A Sequential Pattern Mining Algorithm using Rough Set Theory[*]

Ken Kaneiwa[1] and Yasuo Kudo[2]

[1] Department of Electrical Engineering and Computer Science, Iwate University
4-3-5 Ueda, Morioka, Iwate 020-8551, Japan
kaneiwa@cis.iwate-u.ac.jp
[2] College of Information and Systems, Muroran Institute of Technology
27-1 Mizumoto, Muroran 050-8585, Japan
kudo@csse.muroran-it.ac.jp

Abstract. Sequential pattern mining is a crucial but challenging task in many applications, e.g., analyzing the behaviors of data in transactions and discovering frequent patterns in time series data. This task becomes difficult when valuable patterns are locally or implicitly involved in noisy data. In this paper, we propose a method for mining such local patterns from sequences. Using rough set theory, we describe an algorithm for generating decision rules that take into account local patterns for arriving at a particular decision. To apply sequential data to rough set theory, the size of local patterns is specified, allowing a set of sequences to be transformed into a sequential information system. We use the discernibility of decision classes to establish evaluation criteria for the decision rules in the sequential information system.

1 Introduction

Data mining algorithms have been developed as tools to discover valuable patterns and rules from large amounts of data. In the traditional algorithms, association rules are discovered from attributes found frequently in datasets. By using a more complex approach, sequential pattern mining algorithms [1, 33, 2] enable us to find frequent patterns in sequential datasets. Sequential pattern mining requires the analysis of an ordered list of itemsets (e.g., a list of actions or orders) that can be modeled by a sequence. In order to effectively carry out the task, we have to extract only valuable patterns included in sequences by skipping noisy and meaningless patterns. However, frequent data mining algorithms are not feasible when it comes to extracting local (or implicit) patterns from noisy data. This is because the algorithms may not work when valuable patterns do not appear frequently or when waste patterns appear frequently. In fact, the frequencies of such valuable patterns may be less than a user-specified threshold, but setting a lower threshold leads to the recovery of a number of meaningless patterns.

---

[*] This paper is an extended version of [13].

In order to solve the problem, we have to logically and combinationally analyze patterns in sequences by checking the occurrences of local patterns that consistently result in a decision. For such an analysis, rule generation in rough set theory [10, 17, 16, 4, 14, 11, 12] provides a data mining algorithm based on the notions of attribute reduction and reduced decision rules. One of the advantages of rough set data mining is that it can generate reduced and consistent decision rules by logically checking all combinations of condition and decision attributes in an information system. Thus, rough set theory can be used to generate essential attributes through attribute reduction of logical combinations. However, sequential pattern mining algorithms have not been well studied in the context of rough set theory. Extending this approach to sequential pattern mining entails a logical analysis of local patterns in granular computing, which differs from the frequency analysis of sequential patterns.

In this paper, we propose a sequential pattern mining algorithm using the rule generation from discernibility in rough set theory. This algorithm computes subsequences of a fixed size that are regarded as local patterns hidden inside sequences. A sequential information system consists of the subsequences obtained from a set of sequences so that we can apply sequential data to the rough set data mining. The decision rules generated from a sequential information system are said to be *sequential* decision rules. In each of the rules, the condition attributes represent the occurrences of local patterns in a sequence. In order to estimate the local patterns in the rules, we establish the evaluation of occurrence-based accuracy and coverage for sequential decision rules. This is because the accuracy and coverage measures in rough set theory [18, 19, 26, 27, 20] do not evaluate the occurring sequence patterns in each sequence.

Our algorithm for mining local sequence patterns has the following interesting features.

- **Occurrences of Local Patterns:** Given a set of sequences, a sequential information system is constructed from the attributes that denote the subsequences of a fixed size, where each attribute value represents the number of occurrences of a local pattern in a sequence.
- **Granularities of Sequences:** The different sizes of local sequence patterns determine the diversity of granularities in a sequential information system. In other words, longer subsequences correspond to smaller granularities because they contain more information.
- **Reduced and Consistent Decision Rules:** In rough set theory, attribute reduction generates *reduced* decision rules. In addition, the decision rules are *consistent*, and hence, they are significantly different from the frequent association rules in traditional data mining, because logically inconsistent rules are excluded due to the discernibility of decision classes.

In relation to statistical data mining algorithms, these features are important in that they allow us to obtain implicitly local patterns, particularly when the patterns do not appear frequently. This is because each of the minimal subsets of the condition attributes calculated in rough set theory essentially discerns the decision classes among sequences without evaluating their frequencies.

This paper is arranged as follows. Section 2 briefly recalls the basic notions of rough sets. In Section 3, we describe the extension of rough set data mining to sequential pattern mining. We formalize a transformation from a set of sequences into a sequential information system. We then establish the occurrence-based accuracy and coverage of the sequential decision rules generated from the sequential information system. In Section 4, we present our algorithm for mining local sequence patterns from a set of sequences. The experimental results are reported in Section 5. Finally, we discuss related work in Section 6 and conclude this paper in Section 7.

2   Rough Sets

An attribute a is a mapping $a\colon U \to V_a$ where U is a non-empty finite set of objects (called the universe) and V_a is the value set of a. An information system is a pair $T = (U, A)$ of the universe U and a non-empty finite set A of attributes. Let B be a subset of A. The B-indiscernibility relation is defined by an equivalence relation I_B on U such that $I_B = \{(x, y) \in U^2 \mid \forall a \in B.a(x) = a(y)\}$. The equivalence class of I_B for each object x $(\in U)$ is denoted by $[x]_B$. Let X be a subset of U. We define the lower and upper approximations of X by $\underline{B}(X) = \{x \in U \mid [x]_B \subseteq X\}$ and $\overline{B}(X) = \{x \in U \mid [x]_B \cap X \neq \emptyset\}$. A subset B of A is a reduct of T if $I_B = I_A$ and there is no subset B' of B with $I_{B'} = I_A$ (i.e., B is a minimal subset of the condition attributes without losing discernibility).

A decision table is an information system $T' = (U, A \cup \{d\})$ such that each $a \in A$ is a condition attribute and $d \notin A$ is a decision attribute. Let V_d be the value set $\{d_1, \ldots, d_u\}$ of the decision attribute d. For each value $d_i \in V_d$, we obtain a decision class $U_i = \{x \in U \mid d(x) = d_i\}$ where $U = U_1 \cup \cdots \cup U_{|V_d|}$ and for every $x, y \in U_i$, $d(x) = d(y)$. The B-positive region of d is defined by $P_B(d) = \underline{B}(U_1) \cup \cdots \cup \underline{B}(U_{|V_d|})$. A subset B of A is a relative reduct of T' if $P_B(d) = P_A(d)$ and there is no subset B' of B with $P_{B'}(d) = P_A(d)$.

