Mining Sectorial 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
t_katou@dumbo.ai.kyutech.ac.jp
{hirata,harao}@ai.kyutech.ac.jp

Abstract. In this paper, we introduce a *sectorial episode* of the form $C \mapsto r$, where C is a set of events and r is an event. The sectorial episode $C \mapsto r$ means that every event of C is followed by an event r. Then, by formulating the *support* and the *confidence* of sectorial episodes, in this paper, we design the algorithm Sect to extract all of the *sectorial episodes that are frequent and confidential* from a given event sequence by traversing it just once. Finally, by applying the algorithm Sect to bacterial culture data, we extract sectorial episodes representing *drug-resistant change*.

1 Introduction

The *sequential pattern mining* [3, 6, 7, 9, 10] 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, 5] introduced by Mannila *et al.* [5] 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.* [5] 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.

Concerned with episode mining, in this paper, we introduce a *sectorial episode* of the form $C \mapsto r$, where C is a parallel episode and r is an event. The sectorial episode $C \mapsto r$ means that every event of C is followed by an event r, so we can regard every event in C as a *causation* of r. Note that, since a sectorial episode is captured as the direct precedent-subsequent relationship between events, it is just a candidate of *causality* in database (*cf.* [11]).

We formulate the *support* $supp(C \mapsto r)$ of a sectorial episode $C \mapsto r$ as the ratio of the number of k-windows (i.e., a window with width k) in which $C \mapsto r$

occurs for the number of all k-windows. For the *minimum support* σ such that $0 < \sigma < 1$, we say that $C \mapsto r$ is *frequent* if $supp(C \mapsto r) \geq \sigma$. Also we can show that the sectorial episode preserves *anti-monotonicity*, that is, for $C_1 \subseteq C_2$, if $C_2 \mapsto r$ is frequent then so is $C_1 \mapsto r$. Furthermore, by regarding a sectorial episode as an association rule [1, 2], we introduce the *confidence* $conf(C \mapsto r)$ of a sectorial episode $C \mapsto r$ as the ratio of the number of k-windows in which $C \mapsto r$ occurs for the number of all k-windows in which C occurs. For the *minimum confidence* γ such that $0 < \gamma < 1$, we say that $C \mapsto r$ is *confidential* if $conf(C \mapsto r) \geq \gamma$.

The purpose of this paper is to design the algorithm to extract all of the *sectorial episodes that are frequent and confidential* from a given event sequence. Since our sectorial episode is a combination of parallel and serial episodes, it is possible to extract all of the sectorial episodes that are frequent and confidential by combining the algorithms designed in [5]. On the other hand, in this paper, we design another algorithm Sect to extract them efficiently and appropriate to sectorial episodes.

The algorithm Sect consists of the following two procedures. The first procedure Scan is to extract all of the frequent sectorial episodes of the form $C \mapsto r$ such that $|C| = 1$ and then store the information of windows in which $C \mapsto r$ occurs. The second procedure is to extract all of the frequent sectorial episodes of the form $C \mapsto r$ such that $|C| \geq 2$ for every event r stored by Scan. Here, the frequency of the constructed sectorial episodes is computed by using the anti-monotonicity, as similar as the well-known algorithm AprioriTid [1, 2].

Then, we show that the algorithm Scan runs in $O((l+k)|\mathcal{E}|^2)$ time and space, where l is the time span between the starting time and the ending time in a given event sequence, k is the width of windows and $\mathcal{E}$ is a set of all events, by traversing the event sequence just once. Hence, we show that the algorithm Sect extracts all of the sectorial episodes that are frequent and confidential from the sectorial episodes by Scan in $O((l+k)|\mathcal{E}|^{2M+1})$ time and $O((l+k)|\mathcal{E}|^M)$ space without traversing the event sequence, where $M = \max\{|C| \mid C \mapsto r \text{ is frequent}\}$.

Finally, we apply the algorithm Sect to bacterial culture data. Note that, from the medical viewpoint, in order to extract sectorial episodes concerned with *drug-resistant change*, it is necessary to extract them based on the same detected bacterium and the same sample. Hence, in this paper, we divide the database into *pages* for the detected bacterium and the sample in whole 44 attributes, and then extract sectorial episodes representing drug-resistant changes from them.

2 Sectorial Episodes

As similar as [5], 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 $l_{\mathcal{S}}$. For an event sequence $\mathcal{S} = (S, T_s, T_e)$, we denote the set of all event types of $\mathcal{S}$ at t, that is, $\{E \subseteq \mathcal{E} \mid (E, t) \in S\}$ by $evtyp(\mathcal{S}, t)$.

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. For a window $W = (w, t_s, t_e)$, we denote the starting time t_s of W by $st(W)$ and the ending time t_e of W by $et(W)$.

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 $w(\mathcal{S}, t, k)$. For a k-window $W = w(\mathcal{S}, t, k)$, it is obvious that $st(W) = t$ and $et(W) = t + k$.

