Approximate Sequential Patterns for Incomplete Sequence Database Mining

Céline Fiot, Anne Laurent, Maguelonne Teisseire

Approximate Sequential Pattern for Incomplete Sequence Database Mining

Céline Fiot, Anne Laurent and Maguelonne Teisseire

***Abstract*— Industrial databases often contains a large amount of unfilled information. When these data are mined for frequent sequences, incomplete data are, most of the time, deleted, which leads to an important loss of information. Extracted knowledge then becomes less representative of the database. Two techniques can then be investigated: either using only the available information or estimating the missing values. In this paper we propose an estimation-based approach that represents inclusion of an item within a record by a fuzzy set. Then the membership degree giving the item inclusion is used to compute the frequency of a sequence. Experiments run on various synthetic datasets show the feasibility and validity of our proposal as well in terms of quality as in terms of the robustness to the rate of missing values.**

I. Introduction

For the last decade, data mining application fields have widened. Especially it is the case of sequential pattern mining approaches [1]. This data mining technique aims at discovering knowlegde from temporal databases. First designed to analyse customer behavior, they are now used in various industrial, medical, or biological fields. The searched datasets are sets of time-stamped records. Each of them is constituted of a set of values.

However these kind of data often contain imperfection-ssuch as noise or incomplete data due to breakdowns or errors, for instance.
Moreover, existing sequential patterns methods only allow the analysis of complete data, without considering incomplete records, which may represent an important loss of information. Besides, replacement of missing values or their estimation are often either too simplistic to perform unbiased results, or too time-consuming to be applied on large datasets. Thus, it seems necessary to implement a method for mining sequential patterns within incomplete databases.
To solve this problem two tracks can be investigated. The concepts linked to sequential patterns can be adapted either to ignoremissing informationonly using the available one or to consider likely value of incomplete data.

In this paper we propose an estimation-based approach to discover frequent sequences within databases containing random missing values. This approach is instigated by an association rule mining method within incomplete databases [2], [3] and by a technique often used in machine learning or statistics: assessing missing value with respect to the values in the database. This principle consists in using available information, with a certain likelihood. Thus the whole dataset is used to discover the patterns, while taking into account uncertainty due to incomplete data instead of ignoring it, deleting it or simply replacing it by a single value.
We here redefine this principle for sequential pattern mining, thus proposing to discover approximate maximal frequent sequences within incomplete time-stamped databases. For this reason, we adapted the concepts linked to sequential pattern discovery. Then we implemented the ApSPoID algorithm using a sequential pattern mining algorithm, PSP [4]. This algorithm was tested on synthetic datasets to show the validity of our approach.

The remainder of the paper is organized as follows. Section II introduces the methods for association rule mining in the presence of missing values, and the concepts linked to sequential pattern discovery. Section III details our approach to mine for approximate sequential patterns within incomplete data, and we describe our algorithm in Section IV. Section V is then dedicated to experiments that show the validity of our approach. Finally, we conclude in section VI with a discussion on further work opened by this proposal.

II. From Sequential Patterns to Incompleteness

Sequential patterns are often introduced as an extension of association rules, initially proposed in [5]. They highlight correlations between database records as well as their temporal relationship. Even so these algorithms do not mine incomplete records contained in the database. These missing values must then be removed either by deletion or replacement. Quality of results then depends on this preprocess. Moreover this step is often time-consuming. In order to reduce the preprocessing due to missing values and to improve the sequential pattern quality, we propose a method for sequential pattern mining within incomplete databases. This method is based on association rules approaches. In this section we first define the concepts linked to sequential pattern mining, then we detail our motivations before introducing techniques that allow association rule mining handling missing values.

