International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:2, No:11, 2008

Discovery of Sequential Patterns based on Constraint Patterns

Shigeaki Sakurai, Youichi Kitahata, and Ryohei Orihara

***Abstract*—This paper proposes a method that discovers sequential patterns corresponding to user's interests from sequential data. This method expresses the interests as constraint patterns. The constraint patterns can define relationships among attributes of the items composing the data. The method recursively decomposes the constraint patterns into constraint subpatterns. The method evaluates the constraint subpatterns in order to efficiently discover sequential patterns satisfying the constraint patterns. Also, this paper applies the method to the sequential data composed of stock price indexes and verifies its effectiveness through comparing it with a method without using the constraint patterns.

***Keywords*—Sequential pattern mining, Constraint pattern, Attribute constraint, Stock price indexes

I. Introduction

Owing to the progress of computer and network environments, it is easy to collect and store sequential data such as daily business reports, web log data, and physiological information. The need for the data analysis is increasing, because new knowledge is buried in them. Many studies have been done on this theme. This paper focuses on analysis of discrete sequential data, which is one of the topics related to this theme. Agrawal and Srikant [1], Ayres et al. [2], Pei et al. [4], Z. Yang and M. Kitsuregawa [9], and Zaki [10] propose methods that efficiently discover frequent sequential patterns from discrete sequential data. The patterns are regarded as new knowledge. However, the patterns do not always correspond to the user's interests, because the patterns are common and there are too many of them.

Garofalakis et al. [3], Pei et al. [5], Sakurai and Ueno [6], Sakurai et al. [7], and Srikant and Agrawal [8] propose methods that discover sequential patterns corresponding to the user's interest based on the background knowledge given by the user. That is, Garofalakis et al. [3] propose a method based on the regular expression constraint that uses user-specified regular expressions as the background knowledge. The method applies sequential patterns to the regular expressions and extracts only sequential patterns that satisfy the regular expressions. Srikant and Agrawal [8] propose a method that introduces time constraints, a time window, and taxonomy. Here, the time constraints specify the minimum and the maximum time period between adjacent item sets, the time window specifies items included in the same item set, and the taxonomy allows that sequential patterns include items across all levels of the taxonomy. The method has good scale-up properties with respect to data size. Pei et al. [5] present 7 kinds of constraints, including an item constraint, a superpattern constraint, and a regular expression constraint. Here, the item constraint can extract sequential patterns that include or do not include specific items, and the superpattern constraint can extract sequential patterns that include specific sequential subpatterns. Pei et al. [5] investigate characterization of the constraints and propose a new framework that characterizes the constraints. Sakurai and Ueno [6] propose a method that introduces specific pattern constraints and 7 kinds of time constraints. The time constraints can set more flexible constraints between items than Srikant and Agrawal [8]. The method extracts sequential patterns that include specific patterns and satisfy time constraints.

These methods deal with items expressing two values. That is, the existence of each item expresses whether the item is or is not included in sequential data. The items are independent of one another. These methods cannot deal with items dependent on one another. For this problem, Sakurai et al. [7] propose an attribute constraint. The constraint uses specific relationships among items included in specific item sets. The constraint can restrict items that occur simultaneously and items that occur continuously.

However, the attribute constraints deal with limited constraints between items with three or more values and cannot extract sequential pattern satisfying flexible relationships among the items. Therefore, this paper proposes a new method that flexibly expresses relationships among items in order to efficiently discover sequential patterns corresponding to the user's interests. Also, this paper applies the method to sequential data composed of stock price indexes and verifies its effectiveness through numerical experiments.

II. Constraint patterns

A. Sequential pattern

This paper deals with a sequential pattern composed of rows of item sets. Here, each item set has some items that occur at the same time, but each item set does not have multiple identical items. Formally, a sequential pattern s_x is described as $(l_{x1}, l_{x2}, \cdots, l_{xn_x})$, where l_{xi} is an item set and n_x is the number of the item sets included in the sequential pattern. The number n_x is called length and the pattern s_x is called n_x-th sequential pattern. Each l_{xi} is described as $\{v_{xi1}, v_{xi2}, \cdots,$

Shigeaki Sakurai is with the System Engineering Laboratory, Corporate Research & Development Center, Toshiba Corporation, Kawasaki, e-mail: shigeaki.sakurai@toshiba.co.jp

Youichi Kitahara is with the System Engineering Laboratory, Corporate Research & Development Center, Toshiba Corporation, Kawasaki, e-mail: youichi.kitahata@toshiba.co.jp

Ryohei Orihara is with the HumanCentric Laboratory, Corporate Research & Development Center, Toshiba Corporation, Kawasaki, e-mail: ryohei.orihara@toshiba.co.jp

$v_{xin_{xi}}\}$, where v_{xij} is an item, $v_{xik_1} \neq v_{xik_2}$ $(k_1 \neq k_2)$, and n_{xi} is the number of items included in l_{xi}.