Note that we can regard a set $E = e_1 \cdots e_m$ of event types as a *parallel episode* [5]. Here, we call the above m the *size* of E. Furthermore, in this paper, we newly introduce the following *sectorial episode*.

Definition 1. Let $c_1 \cdots c_m \subseteq \mathcal{E}$ be a parallel episode and r an event type. Then, a *sectorial episode* is of the following form.

$$X = c_1 \cdots c_m \mapsto r.$$

In this paper, we regard every event in $c_1 \cdots c_m$ in a sectorial episode X as a *causation* of r, so we call every c_i and r a *causal type* and a *resulting type* of X, respectively. For a set $C = c_1 \cdots c_m$, we also denote $c_1 \cdots c_m \mapsto r$ by $C \mapsto r$. In particular, we call a sectorial episode $C \mapsto r$ such that $r \in C$ *trivial*.

We call both a parallel episode and a sectorial episode *episodes* simply.

Definition 2. 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 $et(e, \mathcal{S}) = \max\{t \mid t \in T(e, \mathcal{S})\}$.

We say that a parallel episode $e_1 \cdots e_m$ *occurs* in $\mathcal{S}$ if every e_i occurs in $\mathcal{S}$. Also we say that a sectorial episode $c_1 \cdots c_m \mapsto r$ *occurs* in $\mathcal{S}$ if r occurs in $\mathcal{S}$, and, for every i $(1 \leq i \leq m)$, c_i occurs in $\mathcal{S}$ and $st(c_i, \mathcal{S}) < et(r, \mathcal{S})$.

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 $\mathcal{S}$.

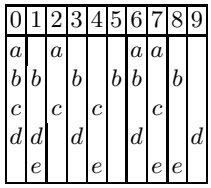

Fig. 1. An event sequence S in Example 1.

Example 1. Let $\mathcal{E} = \{a, b, c, d, e\}$. Then, Figure 1 describes an event sequence $\mathcal{S} = (S, 0, 10)$ on $\mathcal{E}$ where:

$$S = \langle (abcd, 0), (bde, 1), (ac, 2), (bd, 3), (ce, 4), \\ (b, 5), (abd, 6), (ace, 7), (be, 8), (d, 9) \rangle.$$

Furthermore, the event sequence $w = (\langle (ac, 2), (bd, 3), (ce, 4) \rangle, 2, 5)$ is a 3-window of $\mathcal{S}$ starting from 2, that is, $w = w(\mathcal{S}, 2, 3) = (\langle (ac, 2), (bd, 3), (ce, 4) \rangle, 2, 5)$.

For the above event sequence $\mathcal{S}$, it holds that $T(a, \mathcal{S}) = \{1, 2, 6, 7\}$, $st(a, \mathcal{S}) = 1$ and $et(a, \mathcal{S}) = 7$. Also there exist 12 3-windows, of which starting time is from -2 to 9. Furthermore, for the above window w, sectorial episodes $ab \mapsto c$, $bc \mapsto e$ and $acd \mapsto e$ occur in w, for example.

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 sectorial episodes). *Let C_1 and C_2 be parallel episodes such that $C_1 \subseteq C_2$. If $C_2 \mapsto r$ is frequent, then so is $C_1 \mapsto r$.*

Proof. It is sufficient to show that $W(C_2 \mapsto r, \mathcal{S}, k) \subseteq W(C_1 \mapsto r, \mathcal{S}, k)$. Suppose that $l \in W(C_2 \mapsto r, \mathcal{S}, k)$ and let W_l be the l-th k-window in $\mathcal{S}$. Then, it holds that $W_l = w(W(C_2 \mapsto r, \mathcal{S}, k), l, k)$. For every $c \in C_2$, it holds that $st(c, W_i) < et(r, W_l)$. Since $C_1 \subseteq C_2$, it holds that $st(c', W_i) < et(r, W_l)$ for every $c' \in C_1$, so $C_1 \mapsto r$ occurs in W_l. Hence, it holds that $l \in W(C_1 \mapsto r, \mathcal{S}, k)$. □

By regarding a sectorial episode $C \mapsto r$ as an association rule, we can introduce the *confidence* $conf_{\mathcal{S},k}(C \mapsto r)$ of $C \mapsto r$ in $\mathcal{S}$ w.r.t. k as follows.

$$conf_{\mathcal{S},k}(C \mapsto r) = \frac{|W(C \mapsto r, \mathcal{S}, k)|}{|W(C, \mathcal{S}, k)|}.$$

Definition 4. Let γ be the *minimum confidence* such that $0 < \gamma < 1$. Then, we say that a sectorial episode $C \mapsto r$ is *confidential* if $conf(C \mapsto r) \geq \gamma$.

In the following, the subscripts $\mathcal{S}$ and k in $freq_{\mathcal{S},k}(X)$, $supp_{\mathcal{S},k}(X)$ and $conf_{\mathcal{S},k}(X)$ are omitted, if they are clear by the context.