A. Sequential Patterns

Sequential patterns are based on the idea of *maximal frequent sequences*.
Let $\mathcal{R}$ be a set of objects records where each record R consists of three information elements: an object-id, a record timestamp and a set of attributes/items in the record. Let $I = \{i_1, i_2, ..., i_m\}$ be a set of items or attributes. An *itemset* is a non-empty set of attributes i_k, denoted by $(i_1 i_2 \dots i_k)$. It is a non-ordered representation. A *sequence* s is a non-

Céline Fiot , Anne Laurent and Maguelonne Teisseire are with the LIRMM, University of Montpellier II - CNRS, 34392 Montpellier, France (email: {fiot, laurent, teisseire}@lirmm.fr).

empty ordered list of itemsets s_p, denoted by $< s_1 s_2 ... s_p >$. A *n-sequence* is a sequence of n items (or of size n).

Example 1: Let us consider an example of market basket analysis. The object is a customer, and records are the transactions made by this customer. Timestamps are the date of transactions. If a customer purchases products e, a, k, u, and f according to the sequence $s =< (e)\ (a\ k)\ (u)\ (f) >$, then all items of the sequence were bought separately, except products a and k which were purchased at the same time. In this example, s is a 5-sequence.

One sequence $S =< s_1\ s_2 ... s_p >$ is a *subsequence* of another one $S' =<\ s'_1\ s'_2\ ...s'_m >$ if there are integers $l_1 < l_2 < ... < l_p$ such that $s_1 \subseteq s'_{l_1}$, $s_2 \subseteq s'_{l_2}$, ..., $s_p \subseteq s'_{l_p}$.

Example 2: The sequence s' = <(a) (f)> is a subsequence of s because (a) $\subseteq$ (a k) and (f) $\subseteq$ (f). However, <(a) (k)> is not a subsequence of s.

All records from the same object o are grouped together and sorted in increasing order of their timestamp. They are called a data sequence. An object *supports* a sequence s if it is included within the data sequence of this object (s is a subsequence of the data sequence). The *frequency* of a sequence ($freq(s)$) is defined as the percentage of objects supporting s in the whole set of objects $\mathcal{O}$. In order to decide whether a sequence is frequent or not, a minimum frequency value ($minFreq$) is specified by the user and the sequence is said to be frequent if the condition $freq(s) \geq minFreq$ holds. A sequence that may be frequent is a *candidate sequence*. Given a database of object records, the problem of sequential pattern mining is to find all maximal sequences of which the frequency is greater than a specified threshold ($minFreq$) [1]. Each of these sequences represents a sequential pattern, also called a maximal frequent sequence.

Several extensions were proposed to consider incremental mining for sequential patterns [6], to handle numerical and quantitative values [7], [8], [9] or to generalize sequential patterns with respect to various temporal parameters (time-interval between events of a sequence, grouping several records into a single itemset...) [10], [11], [12]. However, no technique was proposed to deal with missing values while sequential pattern mining. For this reason, in the following sections, we propose an approach that can mine maximal frequent sequences from an incomplete sequence database.

B. Motivations

Let us consider the database given by Table I. The goal is to extract the sequential patterns with a minimum frequency equal to 50%.

TABLE I
COMPLETE DATABASE.

Obj.	Sequence
O1	$(a\ b)\ (b\ c\ d)\ (b\ c\ e)$
O2	$(a)\ (b\ c)\ (b\ d)$
O3	$(a\ b)\ (b\ c)\ (b\ c\ d)$

The sequential patterns obtained are: <(a b)(b c d)>, <(a b)(b c)(b c)> and <(a)(b c)(b d)>.

Now, let us consider the incomplete database given by Table II. Some information in the data sequences has not been filled in and these values are missing. Let us consider that these values are identified as missing and unfilled. In order to mine for patterns, previous approaches require the suppression of missing values.

TABLE II
INCOMPLETE DATABASE.

Obj.	Sequence
O1	$(a\ b)\ (?\ ?\ c)\ (?\ b\ c)$
O2	$(a)\ (?\ c)\ (b\ d)$
O3	$(a\ b)\ (?\ c)\ (?\ ?\ c)$

After preprocessing, the database within which sequential patterns are mined is given Table III.

TABLE III
AFTER MISSING VALUE DELETION.