We define a formula $(a_1 = v_1) \wedge \cdots \wedge (a_n = v_n)$ in T' (denoting the condition of a rule) where $a_j \in A$ and $v_j \in V_{a_j}$ $(1 \leq j \leq n)$. The semantics of the formula in T' is defined by $[\![(a_1 = v_1) \wedge \cdots \wedge (a_n = v_n)]\!]_{T'} = \{x \in U \mid a_1(x) = v_1, \ldots, a_n(x) = v_n\}$. Let φ be a formula $(a_1 = v_1) \wedge \cdots \wedge (a_n = v_n)$ in T'. A decision rule for T' is of the form $\varphi \to (d = d_i)$, and it is true if $[\![\varphi]\!]_{T'} \subseteq [\![(d = d_i)]\!]_{T'} (= U_i)$. The accuracy and coverage of a decision rule r of the form $\varphi \to (d = d_i)$ are respectively defined as follows.

$$accuracy(T', r, U_i) = \frac{|U_i \cap [\![\varphi]\!]_{T'}|}{|[\![\varphi]\!]_{T'}|}$$

$$coverage(T', r, U_i) = \frac{|U_i \cap [\![\varphi]\!]_{T'}|}{|U_i|}$$

In the evaluations, $|U_i|$ is the number of objects in a decision class U_i and $|[\![\varphi]\!]_{T'}|$ is the number of objects in the universe $U = U_1 \cup \cdots \cup U_{|V_d|}$ that satisfy condition φ of rule r. Therefore, $|U_i \cap [\![\varphi]\!]_{T'}|$ is the number of objects satisfying the condition φ restricted to a decision class U_i.

3 Sequential Data in Rough Sets

In this section, using rough set theory, we present a new method for expressing the local features of sequences in an information system.

3.1 Sequential Information Systems

An itemset a_i is a non-empty set of items, and the size of a_i is the cardinality of a_i, i.e., $|a_i|$. A sequence s is an ordered list of itemsets $\langle a_1, a_2, \ldots, a_n \rangle$, simply denoted by $a_1 a_2 \cdots a_n$. The size of s (denoted $||s||$) is the number of elements of the list $a_1 a_2 \cdots a_n$, and the length of s is the total number of the sizes $|a_1|$, $|a_2|$, ..., $|a_n|$. A sequence $s_1 = a_1 a_2 \cdots a_n$ is a subsequence of another sequence $s_2 = b_1 b_2 \cdots b_m$ (denoted $s_1 \sqsubseteq s_2$), if there are integers $i_1 < i_2 < \cdots < i_n$ such that $a_1 \subseteq b_{i_1}$, $a_2 \subseteq b_{i_2}$, ..., $a_n \subseteq b_{i_n}$. The empty sequence ϵ is a subsequence of any sequence. A sequence s_1 is a strict subsequence of another sequence s_2 (denoted $s_1 \sqsubseteq^{st} s_2$) if there exists an integer i such that $a_1 \subseteq b_i$, $a_2 \subseteq b_{i+1}$, ..., $a_n \subseteq b_{i+n-1}$.

As a practical example, an ordered list of itemsets can be used to represent a list of sequential actions of an agent where each itemset corresponds to an action, which consists of a set of operations corresponding to items. Let us consider the following four sequences:

$$s_1 = aabcac$$
$$s_2 = bcca$$
$$s_3 = cba$$
$$s_4 = aabca$$

where $a = \{i_1, i_2\}$, $b = \{i_2, i_3, i_4\}$, and $c = \{i_2, i_3\}$ are itemsets and i_1, i_2, i_3, and i_4 are items. The sequence s_1 is the series $aabcac$ of actions of an agent and the sequence s_2 is the series $bcca$ of actions of another agent. In addition, the sequences s_3 and s_4 are the series cba and $aabca$ of actions, respectively, of two other agents.

In order to apply this sequential data to rough set theory, we characterize the local patterns of sequences in an information system that can be used to discern the sequences. In our representation of knowledge, the occurrences of subsequences in each sequence are calculated to express the local features of a set of sequences by using an information system.

Definition 1 (Sequential Information System). *Let $U_{sq} = \{s_1, \ldots, s_n\}$ be a set of sequences and A_{sq} be a set of subsequences of sequences $s_1, \ldots, s_n$ in U_{sq}. A sequential information system is an information system $T = (U_{sq}, A_{sq})$ where for each attribute $a \in A_{sq}$ (named by a subsequence), $a(x)$ maps the number of occurrences of the subsequence a in each sequence $x \in U_{sq}$.*

We denote the concatenation $\overbrace{s \cdots s}^{n}$ of sequence s by s^n (in particular, s^0 denotes the empty sequence ϵ). We can precisely define the number n of occurrences

of subsequence s_1 in sequence s_2 as follows:

$$\Omega_{s_1}(s_2) = n$$

if the concatenation s_1^n is a subsequence of s_2 but the concatenation s_1^{n+1} is not a subsequence of s_2. For example, $\Omega_{ac}(caac) = 1$ and $\Omega_{ac}(abcbacc) = 2$, i.e., ac appears once in the sequence $caac$ and twice in the sequence $abcbacc$.

Definition 2 (Sequential Decision Table). *A sequential decision table is a decision table $T' = (U_{sq}, A_{sq} \cup \{d\})$ such that $T = (U_{sq}, A_{sq})$ is a sequential information system and $d \notin A_{sq}$ is a decision attribute.*

In rough set theory, the decision attribute d classifies the objects of U_{sq} (i.e., sequences) in the decision table. In other words, we can identify the decision classes $U_1, \ldots, U_{|V_d|}$ that divide the sequences of $U_{sq} = U_1 \cup \cdots \cup U_{|V_d|}$ by the decision attribute.

3.2 Granularities of Sequences

The local size of valuable patterns varies depending on the property of sequential data in many application domains. To deal with the diversity of sequential data, we consider the different sizes of subsequences in a sequential information system that set granularities for the features of sequences in rough set theory. As a result of this method, the size k subsequences of a sequence have a smaller granularity than the size $k - 1$ subsequences of that.

In order to capture local patterns from a sequence s, we define the set of subsequences of a size occurring in the sequence s as follows.

Definition 3 (Size k Subsequences). *The set of size k subsequences of s is defined by*

$$Sub_k(s) = \{s' \mid s' \sqsubseteq s \ \& \ ||s'|| = k\}.$$

For sequences s_1, s_2, s_3, and s_4 shown in Section 3.1, we obtain the following sets of size 2 subsequences:

$$\begin{aligned}
Sub_2(s_1) &= \{aa, ab, ac, ba, bc, ca, cc\} \\
Sub_2(s_2) &= \{ba, bc, ca, cc\} \\
Sub_2(s_3) &= \{ba, ca, cb, cc\} \\
Sub_2(s_4) &= \{aa, ab, ac, ba, bc, ca, cc\}
\end{aligned}$$

In $Sub_2(s_1)$ with $s_1 = aabcac$, subsequence aa consists of the first and second itemsets in s_1; subsequence ab consists of the second and third itemsets in s_1. In a non-trivial case, subsequence cc occurs in all the sequences s_1, s_2, s_3, and s_4. The subsequence cc obviously occurs in s_1 and s_2 and it is subsumed by cb in s_3 and bc in s_4, respectively, because $c \subseteq b$ with $b = \{i_2, i_3, i_4\}$ and $c = \{i_2, i_3\}$. In this example, we can intuitively interpret the size 2 subsequences as changes from one action to another when the sequences describe agents' actions. Therefore,

the size 2 subsequence sets $Sub_2(s_1), \ldots, Sub_2(s_4)$ indicate the local changes in actions of the four agents.

Furthermore, local patterns in a sequential information system are analyzed more strictly as follows. By limiting the definition of subsequences, we obtain the set of strict subsequences occurring in the sequence s.