3 Algorithm to Extract Sectorial Episodes

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

We assume the lexicographic order $\prec$ on $\mathcal{E}$. Also we extend $\prec$ to $2^{\mathcal{E}}$ as follows: For sets $Y = y_1 \cdots y_m, Z = z_1 \cdots z_n$ of event types, $Y \prec Z$ if there exists an i $(1 \leq i \leq n)$ such that $y_j = z_j$ $(1 \leq j \leq i-1)$ and $y_i \prec z_i$.

The algorithm Scan described by Figure 2 extracts all of the frequent sectorial episodes of the form $c \mapsto r$, where c and r are events, and stores the set of labels of k-windows $W(c \mapsto r, \mathcal{S}, k)$ in which the sectorial episode $c \mapsto r$ occurs as $W[c][r]$ and the set of labels of k-windows $W(r, \mathcal{S}, k)$ for every $r \in \mathcal{E}$ as $V[r]$, by traversing $\mathcal{S}$ just once as similar as AprioriTid [1, 2].

---

procedure Scan$(\mathcal{S}, \sigma, k)$
$C \leftarrow \emptyset$; $P \leftarrow \emptyset$; $size \leftarrow T_e - T_s + k - 1$;
foreach $e \in evtyp(\mathcal{S}, T_s - k)$ **do** $E \leftarrow E \cup \{e\}$; $V[e] \leftarrow V[e] \cup \{T_s - k\}$;
foreach $e \in evtyp(\mathcal{S}, T_e)$ **do** $E \leftarrow E \cup \{e\}$; $V[e] \leftarrow V[e] \cup \{T_e\}$;
for $i = T_s - k + 1$ **to** $T_e - 1$ **do begin**
  $R \leftarrow \emptyset$;
  foreach $r \in evtyp(w(S, i, k), i + k - 1)$ **do begin**
    $R \leftarrow R \cup \{r\}$; $E \leftarrow E \cup \{r\}$;
    for $j = i$ **to** $i + k$ **do** $V[r] \leftarrow V[r] \cup \{j\}$;
    foreach $c \in C - \{r\}$ **do**
      $P \leftarrow P \cup \{(c, r)\}$; $t \leftarrow et(c, w(S, i, k-1))$;
      for $j = i$ **to** t **do** $W[c][r] \leftarrow W[c][r] \cup \{j\}$;
  end /* foreach */
  $C \leftarrow C - evtyp(w(S, i, k), i)$; $C \leftarrow C \cup R$;
end /* for */
return $(\{(c, r, W[c][r]) \mid (c, r) \in P, |W[c][r]| \geq \sigma \cdot size\}, \{(e, V[e]) \mid e \in E\})$;

Fig. 2. The algorithm Scan.

Lemma 2. *Let $\mathcal{S}$ be an event sequence on $\mathcal{E}$ and k the width of windows. Then, the algorithm* Scan *extracts all of the frequent sectorial episodes of the form $c \mapsto r$ in $O((l_{\mathcal{S}} + k)|\mathcal{E}|^2)$ time and space, by traversing $\mathcal{S}$ just once.*

Proof. First, we show the correctness of the construction of R and C. Fix an index i $(T_s + k - 1 \leq i \leq T_e - 1)$. Then, in the foreach-loop, the algorithm Scan stores all elements in $evtyp(w(S, i, k), i + k - 1)$, where $et(w(S, i, k)) = i + k$, as R. Also it has stored all elements in $\bigcup_{j=i}^{i+k-1} evtyp(w(S, i, k), j)$ as C.

Consider the case shifting i to $i+1$. Since every element of R becomes a causal type and every element of $evtyp(w(S, i, k), i) - \bigcup_{j=i+1}^{i+k-1} evtyp(w(S, i, k), j)$ does not

become a causal type, the algorithm SCAN updates C to $C - evtyp(w(S,i,k),i)$ and then adds R to C.

Next, for $r \in R$ and $c \in C - \{r\}$ on the i-th for-loop, consider the sectorial episode $c \mapsto r$. Let t be $et(c, w(S,i,k-1))$. Then, the causal type c at t is the nearest one to the resulting type r in $w(S,i,k-1)$, so it holds that $c \mapsto r$ appears in the sequence $w(S,j,1)$ of windows for $i \leq j \leq t$. Hence, by the inner for-loop, the algorithm SCAN adds the labels of such a sequence of windows to $W[c][r]$.

Hence, the algorithm SCAN stores all pairs (c,r) such that $c \mapsto r$ appears in some k-window in $\mathcal{S}$ as P, so SCAN outputs the set of all frequent sectorial episodes of the form $c \mapsto r$.