On the other hand, when two sequential patterns s_1 and s_2 are given, their inclusion $s_2 = (l_{21}, l_{22}, \cdots, l_{2n_2}) \subseteq s_1 = (l_{11}, l_{12}, \cdots, l_{1n_1})$ is defined: $\exists$ $\{z_1, z_2, \cdots, z_{n_2}\}$ such as $z_1 < z_2 < \cdots < z_{n_2}$ and $l_{21} \subseteq l_{1z_1}$, $l_{22} \subseteq l_{1z_2}$, $\cdots$, $l_{2n_2} \subseteq l_{1z_{n_2}}$. Figure 1 shows an example of the inclusion. In this figure, each circle shows an item and the same items have the same pattern on the circle. The concept of inclusion is used in evaluating the frequency of sequential patterns. The frequency is the number of sequential data including the patterns. Many sequential pattern mining methods discover sequential patterns whose supports are larger than or equal to the minimum support given by the user. The support is defined by Formula (1) [1]. Here, s is a sequential pattern, $f_s(s)$ is frequency of the pattern s, and N is the total number of sequential data.

$$supp(s) = \frac{f_s(s)}{N} \tag{1}$$

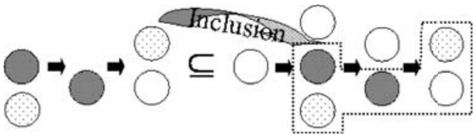


Fig. 1. Inclusion of sequential patterns [7]

B. Expression of items

Many sequential pattern mining methods [3] [5] [6] [8] deal with items independent of one another. However, some items are not independent of one another. For example, we note changes in stock price indexes. Here, the changes are three kinds of values: "up", "flat", and "down". A change for a company at one time is one of the three values. The values do not occur simultaneously. The values are dependent on one another. On the other hand, if the items corresponding to the values are defined, the sequential pattern mining methods generate the combinations of the items. However, the combinations cannot occur simultaneously. The combinations cannot be sequential patterns. That is, we can remove the combinations from candidate sequential patterns without calculating the frequency of the combinations. The relationships among items lead to more efficient discovery of sequential patterns.

Thus, this paper introduces the concept of attributes to items in order to deal with dependent items, where the attributes show groups of items that have a common property. An item v is composed of both an attribute A and an attribute value a. The item v is described by $A : a$. For example, in the case A="α company" and a="up", the item is described as "α company: up". Similarly, in the case A="α company" and a="down", the item is described as "α company: down". We can express the dependent items by using the attribute A.

C. Introduction of constraint patterns

We expect that the user can describe relationships among items corresponding to the user's interests to some extent. For example, we note changes in stock price indexes for three companies: α company, β company, and γ company. The changes are composed of three kinds of values as shown in subsection II-B. That is, 9 items such as "α company: up", "α company: flat", "α company: down", $\cdots$, and "γ company: down" are given. If the user is interested in relationships among three companies such as $\{\alpha$ company: up, γ company: down$\}$, it is reasonable to extract only the sequential patterns including the relationships. We can expect that the extracted patterns correspond to the interests with high probability. However, sequential patterns based on relationships of specific items are limited and it is difficult for the user to set the relationships. Also, the relationships cannot discover sequential patterns based on relationships among items that the user does not notice. The relationships cannot always discover all sequential patterns satisfying the user's interests.

Thus, this paper introduces constraint patterns in order to easily extract valid sequential patterns. The constraint patterns decide concrete attribute values but do not decide concrete attributes. The constraints are most efficient in the case that each attribute has the same kinds of attribute values. Therefore, changes of the stock price indexes are a typical example. Also, the constraints can be used in the case that some attributes have common attribute values. The constraint patterns can efficiently discover sequential patterns satisfying the relationships among multiple attributes and their attribute values. If the sequential subpatterns of the discovered patterns are given, the patterns can be used in order to predict their remaining subpatterns. In the following, a constraint pattern is formally described as shown in Formula (2).

$$\begin{aligned} &(C_1, C_2, \cdots C_m), \\ &\quad C_i = \{x_1 : a_{i1}, x_2 : a_{i2}, \cdots, x_n : a_{in}\} \end{aligned} \tag{2}$$

Here, x_j $(j = 1, 2, \cdots, n)$ is an attribute variable, x_j corresponds to an attribute, and x_{j_1} and x_{j_2} are different attributes in the case that $j_1 \neq j_2$. a_{ij} shows an attribute value included in x_j. The constraint patterns can extract sequential patterns whose lengths are m and each item set is composed of n items. In the following, the constraint pattern is called m-th constraint patterns when the length is m. In particular, the constraint pattern is called a constraint item set when the number of item sets included in the pattern is 1. The constraint item set is called a constraint item when the number of items included in the item set is 1. Also, frequent sequential patterns, frequent item sets, and frequent items satisfying one of the constraint patterns are called characteristic sequential patterns, characteristic item sets, and characteristic items, respectively.

Here, we note that the user is usually interested in sequential patterns whose lengths are within a range corresponding to target tasks. Therefore, some constraint patterns with different lengths are simultaneously set by the user.

