto c \mapsto b$	0	0	0	0	0	1	1	1	0	0
(b, b)	$b \mapsto a \mapsto b$	0	0	0	1	1	1	1	1	0	0
	$b \mapsto b \mapsto b$	0	0	0	1	1	1	1	1	0	0
	$b \mapsto c \mapsto b$	0	0	0	0	0	1	1	1	0	0

(s_1, s_2)	D	0	1	2	3	4	5	6	7	8	9
(c, b)	$c \mapsto a \mapsto b$	0	0	0	1	1	0	0	0	0	0
	$c \mapsto b \mapsto b$	0	0	0	1	1	0	0	0	0	0
	$c \mapsto c \mapsto b$	0	0	0	0	0	0	0	0	0	0
(a, c)	$a \mapsto a \mapsto c$	0	0	0	0	1	1	1	0	0	0
	$a \mapsto b \mapsto c$	0	0	0	0	1	1	1	0	0	0
	$a \mapsto c \mapsto c$	0	0	0	0	0	0	0	0	0	0
(b, c)	$b \mapsto a \mapsto c$	0	0	0	0	1	1	0	0	0	0
	$b \mapsto b \mapsto c$	0	0	0	0	1	1	0	0	0	0
	$b \mapsto c \mapsto c$	0	0	0	0	0	0	0	0	0	0
(c, c)	$c \mapsto a \mapsto c$	0	0	0	0	1	0	0	0	0	0
	$c \mapsto b \mapsto c$	0	0	0	0	1	0	0	0	0	0
	$c \mapsto c \mapsto c$	0	0	0	0	0	0	0	0	0	0

Fig. 4. The set I of bit vectors for a source s_1 and a sink s_2 in the algorithm FreqDmd. Here, D denotes a serial episode $s_1 \mapsto e \mapsto s_2$ corresponding to $U[e] = W[s_1 \mapsto e \mapsto s_2]$ in Example 2.

Then, Figure 5 describes all of the frequent diamond episodes from $\mathcal{S}$ under the minimum support 50%. Note that the algorithm AprioriTid [1, 2] realizes the repetition of the bit-wise conjunction of bit vectors while satisfying the minimum support.

4 Empirical Results

In this section, by applying the algorithm FreqDmd to bacterial culture data of Osaka Prefectural General Medical Center from 1995 to 1998, which are complete data in [10], we extract frequent diamond episodes concerned with *replacement of bacteria*. Here, we regard a pair of "attribute=value" as an event type.

First, we fix the width of windows as 30 days. Since the database contains the patient information not related to replacement of bacteria such as date,

(s_1, s_2)	F	D
(a, a)	$\{a, b, ab\}$	$a \mapsto a \mapsto a,\ a \mapsto b \mapsto a,\ a \mapsto ab \mapsto a$
(b, a)	$\{a, b, ab\}$	$b \mapsto a \mapsto a,\ b \mapsto b \mapsto a,\ b \mapsto ab \mapsto a$
(c, a)	$\emptyset$	
(a, b)	$\{a, b, ab\}$	$a \mapsto a \mapsto b,\ a \mapsto b \mapsto b,\ a \mapsto ab \mapsto b$
(b, b)	$\{a, b, ab\}$	$b \mapsto a \mapsto b,\ b \mapsto b \mapsto b,\ b \mapsto ab \mapsto b$
(c, b)	$\emptyset$	
(a, c)	$\emptyset$	
(b, c)	$\emptyset$	
(c, c)	$\emptyset$	

Fig. 5. All of the extracted frequent diamond episode from $\mathcal{S}$ under the minimum support 50% and the window width 5 in Example 2.

gender, ward and engineer [10], it is necessary to focus the specified attributes. Then, for a diamond episode of the form $s_1 \mapsto E \mapsto s_2$, we select the attributes age, department, sample, fever, catheter, tracheo, intubation, drainage, WBC (white blood cell) count, medication, Urea-WBC, Urea-Nitocide, Urea-Occultblood, Urea-Protein, Urea-amount, the total amount of bacteria, and the sensitivity of antibiotics as the event types in E, and the detected bacterium as a source s_1 and a sink s_2, in whole 44 attributes.

From the medical viewpoint, in order to extract frequent diamond episodes concerned with replacement of bacteria, it is necessary to extract them based on the same sample. Hence, by fixing the sample, we connect data of every patient with the span of 30 days, which is the width of windows.

Then, Figure 6 describes the number of windows and all of the frequent episodes under the minimum support 0.2%. Here, the column "c-episodes" denotes the number of diamond episodes after selecting $s_1 \mapsto E_1 \mapsto s_2$ from the frequent diamond episodes that has no $s_1 \mapsto E_2 \mapsto s_2$ such that $E_2 \subset E_1$ and $supp(s_1 \mapsto E_1 \mapsto s_2) = supp(s_1 \mapsto E_2 \mapsto s_2)$. We call such a frequent diamond episode a frequent *closed* diamond episode, as similar as a *closed* pattern [11–14].

sample	windows	episodes	c-episodes
catheter/others	94597	1180562	7226
respiratory organs	170633	25629974	163476

Fig. 6. The number of windows and all of the frequent and the frequent closed diamond episodes under the minimum support 0.2%.