Since the number of the outer for-loop is $T_e - T_s + k$ and the running time in the for-loop is $O(|\mathcal{E}|^2)$, the time complexity of the algorithm SCAN is $O(2|\mathcal{E}| + (T_e - T_s + k)|\mathcal{E}|^2 + |\mathcal{E}|^2) = O((l_{\mathcal{S}} + k)|\mathcal{E}|^2)$, where the first term $O(2|\mathcal{E}|)$ means the time of the first two foreach loops and the third term $O(|\mathcal{E}|^2)$ means the time checking frequency. Also the space of $W[c][r]$ and $V[e]$ are $O((l_{\mathcal{S}} + k)|\mathcal{E}|^2)$ and $O((l_{\mathcal{S}} + k)|\mathcal{E}|)$, respectively, so the space complexity of the algorithm SCAN is $O((l_{\mathcal{S}} + k)|\mathcal{E}|^2)$. It is obvious that the algorithm SCAN traverses $\mathcal{S}$ just once. □

After applying the algorithm SCAN, we apply the algorithm SECT described by Figure 3 to extract all of the sectorial episodes that are frequent and confidential. Here, the algorithm SECT extracts all frequent sectorial episodes $C \mapsto r$ where $|C| \geq 2$, by designing an APRIORITID-like procedure for every resulting type $r \in R$ [1, 2]. Here, the function $subs(S,T)$ returns 1 if S contains all of the subsets of T of which length is $|T| - 1$. For every resulting type $r \in R$, the algorithm SECT extracts all of the causal types Xc and stores them as $C[r]$ and $D[r]$, where $C[r]$ denotes the set of all causal types that $Xc \mapsto r$ is frequent, and $D[r]$ denotes the set of all causal types that $Xc \mapsto r$ is frequent and confidential.

The algorithm SECT computes the support of $Xc \mapsto r$ by using $W(X, \mathcal{S}, k)$ and $W(c, \mathcal{S}, k)$. The following lemma guarantees the correctness of it.

Lemma 3. *Let C and D be parallel episodes and r an event type such that $r \notin C \cup D$. Then, the following statement holds:*

1. $W(C \cap D, \mathcal{S}, k) = W(C, \mathcal{S}, k) \cap W(D, \mathcal{S}, k)$.
2. $W(C \cap D \mapsto r, \mathcal{S}, k) = W(C \mapsto r, \mathcal{S}, k) \cap W(D \mapsto r, \mathcal{S}, k)$.

Proof. The statement 1 is obvious. For the statement 2, suppose that $C = c_1 \cdots c_m$ and $D = d_1 \cdots d_n$. Also let l be a label such that $l \in W(C \cap D \mapsto r, \mathcal{S}, k)$. Then, $C \cap D \mapsto r$ occurs in the l-th k-window $w(\mathcal{S}, l, k)$, so it holds that $st(c_i, \mathcal{S}) < et(r, \mathcal{S})$ and $st(d_j, \mathcal{S}) < et(r, \mathcal{S})$ for every i and j ($1 \leq i \leq n$, $1 \leq j \leq m$), which implies that $l \in W(C \mapsto r, \mathcal{S}, k) \cap W(D \mapsto r, \mathcal{S}, k)$. The converse direction similarly holds. □

In order to maintain the computation of the support, we use the bit vector a of labels of windows from $T_s - k + 1$ to $T_e - 1$, which is a familiar technique for implementing the algorithm APRIORITID [1, 2]. For the bit vector a of the set A of labels of windows, $i \in A$ if and only if $a_i = 1$. Then, we can compute the intersection $A \cap B$ of two set of labels of windows as the bitwise logical product $a \odot b$ of two bit vectors a of A and b of B.

procedure SECT($\mathcal{S}, \sigma, \gamma, k$) /* $\mathcal{S} = (S, T_s, T_e)$, $0 < \sigma, \gamma < 1$, $k > 0$ */
$(F, E) \leftarrow$SCAN($\mathcal{S}, \sigma, k$);
$R \leftarrow \{r \mid (c, r, W) \in F\}$; *size* $\leftarrow T_e - T_s + k - 1$;
foreach $r \in R$ **do begin**
  $C_1 \leftarrow \{c \mid (c, r, W) \in F\}$; $W[c] \leftarrow \{W \mid (c, r, W) \in F\}$; $V[c] \leftarrow \{V \mid (c, V) \in E\}$;
  $D_1 \leftarrow \{c \mid (c, r, W) \in F$ and $|W[c]| \geq \gamma |V[c]|\}$; $m \leftarrow 1$;
  while $C_m \neq \emptyset$ **do begin**
    $C_{m+1} \leftarrow \emptyset$; $D_{m+1} \leftarrow \emptyset$;
    foreach $X \in C_m$ and $c \in C_1$ s.t. $X \prec c$ **do**
      if $|W[X] \cap W[c]| \geq \sigma \cdot size$ and $subs(C_m, Xc) = 1$ **then**
        $C_{m+1} \leftarrow C_{m+1} \cup \{Xc\}$; $W[Xc] \leftarrow W[X] \cap W[c]$; $V[Xc] \leftarrow V[X] \cap V[c]$;
        if $|W[Xc]| \geq \gamma |V[Xc]|$ **then** $D_{m+1} \leftarrow D_{m+1} \cup \{Xc\}$;
    $m \leftarrow m + 1$;
  end /* while */
  $C[r] \leftarrow \bigcup_{1 \leq i \leq m} C_i$; $D[r] \leftarrow \bigcup_{1 \leq i \leq m} D_i$;
end /* foreach */
return $\{D[r] \mapsto r \mid r \in R\}$;

Fig. 3. The algorithm SECT.