For example, in the case of the changes in stock price indexes, a constraint pattern (C_1, C_2, C_3) is given, where C_1=$\{x_1$: up, x_2: flat$\}$, C_2=$\{x_1$: up, x_2: down$\}$ and C_3=$\{x_1$: flat, x_2: down$\}$. The constraint pattern corresponds to three constraint conditions as shown in Figure 2.

({α company: up, β company: flat}, {α company: up, β company: down}, {α company: flat, β company: down})
({α company: up, γ company: flat}, {α company: up, γ company: down}, {α company: flat, γ company: down})
({β company: up, γ company: flat}, {β company: up, γ company: down}, {β company: flat, γ company: down})

Fig. 2. Examples of constraint conditions corresponding to a constraint pattern

D. Decomposition of constraint patterns

When the kinds of attribute values, the number of the attributes corresponding to the values, and the length of constraint patterns are large, the combinations of attributes and attribute values are very large. The combinations exponentially increase. Therefore, a sequential pattern mining method cannot generate all combinations in advance. Thus, this paper proposes a method that checks the constraint patterns step by step. In the following, we deal with a sequential pattern mining method such as AprioriAll [1]. But, the constraint patterns can be used in sequential pattern mining methods such as PrefixSpan [4] and SPADE [10]. The method generates items, item sets, and sequential patterns in order. The method generates candidate item sets and evaluates whether they are frequent or not. Also, the method generates candidate sequential patterns and evaluates whether they are frequent or not. The method repeats the generation steps and the evaluation steps until all frequent item sets or all frequent sequential patterns are discovered.

At first, we note the generation method of a candidate item set. Figure 3 shows the outline of the method. The method generates a candidate item set with $(i + 1)$ items from two frequent item sets with i items. Here, each frequent item set is composed of $(i - 1)$ common items and a different item. The candidate item set is composed of $(i - 1)$ common items and two different items. In each frequent item set, the items are arranged in a specific order such as an alphabetic order. The items included in the candidate item set also keep the order. The candidate item set is evaluated as to whether it is frequent or not.

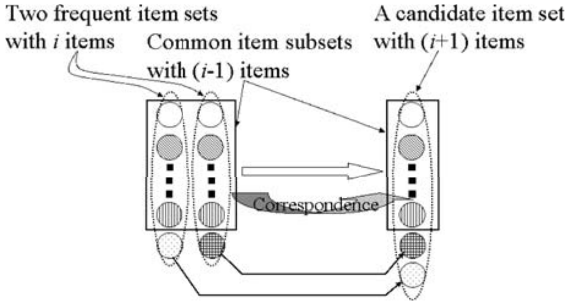


Fig. 3. Generation of a candidate item set

If the candidate item set satisfies a constraint item set, it is necessary for the two frequent item sets to satisfy constraint item subsets. The constraint item subsets are subsets included in the constraint item set. On the contrary, if the frequent item sets do not satisfy the constraint item subsets, the candidate item set does not satisfy the constraint item set. The sequential pattern mining method can avoid generating a candidate item set by checking whether two frequent item sets satisfy the constraint item subsets or not.

Next, we note the generation of candidate sequential patterns. Figure 4 shows the outline of the method. That is, the method generates a $(k + 1)$-th candidate sequential pattern from two k-th frequent sequential patterns. Here, each frequent sequential pattern is composed of $(k - 1)$ common item sets and a different item set. The different item sets are the last item sets in the frequent sequential patterns. The candidate sequential pattern is composed of $(k - 1)$ common item sets and two different item sets. In Figure 4, the last item set in the upper frequent sequential pattern and the last item set in the lower one are arranged in the order. The different order of the last items generates a different candidate sequential pattern. The generated candidate sequential pattern is evaluated as to whether it is frequent or not.

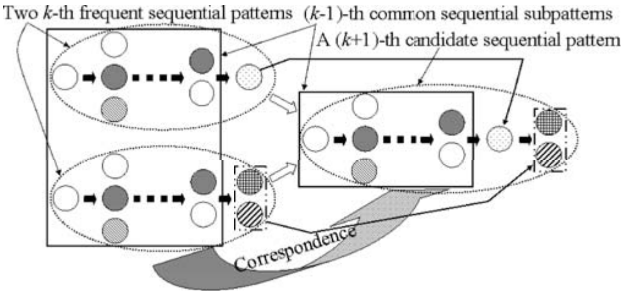

Two k-th frequent sequential patterns
$(k$-1)-th common sequential subpatterns
A $(k$+1)-th candidate sequential pattern
Correspondence

Fig. 4. Generation of a candidate sequential pattern

If the candidate sequential pattern satisfies a constraint pattern, it is necessary for the two frequent sequential patterns to satisfy constraint subpatterns. The constraint subpatterns are subpatterns included in the constraint pattern. The sequential pattern mining method can avoid generating a candidate sequential pattern by checking whether two frequent sequential patterns satisfy the constraint subpatterns or not. The calculation time of the check is much faster than that of frequency of the candidate sequential pattern. Therefore, the method can efficiently discover all sequential patterns satisfying the constraint patterns by checking constraint subpatterns and constraint item subsets.