Definition 4 (Size k Strict Subsequences). *The set of strict size k subsequences of s is defined by*

$$Sub_k^{st}(s) = \{s' \mid s' \sqsubseteq^{st} s \ \& \ ||s'|| = k\}.$$

For example, we have the following sets of strict size 2 subsequences in the sequences s_1, s_2, s_3, and s_4 (shown in Section 3.1):

$$Sub_2^{st}(s_1) = \{aa, ab, ac, bc, ca, cc\}$$
$$Sub_2^{st}(s_2) = \{bc, ca, cc\}$$
$$Sub_2^{st}(s_3) = \{ba, cb, ca, cc\}$$
$$Sub_2^{st}(s_4) = \{aa, ab, ac, bc, ca, cc\}$$

In $Sub_2^{st}(s_1)$ with $s_1 = aabcac$, the local pattern ba in $Sub_2(s_1)$ is not a strict subsequence of s_1, but it is nevertheless a subsequence of s_1. This is because there is an itemset c between b and a (i.e., bca) in the sequence s_1. That is, we can use Sub_k to generate lazy local patterns by skipping itemset c in sequence bca. Similar to the case of size 2 subsequences, strict size 2 subsequence cc occurs in all the sequences s_1, s_2, s_3, and s_4, i.e., cc is a strict subsequence of both s_1 and s_2 and it is subsumed by cb in s_3 and bc in s_4.

Another granularity can be analyzed by extracting size 3 subsequences from the sequences s_1, s_2, s_3, and s_4. Intuitively, in the analysis of actions, the size 3 subsequences imply more complex combinations of action changes than the size 2 subsequences. Similar to the above example, the sets of size 3 subsequences are captured from the sequences s_1, s_2, s_3, and s_4 as follows.

$$Sub_3(s_1) = \{aaa, aab, aac, aba, abc, aca, acc, bac, bca,$$
$$bcc, cac, cca, ccc\}$$
$$Sub_3(s_2) = \{bca, bcc, cca, ccc\}$$
$$Sub_3(s_3) = \{cba, cca\}$$
$$Sub_3(s_4) = \{aaa, aab, aac, aba, abc, aca, acc, bca, cca\}$$

The combinations of itemsets occurring in the size 3 subsequences are more complex (e.g., $Sub_3(s_1)$ contains 12 local patterns) but those in the strict size 3 subsequences are not very complex, as can be seen in the following:

$$Sub_3^{st}(s_1) = \{aab, aac, abc, acc, bca, cac, cca\}$$
$$Sub_3^{st}(s_2) = \{bcc, cca, ccc\}$$
$$Sub_3^{st}(s_3) = \{cba, cca\}$$
$$Sub_3^{st}(s_4) = \{aab, aac, abc, acc, bca, cca\}$$

	aa	ab	ac	ba	bc	ca	cb	cc	d
s_1	1	1	2	1	1	1	0	1	1
s_2	0	0	0	1	1	1	0	1	0
s_3	0	0	0	1	0	1	1	1	0
s_4	1	1	1	1	1	1	0	1	1

Table 1. Size 2 sequential decision table T_1'

Let S be a set of sequences. We denote $Sub_k(S) = \bigcup_{s \in S} Sub_k(s)$ (resp. $Sub_k^{st}(S) = \bigcup_{s \in S} Sub_k^{st}(s)$).

3.3 Transformation from Sequences into an Information System

We define a transformation from a finite set of sequences into a sequential information system with respect to size k subsequences as follows.

Definition 5 (Transformation). *Let $k > 0$ be a non-negative integer, and let $S = \{s_1, \ldots, s_j\}$ be a finite set of sequences. The size k sequential information system is defined as a sequential information system $T = (U_{sq}, A_{sq})$ such that*

$$U_{sq} = S \text{ and } A_{sq} = Sub_k(S).$$

In addition, if A_{sq} is defined by $Sub_k^{st}(S)$, then T is the strict size k sequential information system.

After a finite set of sequences is transformed into a size k sequential information system $T = (U_{sq}, A_{sq})$, the information system is extended to a size k sequential decision table $T' = (U_{sq}, A_{sq} \cup \{d\})$ by adding decision attribute d. For each sequential decision table, a decision attribute for sequences has to be designed on the basis of domain knowledge, i.e., knowledge of human experts. For example, a domain expert can set a decision attribute for some sequences of actions or operations in computers, applications, and networks. In a specific domain, the sequences s_1 and s_4 result in a success (denoted value 1), but the sequences s_2 and s_3 cause a failure (denoted value 0). This setting is modeled by supplementing the decision attribute d to the information system $T = (U_{sq}, A_{sq})$ with $d(s_1) = d(s_4) = 1$ and $d(s_2) = d(s_3) = 0$. In Table 1, we show a sequential decision table that is obtained from the transformation from the sequences s_1, s_2, s_3, and s_4 into the size 2 sequential information system T_1, and the decision attribute d. In the table, the attributes are labeled by the size 2 subsequences

$$aa, \ ab, \ ac, \ ba, \ bc, \ ca, \ cb, \text{ and } cc$$

in $Sub_2(s_1) \cup Sub_2(s_2) \cup Sub_2(s_3) \cup Sub_2(s_4)$. For example, $\Omega_{aa}(s_1) = 1$ and $\Omega_{ac}(s_1) = 2$ indicate that the local patterns aa and ac occur in s_1 once and twice, respectively, and $\Omega_{cb}(s_1) = 0$ indicates that the pattern cb does not occur in s_1.

	aa	ab	ac	ba	bc	ca	cb	cc	d
s_1	1	1	1	0	1	1	0	1	1
s_2	0	0	0	0	1	1	0	1	0
s_3	0	0	0	1	0	1	1	1	0
s_4	1	1	1	0	1	1	0	1	1

Table 2. Strict size 2 sequential decision table T_2'

	aaa	aab	aac	aba	abc	aca	acc	bac	bca	bcc	cac	cba	cca	ccc	d
s_1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
s_2	0	0	0	0	0	0	0	0	1	1	0	0	1	1	0
s_3	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0
s_4	1	1	1	1	1	1	1	0	1	0	0	0	1	0	1

Table 3. Size 3 sequential decision table T_3'

Moreover, Table 2 shows a sequential decision table of a strict size 2 sequential information system transformed from the sequences s_1, s_2, s_3, and s_4 along with the decision attribute d. Notice that the size 2 subsequences in Table 1 contain some discontinuous ordered patterns but the strict size 2 subsequences in Table 2 do not include them. For example, $\Omega_{ba}(s_1) = 0$ means that the strict pattern ba does not occur in sequence s_1, but the lazy pattern ba does occur in the sequence.

From the sets of subsequences in $Sub_3(s_1)$, $Sub_3(s_2)$, $Sub_3(s_3)$, and $Sub_3(s_4)$, the size 3 sequential and strict size 3 sequential information systems T_1 and T_2 in Tables 3 and 4, respectively, are transformed from the sequences s_1, s_2, s_3, and s_4. Consequently, the number of subsequences increases in comparison with the size 2 sequential information systems.