Figure 7 describes the bacteria of a source and a sick in the frequent closed diamond episodes. Note that if the sample is respiratory organs, then the bacterium "Staphylococcus aureus" occurs at either a source or a sink in the frequent closed diamond episode under the minimum support 0.2%.

sample	bacterium of a source	bacterium of a sick	c-episodes
catheter/others	Staphylococcus aureus	Staphylococcus aureus	6396
	Bacteroides fragilis	Bacteroides fragilis	596
	Pseudomonas aeruginosa	Pseudomonas aeruginosa	120
	Escherichia coli	Escherichia coli	18
	Enterococcus faecalis	Enterococcus faecalis	1
	Staphylococcus aureus	Pseudomonas aeruginosa	45
	Bacteroides fragilis	Enterococcus faecalis	22
	Enterococcus faecalis	Pseudomonas aeruginosa	22
	Escherichia coli	Staphylococcus aureus	4
	Bacteroides fragilis	Staphylococcus aureus	2
respiratory organs	Staphylococcus aureus	Staphylococcus aureus	140681
	Pseudomonas aeruginosa	Pseudomonas aeruginosa	10512
	Staphylococcus aureus	Pseudomonas aeruginosa	6586
	Pseudomonas aeruginosa	Staphylococcus aureus	5677
	Staphylococcus aureus	Enterobacter cloacae	10
	Staphylococcus aureus	Serratia marcescens	6
	Enterobacter cloacae	Staphylococcus aureus	3
	Staphylococcus aureus	Klebsiella pneumoniae	1

Fig. 7. The bacteria in a source and a sink in the frequent closed diamond episodes.

Figure 8 and 9 describe the events Ant=R, Ant=I, Ant=S of the resistant for antibiotics (Ant) that is resistant (R), intermediate (I) and susceptibility, for the samples of catheter/others and respiratory organs, respectively, such that a source and a sick are different in Figure 7. Here, antibiotics are benzilpenicillin (PcB), synthetic penicillins (PcS), augmentin (Aug), anti-pseudomonas penicillin (PcAP), 1st generation cephems (Cep1), 2nd generation cephems (Cep2), 3rd generation cephems (Cep3), 4th generation cephems (Cep4), anti-pseudomonas cephems (CepAP), aminoglycosides (AG), macrolides (ML), tetracyclines (TC), lincomycins (LCM), chloramphenicols (CP), carbapenems (CBP), and vancomycin (VCM), and RFP/FOM (RFPFOM).

Hence, we can observe the different resistant for antibiotics for the different replacement of bacteria. Furthermore, if the number of episodes is large, then the episodes sometimes contain both Ant=R and Ant=S. We will report the analysis of the results deeply from the medical viewpoints elsewhere.

5 Conclusion

In this paper, we have introduced a diamond episode of the form $s_1 \mapsto E \mapsto s_2$, where s_1 and s_2 are events and E is a set of events. Then, by formulating the *support* of diamond episodes, we have shown that the diamond episode preserves anti-monotonicity. Also we have designed the algorithm BitSerl to construct the set of bit vectors representing the occurrences of all serial episodes with size 3, by using the bit-wise conjunction and disjunction, and the shift operator. Then, we designed the algorithm FreqDmd to extract all of the *frequent*

bacterium of a source	bacterium of a sick		resistant
Staphylococcus aureus	Pseudomonas aeruginosa	45	PcS=R(2), PcAP=S(1), Cep1=R(8), AG=S(6), TC=R(1), CBP=S(14)
Bacteroides fragilis	Enterococcus faecalis	22	PcB=R(2), PcB=S(1), PcAP=S(16), Cep1=R(6), Cep3=S(4), AG=S(2), ML=S(22), TC=S(22), CBP=S(22), VCM=S(1)
Enterococcus faecalis	Pseudomonas aeruginosa	22	PcB=R(1), Cep1=R(3), Cep2=S(1), TC=S(7), CBP=S(11)
Escherichia coli	Staphylococcus aureus	4	Cep3=S(2), TC=S(4), CBP=S(3)
Bacteroides fragilis	Staphylococcus aureus	2	PcB=R(1)

Fig. 8. The resistant of antibiotics for the sample of catheter/others in Figure 7.

diamond episodes from a given event sequence. Finally, we have applied the algorithm FreqDmd to bacterial culture data in order to extract frequent diamond episodes representing replacement of bacteria.

It is main advantage for the algorithm FreqDmd to scan a given event sequence just once, as same as the algorithm AprioriTid [1, 2] or Sect [5]. However, the algorithm FreqDmd is insufficient to give empirical results efficiently, so it is a future work to improve the algorithm FreqDmd more efficiently.