We study a method that generates constraint subpatterns and constraint item subsets from a constraint pattern. At first, we note the generation of the candidate sequential patterns. The generation implies unique decomposition of a candidate sequential pattern into two frequent sequential patterns. The constraint subpatterns are applied to the two frequent sequential patterns. Therefore, we can decide the subpatterns by the inverse operation. That is, in the case that the constraint pattern is Formula (2), the pattern is decomposed into two constraint subpatterns as shown in Formula (3). The subpatterns are regarded as new constraint patterns. The decomposition is repeated until the length of all unprocessed constraint patterns is 1.

$$\begin{array}{l}(C_1, \cdots C_{m-2}, C_{m-1}),\\ (C_1, \cdots C_{m-2}, C_m)\end{array} \tag{3}$$

Next, we note the generation of the candidate item sets. The generation implies unique decomposition of a candidate item set into two frequent item sets. The constraint item subsets are applied to the two frequent sequential item sets. Therefore, we can decide the subsets by the inverse operation. That is, in the case that a constraint item set C_i is given as shown in Formula (2), the set is decomposed into two constraint subsets as shown in Formula (4). The subsets are regarded as new constraint item sets. The decomposition is repeated until the number of items in all unprocessed constraint item sets is 1.

$$\begin{array}{l}\{x_1 : a_{i1}, \cdots, x_{n-2} : a_{in-2}, x_{n-1} : a_{in-1}\},\\ \{x_1 : a_{i1}, \cdots, x_{n-2} : a_{in-2}, x_n : a_{in}\}\end{array} \tag{4}$$

For example, we note the constraint pattern given in subsection II-C. The constraint is decomposed as shown in Figure 5. That is, the constraint is decomposed into two constraint subpatterns: ($\{x_1$: up, x_2: flat$\}$, $\{x_1$: up, x_2: down$\}$) and ($\{x_1$: up, x_2: flat$\}$, $\{x_1$: flat, x_2: down$\}$). The former constraint subpattern is decomposed into two constraint item subsets: $\{x_1$: up, x_2: flat$\}$ and $\{x_1$: up, x_2: down$\}$, and the latter one is decomposed into two constraint item subsets: $\{x_1$: up, x_2: flat$\}$ and $\{x_1$: flat, x_2: down$\}$. Three constraint item subsets are generated, because the constraint item subset $\{x_1$: up, x_2: flat$\}$ is redundant. Lastly, three constraint items: "up", "flat", and "down" are extracted from the constraint item subsets. We note that attribute variables are neglected in the constraint items, because the difference of the attribute variables does not influence constraint conditions.

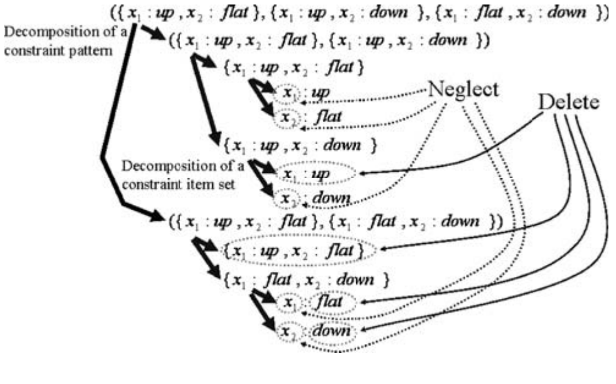

({x1 : up , x2 : flat }, {x1 : up , x2 : down }, {x1 : flat , x2 : down })
Decomposition of a constraint pattern
({x1 : up , x2 : flat }, {x1 : up , x2 : down })
{x1 : up , x2 : flat }
x1 : up
x2 : flat
Neglect
Delete
{x1 : up , x2 : down }
Decomposition of a constraint item set
x1 : up
x2 : down
({x1 : up , x2 : flat }, {x1 : flat , x2 : down })
{x1 : up , x2 : flat }
{x1 : flat , x2 : down }
x1 : flat
x2 : down

Fig. 5. An example of decomposition of a constraint pattern

The user can usually give some constraint patterns. These patterns are decomposed into constraint subpatterns and constrain item subsets, respectively. The same subpatterns and the same subsets are deleted. Table I shows an algorithm of the decomposition of the constraint sequential patterns. In this algorithm, original constraint patterns ConsDB are input to it and it outputs both k-th constraint subpatterns R_k and constraint item subsets with i items $R_{1,i}$. At first, the algorithm sets original constraint patterns to R_k and $R_{1,i}$ by referring to their length and the number of items included in their item sets (step 1∼step 7). Next, the algorithm picks up constraint patterns from R_k in a descending order of their length and decomposes them into constraint subpatterns. The algorithm sets the decomposed subpatterns to R_k according to their length. The decomposed subpatterns are regarded as new constraint patterns. The algorithm repeats the pattern decomposition until the maximum length of unprocessed constraint patterns is equal to 1 (step 8∼step 14). Next, the algorithm assigns constraint patterns included in R_1 into $R_{1,i}$ according to the number of items included in the patterns (step 15∼step 18). Next, the algorithm picks up constraint item sets from $R_{1,i}$ in a descending order of the number of items included in the item sets and decomposes them into constraint item subsets. The algorithm sets the decomposed subsets to $R_{1,i}$ according to the number of items included in them. The decomposed subsets are regarded as new constraint item sets. The algorithm repeats the set decomposition until the maximum number of the items included in unprocessed item sets is 1 (step 19∼step 25).