3.4 Accuracy and Coverage

Using the transformation discussed in Section 3.3, we can obtain a size k sequential information system $T_k = (U_{sq}, A_{sq})$ from a set of sequences. The sequential decision table $T_k' = (U_{sq}, A_{sq} \cup \{d\})$ is constructed by adding a decision attribute d for the sequences in U_{sq} to the information system T_k. This decision table is used to generate decision rules for T_k' of the form:

$$(a_1 = n_1) \wedge \cdots \wedge (a_n = n_n) \Rightarrow (d = v)$$

where each a_i denotes a subsequence and each n_i expresses the number of occurrences of subsequence a_i by a non-negative integer. Let T' be a sequential decision table. A decision rule for T' can be called a sequential decision rule if there is an attribute condition $a_i = n_i$ in the rule such that $n_i \neq 0$.

Here, we discuss the interpretation of such a sequential decision rule. From the sequential decision table $T' = (U_{sq}, A_{sq} \cup \{d\})$, we can generate sequential

	aab	aac	abc	acc	bca	bcc	cac	cba	cca	ccc	d
s_1	1	1	1	1	1	0	1	0	1	0	1
s_2	0	0	0	0	0	1	0	0	1	1	0
s_3	0	0	0	0	0	0	0	1	1	0	0
s_4	1	1	1	1	1	0	0	0	1	0	1

Table 4. Strict size 3 sequential decision table T'_4

decision rules as follows:

$$(cca = 1) \wedge (acc = 1) \Rightarrow (d = 1)$$

This rule implies that if a sequence contains the local patterns cca and acc, then it results in $d = 1$. However, the following decision rule is not valuable for our purpose.

$$(cba = 0) \wedge (bcc = 0) \Rightarrow (d = 1)$$

This is because the condition attributes indicate that no occurrence of the local patterns cba and bcc in a sequence results in the derivation of the decision attribute $d = 1$. In order to analyze agents' behaviors, some patterns that actually occur have to be mined from the sequential data. However, we do not exclude decision rules if they indicate the occurrence and non-occurrence of local patterns, as shown below:

$$(bca = 1) \wedge (ccc = 0) \Rightarrow (d = 1)$$

This rule means that the occurrence of local pattern bca results in the decision attribute $d = 1$ as long as the local pattern ccc does not appear.

We define an evaluation function for sequential decision rules that determines whether each size k sequential information system is well represented when it comes to classifying the decision class. To measure varieties of local sequence patterns for each sequence, we calculate the sum of numbers of the occurring patterns as follows.

Definition 6 (Sum of Occurring Local Patterns). *Let S be a set of sequences and let $A' \subseteq A_{sq}$. The sum of numbers of occurring local patterns $o(s, A')$ in each sequence $s \in S$ is defined by*

$$o(s, A') = \sum_{a \in A'} sign(a(s))$$

where the sign function $sign(n)$ is defined by $sign(n) = 1$ if $n > 0$ and $sign(n) = 0$ if $n = 0$.

We extend the function $o(s, A')$ to a set of sequences S by defining $o(S, A') = \sum_{s \in S} o(s, A')$.

Definition 7 (Occurrence-Based Accuracy and Coverage). *Let S_i be a decision class in S, let $s \in S_i$, and let r be a sequential decision rule for*

a sequential decision table T'. The occurrence-based accuracy o_accuracy and occurrence-based coverage o_coverage of sequential decision rule r of the form $\varphi \rightarrow (d = d(s))$ with $d(s) \in V_d$ are defined as follows.

$$o\_accuracy(T', r, S_i) = \frac{o(S_i \cap [\![\varphi]\!]_{T'}, A_\varphi)}{o([\![\varphi]\!]_{T'}, A_\varphi)}$$

$$o\_coverage(T', r, S_i) = \frac{o(S_i \cap [\![\varphi]\!]_{T'}, A_\varphi)}{o(S_i, A_\varphi)}$$

where $A_\varphi = \{a \in A \mid a = v \text{ occurs in } \varphi\}$.

The occurrence-based accuracy and coverage are measured by using the function $o(s, A')$ in order to delete meaningless decision rules for sequential pattern mining.

We can define another measurement of the occurrence-based coverage by replacing $o(S_i, A_\varphi)$ with $|S_i| \cdot |A_\varphi|$.

Definition 8 (Variant of Occurrence-Based Coverage). *Let S_i be a decision class in S, let $s \in S_i$, and let r be a sequential decision rule for a sequential decision table T'. A variant vo_coverage of the occurrence-based coverage of sequential decision rule r of the form $\varphi \rightarrow (d = d(s))$ with $d(s) \in V_d$ is defined as follows.*

$$vo\_coverage(T', r, S_i) = \frac{o(S_i \cap [\![\varphi]\!]_{T'}, A_\varphi)}{|S_i| \cdot |A_\varphi|}$$

where $A_\varphi = \{a \in A \mid a = v \text{ occurs in } \varphi\}$.

The occurrence-based coverage $o\_coverage(T', r, S_i)$ implies the coverage of rule r in $o(S_i, A_\varphi)$ where $o(S_i, A_\varphi)$ is the number of occurred subsequences (i.e., the number of condition attributes with positive values) in all objects in decision class S_i. In contrast, the variant $vo\_coverage(T', r, S_i)$ implies the occurrence-based coverage of rule r in the product of $|S_i|$ and $|A_\varphi|$ where $|S_i|$ is the number of all objects in decision class S_i and $|A_\varphi|$ is the number of occurred and non-occurred subsequences (i.e., the number of condition attributes) in φ.

4   Sequential Pattern Mining Algorithm

This section describes a sequential pattern mining algorithm $sq\_mining(S, m, d, b)$ for a set of sequences S, a maximum subsequence size m, a decision attribute d, and a bool value b. The maximum subsequence size m is given by a user-specified maximum-size of local patterns. The decision attribute d is defined as a function $d\colon S \rightarrow V_d$ where for every sequence $s \in S$, a decision value is set from $d(s) \in V_d$. The decision attribute can be obtained from several knowledge representations, e.g., the identities of agents, positive and negative values for sequences, etc.

In Fig.1, we show a sequential data mining algorithm that returns a list of sets of (sequential) decision rules $R_2, \ldots, R_m$ (from size 2 to m), such that

---

Algorithm *sq_mining*

---

input: set of sequences $S = \{s_1, \ldots, s_n\}$,
         maximum subsequence size m,
         decision attribute d, bool b
output: list of sets of decision rules $(R_2, \ldots, R_m)$
1:   **begin**
2:       **for** $k = 2$ **to** m **do**
3:          $R_k = \emptyset$;
4:          $A_k = subsq(s_1, k, b) \cup \cdots \cup subsq(s_n, k, b)$;
5:          **for** $s \in S$ and $a \in A_k$ **do**
6:             $a(s) = subsq\_count(s, a, b)$
7:          **rof**
8:          $T'_k = (S, A_k \cup \{d\})$;
9:          $\mathcal{R} = reducts(T'_k)$;
10:         **for** $B \in \mathcal{R}$ **do**
11:            **for** $i = 1$ **to** $|V_d|$ **do**
12:              **for** $s \in S_i$ **do**
13:                $R_k = R_k \cup \{rule(s, B, T'_k)\}$;
14:              **rof**
15:            **rof**
16:         **rof**
17:       **rof**
18:       **return** $(R_2, \ldots, R_m)$;
19: **end**;

Fig. 1. Sequential pattern mining algorithm.

the condition attributes in each rule indicate the occurrences of subsequences. Our algorithm can generate *basic* decision rules for all the reducts in a decision table as described in [4]. That is, the condition attributes in each rule are given by a minimal subset of the condition attributes without losing discernibility in the decision table. From the rules, *consistent* decision rules can be selected by deciding whether the occurrence-based accuracy is 1.0 or not. This algorithm is outlined as follows.