Obj.	Sequence
O1	$(a\ b)\ (c)\ (b\ c)$
O2	$(a)\ (c)\ (b\ d)$
O3	$(a\ b)\ (c)\ (c)$

Resulting patterns for $minFreq$ = 50% are: <(a b)(c)(c)> and <(a)(c)(b)>.

Note that only a short part of the database is used to discover information and only part of frequent patterns and frequent items of the complete dataset are found. In particular, item d is not frequent in the incomplete database. Therefore using the whole dataset without deleting part of it appears essential.

C. Association Rules and Missing Values

Some works were proposed to discover association rules within incomplete databases. The algorithm RAR proposed by [13] allow the user to consider incomplete data while association rule mining within incomplete relational databases, thanks to partial and temporary omission of such incomplete records. The whole database is not used to discover each rule but to generate the whole set of rules. This technique adapted to sequence mining in [14] gives interesting results. However, this approach requires the specification of a minimum significancy threshold and experiments show that if this threshold is not the optimal value, the method does not discover all the frequent patterns or too many parasitic patterns.

Therefore we develop a new approach, instigated by the proposals of [2], [3]. These methods implement an assessment systems based on a probabilistic distribution. With this approach one missing value can represent various values while mining for association rules. In order to consider those assessments, the notion of support (percentage of records in database that include the rule items) and thus confidence (probability for a record to contain the right part of the rule knowing it contains the left part) were reformulated.

From this principle we redefine in the next section the definitions linked to sequential pattern mining previously detailed.

III. ApSPoID: Dealing with Incompleteness

In this section, we present the definitions we used for our algorithm to mine incomplete sequence databases for sequential patterns.

A. Approximate Sequential Patterns for Incomplete Databases

As deleting incomplete records leads to an important loss of information, we adapted an association rule mining approach, robust to missing values. In this section, we describe our method ApSPoID, based on the principles of the $\sim$AR algorithm [2].

The main idea of our approach, as the one of the $\sim$AR method lies on the estimation of the value for unfilled items. While $\sim$AR computes the frequency of an incomplete itemset taking into account the probability of each likely value of a missing value, we propose here to use this estimation for the frequency computation of sequences.

Thus, while computing the item frequency of a sequence, for each missing value, several likely value may be checked. Each one of these possibilities is associated to a degree giving the likely appearance level. The support definition given in Section II is adapted in order to include this inclusion degree of the item.

Then inclusion of an item in an itemset is weighted by the certainty degree of its presence:

- If the item is in the itemset, the appearance level is equal to 1;
- If the item is not in the itemset and this itemset does not contain missing value, the appearance level is nul;
- Else, the appearance level of item i in the record r is in $]0,1[$. It will be denoted by $\mu_r(i)$.

Definition 1: The *frequency* of an itemset I is the average appearance level of the itemset for all the objects in the dataset. For each of them, the best occurrence is kept (i.e. the one being the most certain). The frequency is then computed by the formula (1) :

$$Freq(I) = \frac{\sum_{o \in \mathcal{O}} \perp_{r \in \mathcal{R}_o} \overline{\top}_{i \in I} \mu_r(i)}{|\mathcal{O}|} \quad (1)$$

Definition 2: Le *frequency* of a sequence S is the average appearance rate of the sequence for all the objects in the database. It is defined as the ratio of the sum of the best appearance level of the sequence for all the data sequences over the total number of data sequences. It is given by the formula (2):

$$Freq(S) = \frac{\sum_{o \in \mathcal{O}} \mathcal{F}(o, S)}{|\mathcal{O}|} \quad (2)$$

where $\mathcal{L}(S, o)$ is the appearance level of S in the data sequence of the object o.

Definition 3: The *appearance level* of a sequence S for an object o corresponds to the best inclusion rate of the sequence S in the data sequence of o. This appearance level is computed by the ordered agregation of the itemset inclusion rate in the order of appearance in sequence S. It is given by the formula (3):

$$L(S, o) = \perp_{r \in \mathcal{R}_o} OrdAg_{<s_1 \ldots s_i \ldots s_k>}(\overline{\top}_{j \in s_i} \mu_r(j)) \quad (3)$$

where k is the number of itemset in S.