TABLE I
A CONSTRAINT PATTERN DECOMPOSITION ALGORITHM

step 1:	Initialize $mplen$=-1, $minum$=-1, R_k=ϕ, and $R_{i,1}$=ϕ.
step 2:	If ConsDB is not equal to ϕ, then extract a constraint pattern (p) from ConsDB. Otherwise, go to step 8.
step 3:	Calculate the length ($plen$) of p and the number ($inum$) of items included in item sets of p.
step 4:	If $plen$ is bigger than $mplen$, then $mplen$= $plen$.
step 5:	If $inum$ is bigger than $minum$, then $minum$= $inum$.
step 6:	If $plen$ is not equal to 1, then add p to R_{plen}. Otherwise, add p to $R_{1,inum}$.
step 7:	ConsDB= ConsDB$-p$ and return to step 2.
step 8:	k= $mplen$.
step 9:	If k is smaller than or equal to 1, then go to step 15.
step 10:	If R_k is not equal to ϕ, then extract a constraint pattern (p) form R_k. Otherwise, go to step 14.
step 11:	Decompose p into two constraint subpatterns (p_j, j=1, 2).
step 12:	If p_j is not included in R_{k-1}, then add p_j to R_{k-1}.
step 13:	R_k= $R_k - p$ and return to step 10.
step 14:	k= $k-1$ and return to step 9.
step 15:	If R_1 is not equal to ϕ, then extract a constraint pattern (p) form R_1. Otherwise, go to step 19.
step 16:	Calculate the number ($inum$) of items included in item sets of p.
step 17:	If p is not included in $R_{i,inum}$, then add p to $R_{1,inum}$.
step 18:	R_1= $R_1 - p$ and return to step 15.
step 19:	i= $minum$.
step 20:	If i is smaller than or equal to 1, then terminate the algorithm.
step 21:	If $R_{1,i}$ is not equal to ϕ, then extract a constraint pattern (c) form $R_{1,i}$. Otherwise, go to step 25.
step 22:	Decompose c into two constraint item subsets (c_j, j=1, 2).
step 23:	If c_j is not included in $R_{1,i-1}$, then add c_j to $R_{1,i-1}$.
step 24:	$R_{1,i}$= $R_{1,i} - c$ and return to step 21.
step 25:	i= $i-1$ and return to step 20.

The decomposed constraint patterns are applied to the generation steps of the sequential pattern mining method. If a candidate does not satisfy the constraints, the candidate is excluded without calculating its frequency. The method can efficiently discover all characteristic sequential patterns.

III. Numerical experiments

A. Data

We use sequential data of the stock price indexes on http://homepage1.nifty.com/hdatelier/data.htm (written in Japanese) in order to evaluate the effect of constraint patterns. The data are stock price indexes of each company listed on the Tokyo Stock Exchange. The data are composed of a company code, date, and 5 indexes: the start price, the maximum price, the minimum price, the end price, and the amount of the sales. The data range from September 2005 to September 2007. Also, we regard each company code or each synthetic index as an attribute and each daily change of an index as an attribute value in order to investigate relationships among stock price indexes of the companies and the synthetic indexes. Here, the synthetic indexes are JASDAQ, Nikkei-Average, and TOPIX. On the other hand, we calculate the change by dividing a value of an index in a business day with a value of the index in the next business day and judge which level the change is included in. Here, the level is composed of (2d+1) levels: x<d%, d%≤x<(d−1)%, $\cdots$, −2%≤x< −1%, −1%≤x<1%, 1%≤x<2%, $\cdots$, (d−1)%≤x<d%, and d%<x, where d is natural number and x is a daily change. In the following, the levels are referred to as L_{-d}, $L_{-(d-1)}$, $\cdots$, L_{-1}, L_0, L_1, $\cdots$, L_{d-1}, and L_d, respectively. A very long sequence is generated from a row of indexes corresponding to a company or a synthetic index. The sequence is divided into short sequences by using a sliding window. In this experiment, the sliding window includes 6 levels and goes to 4 levels forward along the direction of time axis after extracting a short sequence. For example, if the long sequence is L_{-3}, L_{-2}, L_{-1}, L_0, L_1, L_2, L_3, L_2, L_1, L_0, $\cdots$, the first short sequence is L_{-3}, L_{-2}, L_{-1}, L_0, L_1, L_2 and the second short sequence is L_1, L_2, L_3, L_2, L_1, L_0. Also, we generate 5 kinds of sequential data corresponding to 5 indexes. In each sequential data, each item set is composed of 28 items corresponding to 25 company codes and three synthetic indexes.