Theorem 1. *Let $\mathcal{S}$ be an event sequence on $\mathcal{E}$. Suppose that $M = \max\{|C| \mid C \in C[r], r \in R\}$. Then, the algorithm* SECT *extracts all of the sectorial episodes that are frequent and confidential in $O((l_{\mathcal{S}}+k)|\mathcal{E}|^{2M+1})$ time and $O((l_{\mathcal{S}}+k)|\mathcal{E}|^{M})$ space by traversing $\mathcal{S}$ just once.*

Proof. By Lemma 2, the algorithm SCAN returns the set F of triples $(c, r, W[c][r])$ such that $c \mapsto r$ is frequent and $W[c][r] = W(c \mapsto r, \mathcal{S}, k)$. Let R be a set $\{r \mid (c, r, W) \in F\}$. For every $r \in R$, it is sufficient to show that $C[r]$ in SECT is the set of all causal types X such that $X \mapsto r$ is frequent and confidential.

For $m \geq 1$, suppose that C_m is the set of all causal types X such that $|X| = m$ and $X \mapsto r$ is frequent. Then, consider the case $m + 1$. Suppose that $X \in C_m$ and $c \in C_1$ such that $X \prec c$. Since $X \prec c$, it holds that $c \notin X$, so $|Xc| = m + 1$. Since $subs(C_m, Xc) = 1$, every subset $Y \subset Xc$ such that $|Y| = m$ is contained in C_m. By induction hypothesis, $Y \mapsto r$ is frequent. Note that $W[X]$ in the algorithm SECT denotes $W(X \mapsto r, \mathcal{S}, k)$. Then, by Lemma 3.2, it holds that $W[X] \cap W[c] = W(X \mapsto r, \mathcal{S}, k) \cap W(c \mapsto r, \mathcal{S}, k) = W(Xc \mapsto r, \mathcal{S}, k)$. Hence, the condition $|W[X] \cap W[c]| \geq \sigma \cdot size$ means that $Xc \mapsto r$ is frequent, so it holds that if $Xc \in C_{m+1}$ then $Xc \mapsto r$ is frequent.

Note that the algorithm SECT construct all causal types Y such that $Y = Xc$ and $Y \mapsto r$ is frequent from every $X \in C_m$ and $c \in C_1$. Hence, C_{m+1} is the set of all causal types Y such that $|Y| = m + 1$ and $Y \mapsto r$ is frequent. Since $C[r]$ is the set $\bigcup_{1 \leq i \leq m} C_i$ such that $C_i \neq \emptyset$ $(1 \leq i \leq m)$ and $C_{m+1} = \emptyset$, $C[r]$ in SECT is the set of all causal types X such that $X \mapsto r$ is frequent.

Finally, since $V[X]$ in SECT denotes $W(X, \mathcal{S}, k)$ and by Lemma 3.1, the condition $|W[X]| \geq \gamma |V[X]|$ means that $X \mapsto r$ is confidential. Hence, every D_i is the set of all causal types X such that $|X| = i$ and $X \mapsto r$ is frequent and

confidential, so $D[r]$ in SECT is the set of all causal types X such that $X \mapsto r$ is frequent and confidential.

Next, consider the time and space complexity of SECT. For the inner foreach-loop, it holds that $|C_m| \leq |\mathcal{E}|^M$. Since both $W[X]$ and $W[c]$ are regarded as bit vectors with length $l_{\mathcal{S}}+k-1$, the time to check whether or not $|W[X] \cap W[c]| \geq \sigma \cdot \mathit{size}$ is $O(l_{\mathcal{S}}+k)$. Also we can determine whether $\mathit{subs}(C_m, Xc) = 1$ or 0 by flipping 1 to 0 in the bit vector of $W[Xc]$ and then by checking whether or not it is contained by C_m. Furthermore, since $V[e]$ is a bit vector with length $l_{\mathcal{S}}+k-1$, the time to check whether or not $|W[Xc]| \geq \gamma|V[Xc]|$ is $O(l_{\mathcal{S}}+k)$ time. Then, the time complexity of the inner foreach-loop is $O(l_{\mathcal{S}}+k+(l_{\mathcal{S}}+k)|\mathcal{E}|^M+l_{\mathcal{S}}+k) = O((l_{\mathcal{S}}+k)|\mathcal{E}|^M)$. Since the number of the outer and inner foreach-loop is at most $|\mathcal{E}|$ and $|\mathcal{E}|^M$, respectively, and by Lemma 2, the time complexity of SECT is $O((l_{\mathcal{S}}+k)|\mathcal{E}|^2 + ((l_{\mathcal{S}}+k)|\mathcal{E}|^M)|\mathcal{E}|^M|\mathcal{E}|) = O((l_{\mathcal{S}}+k)|\mathcal{E}|^{2M+1})$. The space complexity is the space of bit vectors of $W[c][r]$ and $V[e]$, that is, $O((l_{\mathcal{S}}+k)|\mathcal{E}|^2 + (l_{\mathcal{S}}+k)|\mathcal{E}|^M + (l_{\mathcal{S}}+k)|\mathcal{E}|^{M-1}) = O((l_{\mathcal{S}}+k)|\mathcal{E}|^M)$.