1. **Transformation:** For each size k from 2 to m, a set of sequences is transformed into size k (resp. strict size k) sequential information systems if $b = 0$ (resp. $b = 1$) by calling the following subroutines.
   (a) **Subsequence generation:** The set of size k subsequences $Sub(S)$ or strict subsequences $Sub^{st}(S)$ is generated by checking all the partial patterns of given sequences. These subsequences are used to represent attribute names in the sequential information system.
   (b) **Subsequence counting:** The occurrences of subsequences are exhaustively counted to set the values of attributes in the sequential information system.
2. **Rule generation:** By using a rough set rule generation method, reduced decision rules are generated from the sequential decision table where con-

dition attributes are represented by the occurrences of subsequences of size k.

4.1 Transformation

In lines 2 - 17 of the mining algorithm sq_mining, for each size k from 2 to m, the set of size k subsequences $Sub(S)$ or strict size k subsequences $Sub^{st}(S)$ is extracted from sequences in order to construct the size k or the strict size k sequential information system. In line 4, all the subsequences of size k in S are generated as attribute names, which are stored in variable $A_k = subsq(s_1, k, b) \cup \cdots \cup subsq(s_n, k, b)$.

As shown in Fig.2, the subsequence generation algorithm $subsq(s, k, b)$ for sequence s, subsequence size k, and bool value b. This algorithm computes $Sub_k(s)$ if $b = 0$ and $Sub_k^{st}(s)$ if $b = 1$. In $subsq(s, k, b)$, we use some operations for sequences. Let $s = a_1 a_2 \cdots a_n$ be a sequence. Then, $start(s)$ and $other(s)$ return the first itemset a_1 and the sequence of the other itemsets $a_2 \cdots a_n$. Let s_1 and s_2 be two sequences. Then, $concat(s_1, s_2)$ is the concatenation of s_1 and s_2, i.e., $concat(s_1, s_2) = s_1 s_2$. In lines 11 - 13 of algorithm $subsq(s, k, b)$, for every subset x of the first itemset $start(s)$, this algorithm is recursively called in order to generate the set of subsequences $subsq(concat(x, other(s)))$. This is because the subsequences of s contain subsets x of the itemsets of s, i.e., the sequence ab is a subsequence of the sequence ac if $b \subseteq c$ where a, b, and c are itemsets.

After generating the subsequences, in lines 5 - 7 of the mining algorithm, it calculates the numbers of occurrences $a(s)$ of local patterns denoted by the attributes a in A_k and the sequences s in S, which become their attribute values in a sequential decision table $T_k' = (S, A_k \cup \{d\})$ (in line 8). As a subroutine, the subsequence counting algorithm $subsq\_count(s_1, s_2, b)$ shown in Fig.3 counts the number of occurrences of subsequence pattern s_2 in sequence s_1 when two sequences s_1 and s_2 and a bool value b are used as input.

4.2 Rule Generation

In line 9, the set $\mathcal{R} = reducts(T_k')$ [16] of all the relative reducts of size k sequential decision table $T_k' = (U_{sq}, A_{sq} \cup \{d\})$ is computed by the standard reduct set computation in [4]. Each $B \in \mathcal{R}$ is a minimal subset of the condition attributes that are the attributes $a_1, \ldots, a_l$ expressed by subsequences. This means that the subsequences denoted by $a_1, \ldots, a_l$ are essential to discern the decision classes $S_1, \ldots, S_{|V_d|}$. In lines 10 - 16, the reduced decision rules generated by $rule(s, B, T_k')$ are added to the set R_k of decision rules for size k for each relative reduct $B \in \mathcal{R}$ where i is a natural number from 1 to $|V_d|$ and S_i is a decision class of S. That is, the relative reduct B supplies a minimal subset of the condition attributes of sequential decision table T_k'.

4.3 Computation

The complexity of the sequential pattern mining algorithm with rules generation is exponential time in the worst case. This computational property is caused

Algorithm *subsq*

---

input: sequence s, subsequence size k, bool b
output: a set of sequences S
1: **begin**
2:     $\Delta = \emptyset$;
3:     **if** $size(s) < k$ **then return** $\emptyset$
4:     **else if** $k = 0$ **then return** $\{\epsilon\}$
5:     **else if** $b = 0$ **then**
6:         $\Delta = \{concat(start(s), s') \mid s' \in subsq(other(s), k - 1, 0)\}$
7:           $\cup subsq(other(s), k, 0)$;
8:     **else if** $b = 1$ **then**
9:         $\Delta = \{concat(start(s), s') \mid s' \in subsq(other(s) \uparrow k - 1, k - 1, 1)\}$
10:        $\cup subsq(other(s), k, 1)$
11:     **for** $x \subseteq start(s)$ **do**
12:         $\Delta = \Delta \cup subsq(concat(x, other(s)), k, b)$;
13:     **rof**
14:     **return** Δ;
15: **end**;

Fig. 2. Subsequence generation algorithm.

from the standard reduct set computation $reducts(T'_k)$ and the subsequence generation $subsq(s, k, b)$ (in addition, the complexity of $subsq\_count$ depends on $subsq$). In the following, we show that the complexity of the subsequence generation $subsq(s, k, b)$ is reduced if every itemset is independent from each other. [3]

Proposition 1. *Let s be a sequence, n be the size of s, and k be a subsequence size. If every itemset is independent from each other itemset in s, the following time complexity holds:*

1. *The algorithm $subsq(s, k, 0)$ computes the set of size k subsequences of s in $2^{n+1} - 1$ steps.*
2. *The algorithm $subsq(s, k, 1)$ computes the set of strict size k subsequences of s in n^2 steps.*

To compute the set of size k subsequences of a sequence s, the algorithm $subsq(s, k, 0)$ recursively calls $subsq(other(s), k - 1, 0)$ and $subsq(other(s), k, 0)$ (in Lines 6 and 7) but does not $subsq(concat(x, other(s)), k, 0)$ (in Line 12). We can construct a binary tree of s such that the root node is labeled with $subsq(s, k, 0)$ and each non-leaf node labeled with $subsq(s_i, k, 0)$ has two children labeled with two recursive calls $subsq(other(s_i), k-1, 0)$ and $subsq(other(s_i), k, 0)$. So, the height of a binary tree of s is limited to the size n of s because $other(s_i)$ in both recursive calls makes the sizes of inputed sequences decrease in each call. In the worst case, the number of all the nodes of a binary tree of s is bounded by

---

[3] An itemset a is independent from another itemset b if $a \not\subseteq b$ and $b \not\subseteq a$.

Algorithm *subsq_count*

---

input: sequence s_1, sequence s_2, bool b
output: number of subsequences ct
1: **begin**
2:      $\pi = s_2$; $ct = 0$;
3:      **while** $\pi \in subsq(s_1, |\pi|, b)$ **do**
4:          $\pi = concat(\pi, s_2)$;
5:          $ct = ct + 1$;
6:      **elihw**
7:      **return** ct;
8: **end**;

Fig. 3. Subsequence counting algorithm.