B. Evaluation method

We evaluate the number of sequential patterns in the case of using constraint patterns and not using the constraint patterns in order to verify the effectiveness of the constraint patterns. We use the 5 kinds of sequential data explained in subsection III-A. Also, we use constraint pattern sets as shown in Figure 6 and Figure 7. In these figures, an up-oblique arrow, a down-oblique arrow, and a flat arrow show levels L_d, L_0, L_{-d}, respectively. The connected arrows correspond to a constraint of an attribute. In the case of stock price indexes, the user is interested in the changes. In addition, the time interval between neighboring items of sequential patterns is 1. The constraint pattern sets can discover characteristic sequential patterns that focus on changes of two or three values included in different attributes.

On the other hand, the number of items included in frequent item sets is restricted within 3 in comparative experiments not using the constraint patterns. Also, we use 1%, 3%, 5%, 10%, 15%, and 30% as the minimum supports and use 3, 5, and 7 as the number of levels. We perform comparative experiments for each sequential data, each constraint pattern, and each minimum support.

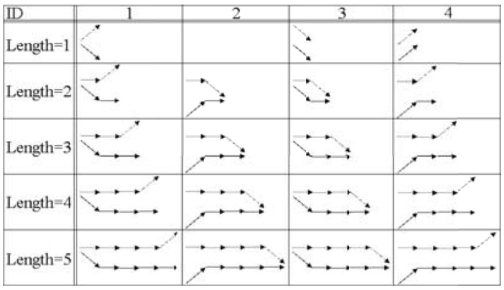


Fig. 6. Constraint patterns in the case that the number of attributes is 2

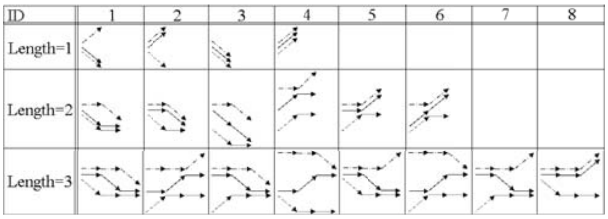


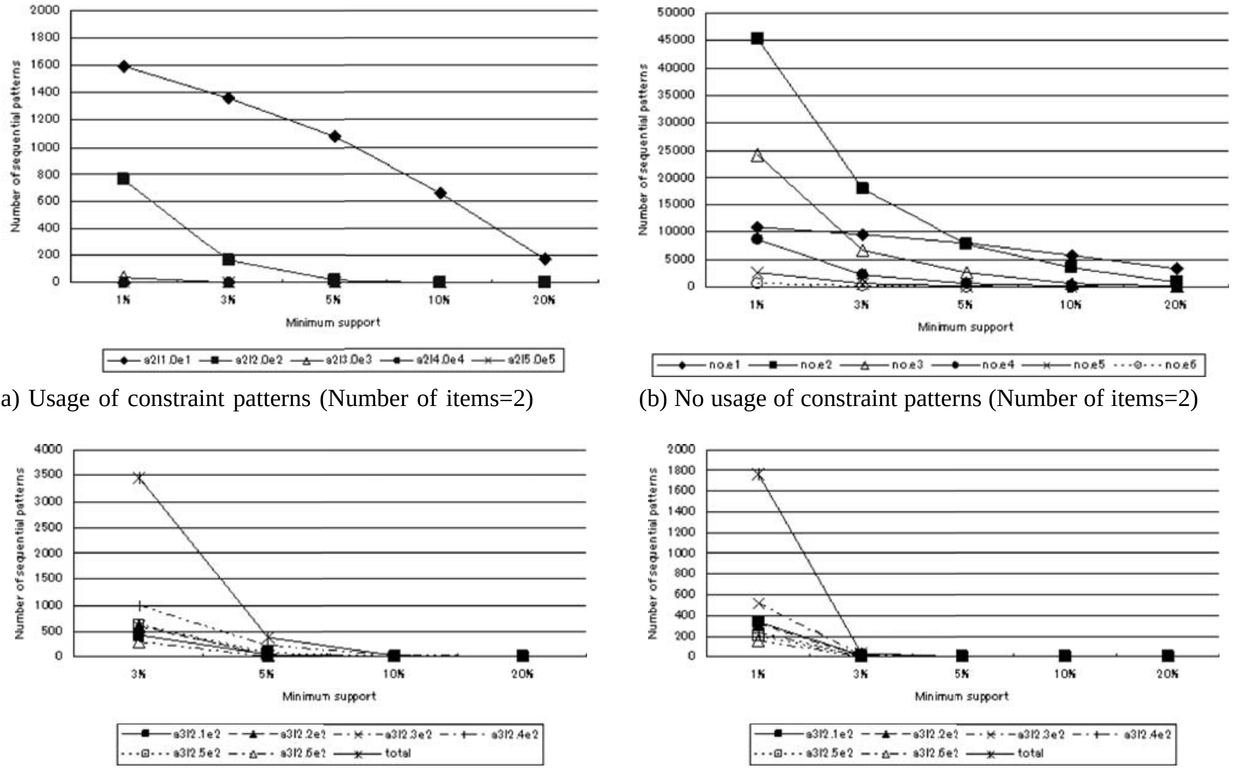
Fig. 7. Constraint patterns in the case that the number of attributes is 3