In the evaluation in Section 4, we have introduced a *closed* diamond episode, because of reducing the number of extracted episodes. On the other hand, there are many algorithm to extract frequent closed patterns such as Lcm [11] and Charm [14] for frequent itemsets and Bide [12] and CloSpan [13] for frequent sequences. Then, it is a future work to design an algorithm to extract frequent closed diamond episodes directly, by combining Lcm or Charm instead of AprioriTid in the algorithm FreqDmd.

In this paper, we have given empirical results for bacterial culture data. It is an important future work to apply our algorithm FreqDmd to various data in order to extract frequent diamond episodes.

References

1. R. Agrawal, H. Mannila, R. Srikant, H. Toivonen, A. I. Verkamo: *Fast discovery of association rules*, in U. M. Fayyed, G. Piatetsky-Shapiro, P. Smyth, R. Uthurusamy (eds.): *Advances in Knowledge Discovery and Data Mining*, AAAI/MIT Press, 307–328, 1996.
2. R. Agrawal, R. Srikant: *Fast algorithms for mining association rules in large databases*, Proc. 20th VLDB, 487–499, 1994.
3. R. Agrawal, R. Srikant: *Mining sequential patterns*, Proc. 11th ICDE, 3–14, 1995.
4. C. Bettini, S. Wang, S. Jajodia, J.-L. Lin: *Discovering frequent event patterns with multiple granularities in time sequences*, IEEE Trans. Knowledge and Data Engineering **10**, 222–237, 1998.
5. T. Katoh, K. Hirata, M. Harao: *Mining sectrial episodes from event sequences*, Proc. 10th DS, LNAI **4265**, 137–145, 2006.

bacterium of a source	bacterium of a sick		resistant
Staphylococcus aureus	Pseudomonas aeruginosa	6586	PcB=R(974), PcS=R(4390), Aug=R(964), PcAP=S(1508), Cep1=R(3868), Cep2=R(567), Cep3=S(289), CepAP=S(1236), AG=R(139), AG=S(1868), TC=R(1265), TC=S(111), CBP=R(95), CBP=S(1994), VCM=S(1968), RFPFOM=S(109)
Pseudomonas aeruginosa	Staphylococcus aureus	5677	PcB=R(635), PcS=R(2639), Aug=R(1118), PcAP=S(1334), Cep1=R(2726), Cep2=R(382), Cep3=S(270), CepAP=S(864), AG=R(120), AG=S(1566), ML=S(63), TC=R(1212), TC=S(88), CBP=R(6), CBP=S(1533),VCM=S(457), RFPFOM=S(45)
Staphylococcus aureus	Enterobacter cloacae	10	PcS=R(2), Cep1=R(1), AG=S(1), CBP=S(4)
Staphylococcus aureus	Serratia marcescens	6	AG=S(1), CBP=S(1)
Enterobacter cloacae	Staphylococcus aureus	3	PcS=R(1)
Staphylococcus aureus	Klebsiella pneumoniae	1	—

Fig. 9. The resistant of antibiotics for the sample of respiratory organs in Figure 7.

6. T. Katoh, K. Hirata, M. Harao, S. Yokoyama, K. Matsuoka: *Extraction of sectrial episodes representing changes for drug resistant and replacements of bacteria*, Proc. CME'07, 2007 (to appear).
7. H. Mannila, H. Toivonen, A. I. Verkamo: *Discovery of frequent episodes in event sequences*, Data Mining and Knowledge Discovery **1**, 259–289, 1997.
8. J. Pei, J. Han, B. Mortazavi-Asi, J. Wang, H. Pinto, Q. Chen, U. Dayal, M.-C. Hsu: *Mining sequential patterns by pattern-growth: The PrefixSpan approach*, IEEE Trans. Knowledge and Data Engineering **16**, 1–17, 2004.
9. R. Srikant, R. Agrawal: *Mining sequential patterns: Generalizations and performance improvements*, Proc. 5th EDBT, 3–17, 1996.
10. S. Tsumoto: *Guide to the bacteriological examination data set*, in E. Suzuki (ed.): Proc. International Workshop of KDD Challenge on Real-World Data (KDD Challenge 2000), 8–12, 2000.
11. T. Uno, T. Asai, Y. Uchida, H. Arimura: *An efficient algorithm for enumerating closed patterns in transaction databases*, Proc. 7th DS, LNAI **3245**, 16–31, 2004.
12. J. Wang, J. Han: *BIDE: Efficient mining of frequent closed sequences*, Proc. 20th ICDE, 2004.
13. X. Yan, J. Han, R. Afshar: *CloSpan: Mining closed sequential patterns in large datasets*, Proc. 3rd SDM, 2003.
14. M. J. Zaki, C.-J. Hsiao: CHARM*: An efficient algorithm for closed itemset mining*, Proc. 2nd SDM, 457–478, 2002.