Finally, since the algorithm SECT does not traverse $\mathcal{S}$ and by Lemma 2, the algorithm SECT traverses $\mathcal{S}$ just once. □

Example 2. Consider the event sequence $\mathcal{S}$ given in Figure 1. We give an running example of SECT$(\mathcal{S}, 0.5, 0.6, 3)$. Here, $W(\mathcal{S}, 3) = 12$.

Consider SCAN$(\mathcal{S}, 0.5, 3)$. Figure 4 (left) describes the sets C, $\mathit{evtyp}(w(S,i,3),i)$ and R of event types at the end of foreach-loop for every i $(-1 \leq i \leq 8)$ in SCAN. Then, Figure 4 (right) describes the output of SCAN$(\mathcal{S}, 0.5, 3)$. Here, the column *freq* denotes the number of elements of $W[c][r]$ and the symbol $\bullet$, which is a label that $c \mapsto r$ is frequent.

Hence, SCAN$(\mathcal{S}, 0.5, 3)$ returns (F, E), where F and E are the sets of triples and of pairs, respectively, as follows.

$$F = \left\{ \begin{array}{l} (a, b, 1111000110),\ (c, b, 1111110110),\ (b, d, 1110011011), \\ (c, d, 1111010010),\ (a, e, 1101001110),\ (b, e, 1101101100), \\ (d, e, 1101101100),\ (b, c, 0111101100),\ (d, c, 0111101100) \end{array} \right\},$$
$$E = \left\{ \begin{array}{l} (a, 111101111100),\ (b, 111111111110),\ (c, 111111111100), \\ (d, 111111111111),\ (e, 011111111110) \end{array} \right\}.$$

Next, consider SECT$(\mathcal{S}, 0.5, 0.6, 3)$. From the set F, we obtain R as $\{b, c, d, e\}$. In the following, we consider the constructions of $C[r]$ and $D[r]$, respectively, for every $r \in R$.

For $b \in R$, it holds that $C_1 = \{a, c\}$. Since $W[a] \cap W[c] = 1111000110 \odot 1111110110 = 1111000110$, it holds that $|W[a] \cap W[c]| = 6$, so it holds that $C_2 = \{ac\}$. Hence, it holds that $C[b] = \{a, c, ac\}$.

On the other hand, for $C_1 = \{a, c\}$, it holds that $|W[a]| = 6$ and $|V[a]| = 10$, while $|W[c]| = 8$ and $|V[c]| = 9$. Then, it holds that $D_1 = \{a, c\}$. Also since $V[a] \cap V[c] = 111101111100 \odot 111111111100 = 111101111100$, it holds that $|V[ac]| = 9$. Since $|W[ac]| = 6$, it holds that $D_2 = \{ac\}$, so $D[b] = \{a, c, ac\}$.

For $c \in R$, it holds that $C_1 = \{b, d\}$. Since $W[b] \cap W[d] = 0111101100 \odot 0111101100 = 0111101100$, it holds that $|W[b] \cap W[d]| = 6$, so it holds that $C_2 = \{bd\}$. Hence, it holds that $C[c] = \{b, d, bd\}$.

i	C	$evtyp(w(S,i,3),i)$	R
-2	$\emptyset$	$\emptyset$	$abcd$
-1	$abcd$	$\emptyset$	bde
0	$abced$	acd	ac
1	$abce$	be	bd
2	$abcd$	ac	ce
3	$bede$	bd	b
4	bce	ce	ad
5	abd	b	ac
6	acd	ad	be
7	bce	ac	d
8	abe	be	$\emptyset$

c	r	$W[c][r]$									$freq$	
		-1	0	1	2	3	4	5	6	7	8	
a	b	1	1	1	1	0	0	0	1	1	0	6 •
c	b	1	1	1	1	1	1	0	1	1	0	8 •
d	b	1	1	1	0	1	0	0	1	0	0	5
a	d	1	1	1	1	0	0	0	0	1	0	5
b	d	1	1	1	0	0	1	1	0	1	1	7 •
c	d	1	1	1	1	0	1	0	0	1	0	6 •
a	e	1	1	0	1	0	0	1	1	1	0	6 •
b	e	1	1	0	1	1	0	1	1	0	0	6 •
c	e	1	1	0	1	0	0	0	1	1	0	5
d	e	1	1	0	1	1	0	1	1	0	0	6 •
b	a	0	1	1	0	0	1	1	1	0	0	5
c	a	0	1	0	0	0	1	0	0	0	0	2
d	a	0	1	1	0	0	0	1	1	0	0	4
e	a	0	1	1	0	0	1	0	0	0	0	3
a	c	0	1	0	1	0	0	1	1	0	0	4
b	c	0	1	1	1	1	0	1	1	0	0	6 •
d	c	0	1	1	1	1	0	1	1	0	0	6 •
e	c	0	1	1	0	0	0	0	0	0	0	2
e	b	0	0	1	0	1	1	0	1	1	0	5
e	d	0	0	1	0	0	1	0	0	1	1	4