C. Experimental results

Figure 8 shows parts of experimental results. Figure 8 (a) shows results in the case that the constraint patterns in Figure 6 are used and the number of levels is 5, Figure 8 (b) shows results in the case that the constraint patterns are not used and the number of levels is 5, Figure 8 (c) shows results in the case that the constraint patterns with length 2 in Figure 7 are used and the number of levels is 3, Figure 8 (d) shows results in the case that the constraint patterns with length 2 in Figure 7 are used and the number of levels is 5. In each figure, horizontal axis shows the minimum support and vertical axis shows the number of sequential patterns. In each graph, the numbers behind "a" and "e" indicate the number of attributes and the length of extracted sequential patterns, respectively. "no" indicates the case in which constraint patterns are not used. The two numbers preceding "l" indicate the length of a constraint pattern and its ID, respectively. But, the second number is "0" when all constraint patterns with the length are applied simultaneously. For example, "a3l2.1e2" shows a result in the case that the number of attributes is 3, the length of a constraint pattern is 2, its ID is 1, and the length of extracted sequential patterns is 2. Also, "total" shows results that accumulate individual results corresponding to constraint patterns with the length. The results clearly correspond to the results that simultaneously apply all constraints with the length.

D. Discussion

Number of sequential patterns: We note the results in Figure 8 (a) and Figure 8 (b). The number of sequential patterns



(a) Usage of constraint patterns (Number of items=2)

(b) No usage of constraint patterns (Number of items=2)

(c) Second sequential patterns (Number of items=3, Number of levels=3)

(d) Second sequential patterns (Number of items=3, Number of levels=5)

Fig. 8. Experimental results

in the case that the constraint patterns are not used is 10 times larger than the one in the case that they are used. The results show that the constraint patterns can greatly reduce the number of sequential patterns. On the other hand, the sequential pattern mining method cannot discover all sequential patterns owing to the restriction of computer environments, if the constraint patterns are not used. It is possible for the constraint patterns to analyze relationships among three items. Therefore, the constraint patterns are efficient for the analysis of sequential data including dependent relationships.

Number of attribute values: We note the results in Figure 8 (c) and Figure 8 (d). The results show that the number of sequential patterns decreases as the number of attribute values becomes large. The decrease is due to the decrease of items with the same attribute values. If the number of attribute values is large, the sequential pattern mining method can extract sequential patterns with smaller support. It is important to design attribute values by referring to frequency of the items.

Length of constraint patterns: We note the results in Figure 8 (a) once more. The results show that the number of sequential patterns decreases as the length of constraint patterns becomes long. The long constraint patterns represent many constraints. The number of sequential patterns satisfying the constraints decreases dramatically. If the long constraint patterns are applied, the sequential pattern mining method can extract sequential patterns with smaller supports. Also, we note the results in Figure 8 (b). The results show that 6th sequential patterns are discovered in the case that the constraint patterns are not used. The sequential patterns do not correspond to the user's interest. The method can avoid the generation of redundant sequential patterns.

On the other hand, the constraint patterns restrict the length of sequential patterns. The constraint patterns cannot discover sequential patterns, if the length is unclear. However, the sequential pattern mining method can deal with the constraint patterns with different length simultaneously. That is, the user can define possible constraint patterns with the different length. Therefore, the restriction is irrelevant to us.

Decomposition of constraint patterns: The sequential pattern mining method can use all constraint patterns in Figure 6 and Figure 7 simultaneously. The large number of constraint patterns leads to the reduction of total calculation time, because the calculation of the common constraint pattern is performed at one time. On the other hand, the large number leads to generating a large number of candidate sequential patterns in order to generate longer sequential patterns, even if the candidates are not extracted as characteristic sequential patterns. Therefore, a computer may not be able to deal with the constraint patterns simultaneously owing to the restriction of available computer environments. Actually, the experiments based on all constraint patterns with the length 3 cannot discover all sequential patterns. The sequential pattern mining method requires individually applying each constraint pattern. In future work, it is necessary to decide the number of constraint patterns that is simultaneously set by referring to available computer environments.

Validity of sequential patterns: In this experiment, the constraint patterns are defined in order to investigate changes of the indexes among different companies and synthetic indexes. In the case of the stock price indexes, the user is interested in the changes. Therefore, the discovered sequential patterns are important to some extent, even if we do not check the individual sequential patterns in detail. The constraint patterns can discover the sequential patterns efficiently.

On the other hand, the constraint patterns depend on target tasks. Other target tasks may require different constraint patterns. We think that there are some typical constraint patterns corresponding to the user's interests. In future work, we intend to prepare for the constraint patterns processing typical target tasks.

Previous methods: Previous studies proposed sequential pattern mining methods based on other constraints. However, these constraints cannot sufficiently deal with items with dependent attribute values. That is, the regular expression constraint [3] cannot be described intuitively, even if it can express various relationships among items. The constraint requires users to be familiar with its description. The item constraint and the superpattern constraint [5] need specific items and specific subpatterns. These constraints require the users to have definite background knowledge in order to select the items and the subpatterns. Also, the attribute constraint [7] deals only with specific relationships among items. The constraint depends excessively on target tasks. In addition, the time constraint [6] [8] only restricts the interval between items and cannot deal with their contents.