$2^{n+1} - 1$. Moreover, the complexity of the the algorithm $subsq(s, k, 1)$ is reduced into polynomial time if the set of strict size k subsequences of s is computed. It recursively calls $subsq(other(s) \uparrow k-1, k-1, 1)$ and $subsq(other(s), k, 0)$ (in Lines 9 and 10) but does not $subsq(concat(x, other(s)), k, 1)$ (in Line 12) where $other(s) \uparrow k-1$ is the sequence of the first $k-1$ itemsets of $other(s)$. As a result, the number of recursive calls is limited to $n \times k$. So, we have $n \cdot k \leq n^2$ since $k \leq n$.

The complexity of the subsequence generation can be reduced to polynomial time if every itemset is independent from each other and the set of strict size k subsequences is generated. Unfortunately, our reduct set computation is still not optimized. In order to reduce the total complexity of the sequential pattern mining algorithm, we will have to combine the above restriction with an optimized algorithm for the reduct set computation in a future work.

5 Experimental Results

We implemented the sequential pattern mining algorithm sq_mining in Java. In order to evaluate this mining algorithm, we discuss the sequential decision rules that were generated from the four sequences s_1, s_2, s_3, and s_4 in Section 3.1. Consider the set of sequences $S = \{s_1, s_2, s_3, s_4\}$, the maximum subsequence size $m = 3$, and the decision attributes $d(s_1) = d(s_4) = 1$ and $d(s_2) = d(s_3) = 0$. First, the sequential pattern mining algorithm $sq\_mining(S, k, d)$ constructs the sequential decision tables T_1', T_2', T_3', and T_4' in Tables 1, 2, 3, and 4 from the sequences s_1, s_2, s_3, and s_4 in S. Second, it generates the sequential decision rules from the sequential information systems. All the computations are performed in 3.572 s on two Intel CPUs of 2.66 GHz, 4G memory, running Windows Vista.

Tables 5 and 6 show the sequential decision rules for size 2 and strict size 2 subsequences and the occurrence-based coverage of the rules, respectively. From the outcomes of applying the algorithm, we obtain the four and three sequential decision rules with the decision $d = 1$ for size 2 and strict size 2 subsequences, respectively, but do not find any sequential decision rule with decision $d = 0$.

Table 5. Occurrence-based coverage of sequential decision rules for size 2

decision rules for size 2	o_coverage	vo_coverage
1: $(aa = 1) \Rightarrow (d = 1)$	1.0	1.0
2: $(ab = 1) \Rightarrow (d = 1)$	1.0	1.0
3: $(ac = 2) \Rightarrow (d = 1)$	0.5	0.5
4: $(ac = 1) \Rightarrow (d = 1)$	0.5	0.5

Table 6. Occurrence-based coverage of sequential decision rules for strict size 2

decision rules for strict size 2	o_coverage	vo_coverage
1: $(ac = 1) \Rightarrow (d = 1)$	1.0	1.0
2: $(aa = 1) \Rightarrow (d = 1)$	1.0	1.0
3: $(ab = 1) \Rightarrow (d = 1)$	1.0	1.0

We notice that the two rules $(aa = 1) \Rightarrow (d = 1)$ and $(ab = 1) \Rightarrow (d = 1)$ with the occurrence-based coverage $o\_coverage = 1.0$ (and $vo\_coverage = 1.0$) are valuable because they are common rules for both size 2 and strict size 2 subsequences. Hence, the two rules can be combined as having the local patterns to characterize the decision $d = 1$ as follows:

$$(aa = 1) \vee (ab = 1) \Rightarrow (d = 1)$$

where aa and ab are size 2 or strict size 2 subsequences. This combined rule enables us to predict whether a newly input sequence leads to the decision $d = 0$ or $d = 1$. If the rule does not hold for the new sequence, then we can further use the sequential decision rule for strict size 2 subsequences with $o\_coverage = 1.0$ (and $vo\_coverage = 1.0$) (in Table 6):

$$(ac = 1) \Rightarrow (d = 1)$$

where ac is a strict size 2 subsequence.

Tables 7 and 8 list the sequential decision rules for size 3 and strict size 3 subsequences and the occurrence-based coverages of the rules, respectively. Our mining algorithm generates 31 and 9 sequential decision rules with the decision $d = 1$ or $d = 0$ for size 2 and strict size 2 subsequences, respectively. It should be said that the six and four sequential decision rules (Nos.1 - 5, 7 in Table 7 and Nos.1 - 3, 5 in Table 8) are specific cases of the two rules $(aa = 1) \Rightarrow (d = 1)$ and $(ab = 1) \Rightarrow (d = 1)$ for size 2 and strict size 2 subsequences when selecting the rules with $o\_coverage = 1.0$ (and $vo\_coverage = 1.0$). This means that these 10 rules can be represented by the combined rule $(aa = 1) \vee (ab = 1) \Rightarrow (d = 1)$ where aa and ab are size 2 or strict size 2 subsequences. As another decision rule for size 3 and strict size 3 subsequences, the following rule has $o\_coverage = 1.0$ (and $vo\_coverage = 1.0$):

$$(acc = 1) \Rightarrow (d = 1)$$

where acc is a size 3 or strict size 3 subsequence. In addition, we obtain the following decision rule with $o\_coverage = 1.0$ (and $vo\_coverage = 1.0$), but this

Table 7. Occurrence-based coverage of sequential decision rules for size 3

decision rules for size 3	o_coverage	vo_coverage
1: $(aba = 1) \Rightarrow (d = 1)$	1.0	1.0
2: $(aab = 1) \Rightarrow (d = 1)$	1.0	1.0
3: $(aac = 1) \Rightarrow (d = 1)$	1.0	1.0
4: $(abc = 1) \Rightarrow (d = 1)$	1.0	1.0
5: $(aaa = 1) \Rightarrow (d = 1)$	1.0	1.0
6: $(acc = 1) \Rightarrow (d = 1)$	1.0	1.0
7: $(aca = 1) \Rightarrow (d = 1)$	1.0	1.0
8: $(cba = 0) \wedge (ccc = 1) \wedge (bac = 1) \Rightarrow (d = 1)$	1.0	0.333
9: $(cac = 1) \wedge (cba = 0) \wedge (bcc = 1) \Rightarrow (d = 1)$	1.0	0.333
10: $(bcc = 1) \wedge (bac = 1) \wedge (bca = 1) \Rightarrow (d = 1)$	0.75	0.5
11: $(cac = 1) \wedge (cba = 0) \wedge (ccc = 1) \Rightarrow (d = 1)$	1.0	0.333
12: $(cac = 1) \wedge (bcc = 1) \wedge (bca = 1) \Rightarrow (d = 1)$	0.75	0.5
13: $(ccc = 1) \wedge (bac = 1) \wedge (bca = 1) \Rightarrow (d = 1)$	0.75	0.5
14: $(cac = 1) \wedge (ccc = 1) \wedge (bca = 1) \Rightarrow (d = 1)$	0.75	0.5
15: $(cba = 0) \wedge (bcc = 1) \wedge (bac = 1) \Rightarrow (d = 1)$	1.0	0.333
16: $(bcc = 0) \wedge (bac = 0) \wedge (bca = 1) \Rightarrow (d = 1)$	0.25	0.166
17: $(cac = 0) \wedge (bcc = 0) \wedge (bca = 1) \Rightarrow (d = 1)$	0.25	0.166
18: $(cac = 0) \wedge (ccc = 0) \wedge (bca = 1) \Rightarrow (d = 1)$	0.25	0.166
19: $(ccc = 0) \wedge (bac = 0) \wedge (bca = 1) \Rightarrow (d = 1)$	0.25	0.166
20: $(cba = 0) \wedge (ccc = 1) \wedge (bac = 0) \Rightarrow (d = 0)$	0.5	0.166
21: $(cac = 0) \wedge (cba = 0) \wedge (bcc = 1) \Rightarrow (d = 0)$	0.5	0.166
22: $(bcc = 1) \wedge (bac = 0) \wedge (bca = 1) \Rightarrow (d = 0)$	1.0	0.333
23: $(cac = 0) \wedge (cba = 0) \wedge (ccc = 1) \Rightarrow (d = 0)$	0.5	0.166
24: $(cac = 0) \wedge (bcc = 1) \wedge (bca = 1) \Rightarrow (d = 0)$	1.0	0.333
25: $(ccc = 1) \wedge (bac = 0) \wedge (bca = 1) \Rightarrow (d = 0)$	1.0	0.333
26: $(cac = 0) \wedge (ccc = 1) \wedge (bca = 1) \Rightarrow (d = 0)$	1.0	0.333
27: $(cba = 0) \wedge (bcc = 1) \wedge (bac = 0) \Rightarrow (d = 0)$	0.5	0.166
28: $(cba = 1) \wedge (ccc = 0) \wedge (bac = 0) \Rightarrow (d = 0)$	0.5	0.166
29: $(cac = 0) \wedge (cba = 1) \wedge (bcc = 0) \Rightarrow (d = 0)$	0.5	0.166
30: $(cac = 0) \wedge (cba = 1) \wedge (ccc = 0) \Rightarrow (d = 0)$	0.5	0.166
31: $(cba = 1) \wedge (bcc = 0) \wedge (bac = 0) \Rightarrow (d = 0)$	0.5	0.166