Fig. 4. C, $evtyp(w(S,i,3),i)$ and R at the end of foreach-loop (left) and $W[c][r]$ in SCAN (right).

On the other hand, for $C_1 = \{b, d\}$, it holds that $|W[b]| = 6$ and $|V[b]| = 11$, while $|W[d]| = 6$ and $|V[d]| = 12$. Then, it holds that $D_1 = \emptyset$. Also since $V[b] \cap V[d] = 111111111110 \odot 111111111111 = 111101111110$, it holds that $|V[bd]| = 11$. Since $|W[bd]| = 6$, it holds that $D_2 = \emptyset$, so $D[b] = \emptyset$.

For $d \in R$, it holds that $C_1 = \{b, c\}$. Since $W[b] \cap W[c] = 1110011011 \odot 1111010010 = 1110010010$, it holds that $|W[b] \cap W[d]| = 5$, so it holds that $C_2 = \emptyset$. Hence, it holds that $C[d] = \{b, c\}$.

On the other hand, for $C_1 = \{b, c\}$, it holds that $|W[b]| = 7$ and $|V[b]| = 11$, while $|W[c]| = 6$ and $|V[c]| = 10$. Then, it holds that $D_1 = \{b, c\}$. Also since $V[b] \cap V[c] = 111111111110 \odot 111111111100 = 111101111100$, it holds that $|V[bc]| = 10$. Since $|W[bc]| = 5$, it holds that $D_2 = \emptyset$, so $D[d] = \{b, c\}$.

For $e \in R$, it holds that $C_1 = \{a, b, d\}$. Then, the following statement holds.

$$W[a] \cap W[b] = 1101001110 \odot 1101101100 = 1101001100,$$
$$W[a] \cap W[d] = 1101001110 \odot 1101101100 = 1101001100,$$
$$W[b] \cap W[d] = 1101101100 \odot 1101101100 = 1101101100.$$

Then, it holds that $|W[a] \cap W[b]| = 5$, $|W[a] \cap W[d]| = 5$ and $|W[b] \cap W[d]| = 6$. Hence, it holds that $C_2 = \{bd\}$, so it holds that $C[e] = \{a, b, d, bd\}$.

On the other hand, for $C_1 = \{a, b, d\}$, it holds that $|W[a]| = 6$, $|V[a]| = 9$, $|W[b]| = 6$, $|V[b]| = 11$, $|W[d]| = 6$ and $|V[d]| = 12$. Then, it holds that $D_1 = \{a\}$.

Also since $|V[a] \cap V[b]| = 9$, $|V[a] \cap V[d]| = 9$ and $|V[b] \cap V[d]| = 11$, it holds that $D_2 = \emptyset$, so $D[e] = \{a\}$.

As the result, the algorithm $\textsc{Sect}(\mathcal{S}, 0.5, 0.6, 3)$ returns the following frequent sectorial episodes, where ones with bold faces are frequent and confidential.

$$\begin{array}{llll} \mathbf{a \mapsto b} & \mathbf{c \mapsto b} & \mathbf{ac \mapsto b} & \\ b \mapsto c & d \mapsto c & bd \mapsto c & \\ \mathbf{b \mapsto d} & \mathbf{c \mapsto d} & & \\ \mathbf{a \mapsto e} & b \mapsto e & d \mapsto e & bd \mapsto e \end{array}$$

4 Empirical Results

In this section, by applying the algorithm Sect to bacterial culture data, which are complete data in [8] from 1995 to 1998, we extract sectorial episodes concerned with *drug-resistant change*.

First, we fix the width of windows as 30 days. Since the database contains the patient information not related to drug-resistant change such as date, gender, ward and engineer [8], it is necessary to focus the specified attributes. Then, 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, the total amount of bacteria, the detected bacterium, and the sensitivity of antibiotics as the causal types, and the sensitivity of antibiotics as the resulting type.

From the medical viewpoint, in order to extract sectorial episodes concerned with drug-resistant change, it is necessary to extract them based on the same detected bacterium and the same sample. Hence, in this paper, we divide the database into *pages* for the detected bacterium and the sample described as Figure 5 in whole 44 attributes. The column "patients" denotes the number of different patients consisting of more than two records in a page, and the column "max." denotes the maximum number of records for patients in a page. Here, the detected bacteria are Staphylococci (`bac1`), Enteric bacteria (`bac7`), glucose-nonfermentative gram-negative bacteria (`bac8`) and Anaerobes (`bac11`), and the samples are catheter/others (`spl1`), urinary and genital organs (`spl3`) and respiratory organs (`spl5`). Then, by connecting all records in a page into one event sequence such that the span between all records for one patient and ones for other patient is at least 30 days (the width of windows), we apply the algorithm Sect to it.