B. Choosing Operators

Several operators could be used to agregate the item inclusion rates of an itemset or the itemset inclusion rate for a sequence. The method proposed by [2] uses the average agregation for item inclusion rates. However we consider that this type of agregation can hide a low appearance level of several items in a record if the other ones are certain. Therefore we propose to use a t-norm operator for this computation. More especially we will implement our approach with the min operator for its idempotency property [9].

Regarding the ordered agregation, we propose to use the average agregation, each itemset having the same "interest" in the sequence. Uncertainty is indeed already taken into account by the use of the t-norm. It is then not necessary to consider it a second time at the sequence level [9].

C. Computing the Appearance Level

First step of our method consits in determining each possible value for missing values and the certainty degree of each one.

An aproximation of incomplete record composition should be determined. To do so, the dataset is scanned to compute appearance rate of each item. As the database is incomplete, the appearance rate is computed as the frequency with respect to the number of records using the disabling technique of RAR [13]. This computation results in the frequency distribution that will give the appearance level of an item in a record. This value distribution is here computed with respect to the appearance within a record, but it could be computed with respect to the appearance within data sequences or only considering complete records.

Thus, parsing the database for a candidate sequence, when an itemset is found complete in a record, its appearance level is 1. When it is partially found in an incomplete record, the frequency distribution is used to determine the appearance level of the missing part of the itemset.

We present some experiment results that show the consequences of choosing one distribution or another on discovered patterns.

IV. Implementation

In this section, we first apply our defintions and principles on a brief example. Then, we present how our algorithm runs.

A. Example

Consider the incomplete database given by Table II. $minSup$ is equal to 50%.

First, the appearance rate distribution is computed. We use the formula given by [13] :

$$Freq(i) = \frac{|\{r \in \mathcal{R}/i \in r\}|}{|\mathcal{R}| - |\{r \in \mathcal{R}/r.isIncomplete(i)\}|}$$

For i = a : 3/(9-5) = 0.75. Table IV gives the frequency distribution for each value that could replace a missing one.

TABLE IV
FREQUENCY DISTRIBUTION FOR MISSING VALUE.

Item	a	b	c	d	e
Fréquence	0.75	0.8	0.56	0.25	0

Thus for the second record of the first object, the appearance level of c is 1, but a, b, d and e, it is given by the frequency distribution.

Then sequential patterns mining starts with the frequency computation for each item. Item a certainly appears in each data sequence then its frequency is 3/3=100%, the same for items b and c. The item d, could be supported by objects 1 and 3, iots frequency is then given by the following calculation:

$$\begin{aligned} Freq(< d >) &= \frac{\perp(0.25,0.25)+\perp(0.25,1)+\perp(0.25,0.25)}{3} \\ &= \frac{0.25+1+0.25}{3} \\ &= 0.5 \end{aligned}$$

Now, consider sequence S = $< (ab)\ (abcd) >$. This candidate sequence cannot be supported by none of the data sequences because none of them includes an itemset composed of 4 items. Consider sequence S' = $< (ab)\ (bc) >$, it is supported by object 1, cannot be supported by object 2but could be supported by the third object. For the frequency of S', we get:

$$Freq(S') = \frac{1 + 0 + (1 + \top(0.8, 1))/2}{3} = \frac{1 + 0.9}{3} = 63\%$$

Sequence S' is then frequent.

Applying our method, discovered sequential patterns for $minSup$=50% are : $< (ab)(bc)(abc) >$, $< (ab)(ac)(abc) >$, $< (a)(bc)(bd) >$, $< (a)(ac)(bd) >$, $< (b)(c)(bd) >$ and $< (a)(d)(b) >$.

Even if these patterns are not exactly the one obtained on the complete database, they are closer to the one we should get than the one discovered using the preprocessed dataset. Experiments detailed in section V show that there exists a value of the minimum representativity for which the algorithm SPoID extracts the whole set of sequential patterns of the complete database from an incomplete one.