Table 8. Occurrence-based coverage of sequential decision rules for strict size 3

decision rules for strict size 3	o_coverage	vo_coverage
1: $(aab = 1) \Rightarrow (d = 1)$	1.0	1.0
2: $(aac = 1) \Rightarrow (d = 1)$	1.0	1.0
3: $(abc = 1) \Rightarrow (d = 1)$	1.0	1.0
4: $(bca = 1) \Rightarrow (d = 1)$	1.0	1.0
5: $(acc = 1) \Rightarrow (d = 1)$	1.0	1.0
6: $(cba = 0) \wedge (bcc = 1) \Rightarrow (d = 0)$	0.25	0.5
7: $(cba = 0) \wedge (ccc = 1) \Rightarrow (d = 0)$	0.25	0.5
8: $(cba = 1) \wedge (bcc = 0) \Rightarrow (d = 0)$	0.25	0.5
9: $(cba = 1) \wedge (ccc = 0) \Rightarrow (d = 0)$	0.25	0.5

is only for strict size 3 subsequences:

$$(bca = 1) \Rightarrow (d = 1)$$

where acc is a strict size 3 subsequence.

In decision rules Nos.8 - 31 of Tables 7, we can see that $o\_coverage$ and $vo\_coverage$ have different values, i.e., $vo\_coverage$ returns lower values than $o\_coverage$. For example, decision rule No.7 has $o\_coverage = 1.0$ and $vo\_coverage = 1.0$ but decision rule No.8 has $o\_coverage = 0.333$ and $vo\_coverage = 1.0$. The variant of occurrence-based coverage $vo\_coverage$ derives some different evaluation results for the rules. In other words, decision rule No.8 cover all the occurred subsequences but does not cover all the occurred and non-occurred subsequences in the two inputed sequences.

6   Related Work

Many algorithms have been developed for sequential pattern mining in the area of database and knowledge discovery. As the first approach to the sequential pattern, Agrawal and Srikant [1] proposed apriori-based algorithms such as AprioriAll, AprioriSome, DynamicSome, and GSP (Generalized Sequential Pattern).

Following a similar approach, Pei et al. [22] provided a more efficient algorithm by using projections based on growing frequent prefixes, which is called PrefixSpan (Prefix-projected Sequential Pattern Mining). Similar to PrefixSpan, Ayres et al. [2] developed SPAM (Sequential Pattern Mining) with a pruning method using a bitmap representation to store each sequence. Zaki [32] presented the algorithm SPADE (Sequential Pattern Discovery using Equivalence classes) using a vertical id-list format method such that frequent sequences are expressed by a list of pairs of itemsets and item identifiers. Garofalakis et al. [6] proposed a sequential pattern mining method of user-specified regular expression constraints, whose algorithm is called SPIRIT (Sequential Pattern mIning with Regular expressIon consTraints). Furthermore, in order to generate more compact frequent patterns, the mining of closed sequential patterns has been studied with the algorithms CloSpan [30], BIDE [28], and IMCS [5].

Standard sequential pattern mining does not address partial patterns in sequence databases. To our knowledge, periodic pattern mining in temporal databases is most closely related to our proposed local pattern mining problem in rough set theory. In studying sequential pattern mining from temporal databases, the goal of looking at periodic pattern mining problems is to find frequent partial patterns in the period segments of a sequence. Han et al. [7] designed an Apriori-like algorithm to efficiently mine partial periodic patterns in time series databases; however, the patterns identified are synchronous in time series data. To enable more flexible periodic pattern mining, Yang at al. [31] addressed the problem of asynchronous periodic pattern mining by searching for all of the periodic patterns whose positions may be shifted in time series data. In this approach, the number of occurrences of each pattern is counted within a

user-specified maximum number of disturbances between segments of the time series data.

Our proposed method of pattern mining as well as periodic pattern mining can discover partial patterns; however, we do not set segments of a sequence because the distance between itemsets is not important when action sequences are analyzed. For example, the partial pattern ab of the two actions a and b is discovered from the two action sequences $accb$ and $abcd$, which the periodic pattern mining algorithm cannot find. In addition, our problem focuses on the discernibility of local patterns between sequences, while the periodic pattern mining discovers patterns that repeat themselves in sequences.

There are many discussions about accuracy and coverage measures of decision rules. Pawlak [18, 19] discussed these measures from the viewpoint of Bayes' theorem. Tsumoto [26, 27] also discussed theoretical characteristics of the accuracy and coverage measures in rough set theory. From the viewpoint of attribute dependency, Pawlak [20] and Pawlak and Skowron [21] discussed the degree of attribute dependency between condition and decision attributes using positive regions of decision classes. Yamaguchi [29] pointed out technical problems in Pawlak's attribute dependency model and proposed a new attribute dependency measure using the discernibility matrix [23] of decision table proposed by Skowron and Rauszer. Moreover, Holeňa [9] discussed generalization of evaluation criteria of individual rules to whole sets of rules extracted from data. These accuracy and coverage measures of decision rules in rough set theory are based on semantics of decision rules. However, our proposed accuracy and coverage measures for sequential decision rules are based on the occurrence of subsequences in sequential decision rules. That is, varieties of local sequence patterns occurring in each sequence are measured by calculating the sum of numbers of the occurring patterns. Importantly, if every pattern in a sequential decision rule indicates no occurrence (e.g., $(cba = 0) \wedge (bcc = 0) \Rightarrow (d = 1)$), it is excluded in our accuracy and coverage measures for sequential decision rules.