For the minimum support $\sigma = 0.15$ and the minimum confidence $\gamma = 0.8$, the column "episodes" in Figure 5 describes the number of the extracted sectorial episodes. Furthermore, we focus on the extracted sectorial episode $C \mapsto r$ such that, for antibiotics `Ant`, C contains "`Ant=S` (susceptibility)" and r is "`Ant=R` (resistant)." In the column "antibiotics" in Figure 5, `Ant`(n) denotes that n is the number of extracted sectorial episodes of the form $C \mapsto$ (`Ant=R`) such that (`Ant=S`) $\in C$. Here, antibiotics are benzilpenicillin (`PcB`), augmentin (`Aug`), anti-pseudomonas penicillin (`PcAP`), 1st generation cephems (`Cep1`), 2nd generation cephems (`Cep2`), 3rd generation cephems (`Cep3`), anti-pseudomonas cephems

(CepAP), aminoglycosides (AG), macrolides (ML), tetracyclines (TC), carbapenems (CBP), and RFP/FOM (RFPFOM).

bacterium	sample	patients	max.	episodes	antibiotics
bac1	spl1	296	34	2668	RFPFOM(8), TC(1)
	spl5	319	19	27174	CBP(225), RFPFOM(148), ML(1), TC(1)
bac7	spl3	131	10	21957	Cep3(340), AG(323), CepAP(79), TC(16), Cep2(5), Cep1(3), Aug(2)
	spl5	301	21	76951	CBP(1334), CepAP(1158), Cep3(184), PcAP(128), TC(98), Aug(2), Cep1(2), Cep2(2)
bac8	spl1	81	7	8127	AG(1), CepAP(1), PcAP(1)
	spl5	296	21	37938	CepAP(40), AG(8), CBP(1)
bac11	spl1	208	22	16720	CBP(304), Cep2(81), Cep3(81), Cep1(1), ML(1), PcAP(1), PcB(1)

Fig. 5. The pages for the detected bacterium and the sample from bacterial culture data, the number of extracted sectorial episodes under $\sigma = 0.15$ and $\gamma = 0.8$, and the antibiotics, where Ant(n) denotes that n is the number of extracted sectorial episodes of the form $C \mapsto$ (Ant=R) such that (Ant=S) $\in C$.

Figure 5 means that different sectorial episodes representing drug-resistant change are extracted for every detected bacterium and sample. For (bac1,spl5), (bac7,spl5) and (bac11,spl1), the drug-resistant change for CBP occurs in the extracted episodes. In particular, for (bac7,spl5), the drug-resistant change CepAP also occurs in the extracted episodes. On the other hand, for (bac7,spl3), the drug-resistant change Cep3 and AG occurs in the extracted episodes.

5 Conclusion

In this paper, we have newly introduced the *sectorial episode* together with the *parallel episode* [5]. Then, we have designed the algorithm Sect to extract all of the sectorial episodes that are frequent and confidential. Finally, we have applied the algorithm Sect to bacterial culture data, and extracted sectorial episodes representing drug-resistant change.

Since the number of extracted sectorial episodes in Section 4 is large, it is a future work to introduce the concept of *closed* sectorial episodes like as closed sequential patterns [9, 10], in order to reduce the number of extracted episodes.

In Section 4, we have treated the pages for patients as one event sequence such that every page is far from at least the width of windows. Then, it is a future work to introduce another frequency measure like as the frequency of pages. Also it is a future work to apply our algorithm to another time-related data and extract sectorial episodes from them.

As stated in Section 1, since a sectorial episode has been captured as the direct precedent-subsequent relationship of events, it is just a candidate of *causality* in database (*cf.* [11]). Hence, it is an important future work to incorporate such causality with our sectorial episodes and to design the algorithm to extract sectorial episodes concerned with causality.

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. H. Mannila, H. Toivonen, A. I. Verkamo: *Discovery of frequent episodes in event sequences*, Data Mining and Knowledge Discovery **1**, 259–289, 1997.
6. 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.
7. R. Srikant, R. Agrawal: *Mining sequential patterns: Generalizations and performance improvements*, Proc. 5th EDBT, 3–17, 1996.
8. 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.
9. J. Wang, J. Han: *BIDE: Efficient mining of frequent closed sequences*, Proc. 20th ICDE, 2004.
10. X. Yan, J. Han, R. Afshar: *CloSpan: Mining closed sequential patterns in large datasets*, Proc. 3rd SDM, 2003.
11. C. Zhang, S. Zhang: *Association rule mining*, Springer-Verlag, 2002.