On the other hand, the proposed constraint patterns can be described intuitively by using sequential combination of attribute values. The users can easily describe the constraint patterns. Also, they can use background knowledge including ambiguity by introducing the attribute variables. In addition, they can describe various types of relationships among items by processing constraint patterns whose lengths are different. Therefore, the constraint patterns can efficiently use background knowledge.

In light of the above the discussions, we believe that the proposed constraint patterns are important and the sequential mining method based on the patterns is effective.

IV. Summary and future work

This paper proposed a method that describes constraint patterns to discover sequential patterns corresponding to the user's interests using the user's background knowledge. The sequential mining method incorporating the constraint patterns efficiently discovers sequential patterns satisfying the interests described by the constraint patterns. Lastly, this paper verified the effectiveness of the proposed method by applying the method to sequential data of stock price indexes.

In future work, we will try to verify the validity of discovered patterns in more detail by listening to the opinions of the user. Also, we will try to clarify a method for simultaneous usage of constraint patterns based on the minimum support and the restriction of available computer environments. In addition, we will try to prepare for typical constraint patterns for other target tasks and to verify the effectiveness of the constraint patterns by applying them to other sequential data.

References

[1] R. Agrawal and R. Srikant, Mining Sequential Patterns, *Proc. of the 11th Intl. Conf. Data Engineering*, 1995, Taipei, Taiwan, pp. 3-14.

[2] J. Ayres, J. Flannick, J. Gehrke, and T. Yiu, "Sequential pattern mining using a bitmap representation," *Proc. of the 8th Intl. Conf. on Knowledge Discovery and Data Mining*, 2002, Edmonton, Alberta, Canada, pp. 429-435.

[3] M. N. Garofalakis, R. Rastogi, and K. Shim, "SPIRIT: Sequential Pattern Mining with Regular Expression Constraints," *Proc. of the Very Large Data Bases Conf.*, 1999, Edinburgh, Scotland, UK, pp. 223-234.

[4] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, M. -C. Hsu, "Mining Sequential Patterns by Pattern-Growth: The PrefixSpan Approach," *IEEE Trans. on Knowledge and Data Engineering*, vol. 16, no. 11, pp. 1424-1440, 2004.

[5] J. Pei, J. Han, and W. Wang, "Mining Sequential Patterns with Constraints in Large Databases," *Proc. of the 11th ACM Intl. Conf. on Information and Knowledge Management*, 2002, McLean, Virginia, USA, pp. 18-25.

[6] S. Sakurai, K. Ueno, and R. Orihara, "Discovery of Time Series Event Patterns based on Time Constraints from Textual Data," *Intl. J. of Computational Intelligence*, vol. 4, no. 2, pp. 144-151, 2008.

[7] S. Sakurai, Y. Kitahara, R. Orihara, K. Iwata, N. Honda, and T. Hayashi, "Discovery of Sequential Patterns Coinciding with Analysts' Interests," *J. of Computers*, vol. 3, issue 7, pp. 1-8, 2008.

[8] R. Srikant and R. Agrawal, "Mining Sequential Patterns: Generalizations and Performance Improvements," *Proc. of the 5th Intl. Conf. Extending Database Technology*, 1996, Avignon, France, pp. 3-17.

[9] Z. Yang and M. Kitsuregawa, "LAPIN-SPAM: An Improved Algorithm for Mining Sequential Pattern," *Proc. of the 21st Intl. Conf. on Data Engineering Workshops*, 2005, Tokyo, Japan, pp. 1222.

[10] M. J. Zaki, "SPADE: An Efficient Algorithm for Mining Frequent Sequences," *Machine Learning*, vol. 42, no. 1/2, pp, 31-60, 2001.

Shigeaki Sakurai received an MS degree in mathematics and a Ph.D. degree in industrial administration from Tokyo University of Science, Japan, in 1991 and 2001, respectively. He was a Professional Engineer of Japan in the field of information engineering in 2004.

He is a research scientist at the System Engineering Laboratory, Corporate Research & Development Center, Toshiba Corporation. His research interests include data mining, soft computing, and web technology.

Dr. Sakurai is a member of IEICE, SOFT, and JSAI.

Youichi Kitahara received an MS degree in earth system science and technology from Kyushu University, Japan, in 2003.

He works at the System Engineering Laboratory, Corporate Research & Development Center, Toshiba Corporation. His research interests include data mining and machine learning.

Ryohei Orihara received a BS degree, an MS degree, and a Ph.D. degree in engineering from the University of Tsukuba, Japan, in 1986, 1988 and 1999, respectively.

He is the laboratory leader at the HumanCentric Laboratory, Corporate Research & Development Center, Toshiba Corporation. He is also a part-time associate professor at Tokyo Institute of Technology, Japan. His research interests include machine learning, creativity support systems, analogical reasoning, metaphor understanding, data mining and text mining.

Dr. Orihara is a member of IPSJ, JSAI, and JSSST.