In the area of rough set theory, there are a few studies that concern with sequential pattern mining. Skowron and Synak [24, 25] and Synak et al. [15] proposed spatio-temporal approximate reasoning and production rules based on hierarchical information maps. In particular, Bazan [3] introduced ideas of temporal concepts defined on time windows and temporal patterns. Temporal concepts are granules of time points with some meanings that are specified by human experts. He proposed temporal information systems that describe sequences of time points using time windows, while sequential information systems we proposed in this paper describe sequences of items. Hirano and Tsumoto [8] presented a method for finding patterns from spatio-temporal data using rough set-based clustering. This approach can group sequences from a single spatio-temporal information system wherein data are associated with time and spatial positions.

7 Conclusion and Future Work

We have proposed an alternative method for mining sequential patterns for sequential data using rough set theory. In our method, we represent the local features of sequences by using a sequential information system where attributes correspond to the occurrence of size k subsequences as local patterns. The proposed mining algorithm computes sequential decision rules according to the size of subsequences by changing the size from 2 to a maximal number in order to check different granulates for sequential data. We evaluate the occurrence-based accuracy and coverage of the sequential decision rules so that we can discover local patterns of sequences that result in a decision.

We plan to develop a sequential pattern mining algorithm for sequential data with time stamps. Such sequential data is more complicated to mine because we have to analyze the timing of data in addition to the order of data. For example, with the same two sequences ab and ab, the temporal gaps between a and b may be different in the time stamps of a and b.

Acknowledgment

References

1. R. Agrawal and R. Srikant. Mining sequential patterns. In *Proceedings of the 11th international conference on data engineering (ICDE'95)*, pages 3–14, 1995.
2. J. Ayres, J. Gehrke, T. Yiu, and J. Flannick. Sequential pattern mining using a bitmap representation. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 429–435, 2002.
3. J. G. Bazan. Hierarchical classifiers for complex spatio-temporal concepts. *Transactions on Rough Sets*, IX:474–750, 2008.
4. J. G. Bazan, H. S. Nguyen, S. H. Nguyen, P. Synak, and J. Wróblewski. Rough set algorithms in classification problem. In L. Polkowski, S. Tsumoto, and T. Young Lin, editors, *Rough set methods and applications*, pages 49–88. Physica-Verlag, 2000.
5. L. Chang, T. Wang, D. Yang, H. Luan, and S. Tang. Efficient algorithms for incremental maintenance of closed sequential patterns in large databases. *Data Knowl. Eng.*, 68(1):68–106, 2009.
6. M. N. Garofalakis, R. Rastogi, and K. Shim. Mining sequential patterns with regular expression constraints. *IEEE Trans. Knowl. Data Eng.*, 14(3):530–552, 2002.
7. J. Han, G. Dong, and Y. Yin. Efficient mining of partial periodic patterns in time series database. In *Proceedings of the 15th International Conference on Data Engineering (ICDE)*, pages 106–115, 1999.
8. S. Hirano and S. Tsumoto. A clustering method for spatio-temporal data and its application to soccer game records. In *Proceedings of the 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 612–621, 2005.

9. M. Holeňa. Measures of ruleset quality for general rules extraction methods. *Int J Approx. Reason.*, 50:867–879, 2009.

10. M. Inuiguchi. Rule induction based on multi-agent rough sets. In *Computational Intelligence and Industrial Applications: Proceedings of ISCIIA 2006*, pages 320–327, 2006.

11. K. Kaneiwa. A rough set approach to mining connections from information systems. In *Proceedings of the 25th ACM Symposium on Applied Computing (ACM SAC 2010), Track on Data Mining*, pages 990–996, 2010.

12. K. Kaneiwa. A rough set approach to multiple datasets analysis. *Applied Soft Computing*, 11(2):2538–2547, 2011.

13. K. Kaneiwa and Y. Kudo. Local pattern mining from sequences using rough set theory. In *Proceedings of the 2010 IEEE International Conference on Granular Computing (GrC 2010)*, pages 247–252, 2010.

14. Y. Kudo and T. Murai. A note on characteristic combination patterns about how to combine objects in object-oriented rough set models. In *Third International Conference on Rough Sets and Knowledge Technology (RSKT 2008)*, pages 115–123, 2008.

15. A. Skowron P. Synak, J. G. Bazan and J. F. Peters. Spatio-temporal approximate reasoning over complex objects. *Fundam. Inform.*, 67:249–269, 2005.

16. S. K. Pal and P. Mitra. *Pattern Recognition Algorithms for Data Mining: Scalability, Knowledge Discovery, and Soft Granular Computing.* Chapman & Hall, Ltd., 2004.

17. Z. Pawlak. *Rough Sets: Theoretical Aspects of Reasoning about Data.* Kluwer Academic Publishers, 1992.

18. Z. Pawlak. Decision rules, bayes' rule and rough sets. In *Proceedings of the 7th International Workshop on Rough Sets and Current Trend in Computing*, pages 1–9, 1999.

19. Z. Pawlak. Drawing conclusion from data—the rough set way. *Int J Intell. Syst.*, 16:3–11, 2001.

20. Z. Pawlak. Decision rules and dependencies. *Fundam. Inform.*, 60:33–39, 2004.

21. Z. Pawlak and A. Skowron. Rudiments of rough sets. *Inform. Scis.*, 177:3–27, 2007.

22. J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Mining sequential patterns by pattern-growth: The prefixspan approach. *IEEE Trans. Knowl. Data Eng.*, 16(11):1424–1440, 2004.

23. A. Skowron and C. M. Rauszer. The discernibility matrix and functions in information systems. In Roman Słowiński, editor, *Intelligent Decision Support: Handbook of Application and Advances of the Rough Set Theory*, pages 331–362. Kluwer Academic Publisher, 1992.

24. A. Skowron and P. Synak. Hierarchical information maps. In *Proceedings of the 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 622–631. Springer-Verlag, 2005.

25. A. Skowron and P. Synak. Planning based on reasoning about information changes. In *Proceedings of the 5th International Conference on Rough Sets and Current Trend in Computing*, pages 165–173. Springer-Verlag, 2006.

26. S. Tsumoto. Accuracy and coverage in rough set rule induction. In *Proceedings of the Third International Conference on Rough Sets and Current Trend in Computing*, pages 373–380. Springer-Verlag, 2002.

27. S. Tsumoto. Characteristics of accuracy and coverage in rule induction. In *Proceedings of the 9th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing*, pages 237–244. Springer-Verlag, 2003.

28. J. Wang and J. Han. Bide: Efficient mining of frequent closed sequences. In *Proceedings of the 20th International Conference on Data Engineering (ICDE 2004)*, pages 79–90, 2004.

29. D. Yamaguchi. Attribute dependency functions considering data efficiency. *Int J Approx. Reason.*, 51:89–98, 2009.

30. X. Yan, J. Han, and R. Afshar. CloSpan: Mining closed sequential patterns in large databases. In *Proceedings of the Third SIAM International Conference on Data Mining*, pages 166–177, 2003.

31. J. Yang, W. Wang, and P. S. Yu. Mining asynchronous periodic patterns in time series data. *IEEE Trans. Knowl. Data Eng.*, 15(3):613–628, 2003.

32. M. J. Zaki. Spade: An efficient algorithm for mining frequent sequences. *Machine Learning*, 42(1/2):31–60, 2001.

33. Q. Zhao and S. S. Bhowmick. Sequential pattern mining: A survey. Technical Report 2003118, Nanyang Technological University, Singapore, 2003.