B. Algorithm

The algorithm SPoID runs similarly to the generate-prune sequential pattern mining algorithms. It consists in generating all the candidate k-sequences from the frequent (k-1)-sequences. Then the database is scanned to count the number of data sequences that support each candidate sequence. The main difference stands in the counting of incomplete data sequences. This counting step uses the same counting principles used by the algorithm TotallyFuzzy for fuzzy sequential pattern mining [9]. It is described by Alg. 1.

ApSPoID - **Input:** $\mathcal{O}$, a sequence database ;
minFreq, user-specified minimum frequency
Ouput: $SPList$, list of frequent sequences

[Computation of the frequency distribution]
float[] $freqDist \leftarrow$ calcDist() ;

[Search for frequent items]
$C \leftarrow \{i \in \mathcal{I}\}$; k = 1 ;
$F \leftarrow$ getFrepnRep(C,$minFreq$) ;
$SPList$.add(F) ;

[Search for frequent sequences, size $\geq$ 2]
While ($C \neq \varnothing$) **do**
  k++ ; $C \leftarrow$ generate(F,k) ;
  For each candidate sequence $s \in C$ **do**
    For each objet $o \in \mathcal{O}$ **do**
      float[] degTab; $ag \leftarrow 0$; $id \leftarrow 1$;
      For (j=1 to $|\mathcal{R}_o|$ **do**
        $Trans \leftarrow \{r_j, ...r_{|\mathcal{R}_o|}\}$;
        findSequence(s,$Trans$,$degTab$) ;
        $ag \leftarrow \max(ag, \sum_{i=1}^{i=id} degTab[i]/id)$;
      End For
      $tmpFreq \leftarrow tmpFreq + ag$;
    End For
    $Freq(s) \leftarrow tmpFreq(s)/|(O)|$;
    If (($Freq(s) < minFreq$)) **Then**
      prune(s) ;
    End If
  End For
  $SPList$.add(F) ;
End While
return $SPList$;

Alg. 1: *ApSPoID* - Main algorithm.

For each candidate sequence, data sequences are parsed for finding the itemsets of the candidate sequence. The subfunction ***findSequence*** (Alg. 2) is a recursive function that parses a data sequence t for finding a candidate sequence s. When the searched itemset is found then ***findSequence*** searches for the next itemset of s in the data sequence t. Else it keeps on looking for the same itemset in the remainder part of s.

findSequence - **Input:** s, a candidate sequence ; t, a data sequence
d, table of appearance level for the itemsets of s ;
n, number of the currently searched itemset
Ouput: d

$sTail \leftarrow s \backslash s$.first() ; $tTail \leftarrow t \backslash t$.first() ;
If (($sTail \neq \varnothing$)&($tTail \neq \varnothing$)) **Then**
  If ($\min_{i \in s.\text{first}} \mu_{s.\text{first}}(i)$) **Then**
    $d[n] \leftarrow \min_{i \in s.\text{first}} \mu_{s.\text{first}}(i)$;
    findSequence($sTail$, $tTail$, d, $n+1$);
  Else
    findSequence(s, $tTail$, d, n);
  End If
Else
  return d ;
End If

Alg. 2: *findSequence* - data sequence parsing algorithm.

If the first itemset is found complete, the appearance level for the first itemset is 1, then the following itemset is searched. If the first itemset is partially found in an incomplete record, then appearance level for this itemset I is $\min_{i \in I} \mu_r(i)$, then the following part of the sequence is searched. On the other hand, if the record only partially supports the itemset and if the number of missing value des not allow to complete it, the first itemset is not found. Then it is searched in the remainder of the data sequence.
If the sequence is found, its appearance level is computed by an average agregation. If it is found several times, the best appearance level is kept. Once the whole database is parsed, the sum of appearance level is divided by the number of objects in database.
Then all the candidate sequences having a frequency lower than $minFreq$ are pruned.

The temporal complexity of this algorithm is the same as the one of the algorithm TotallyFuzzy presented by [9]. We use the same kind of optimizations to reduce the number of database scans. The *path* structure described for TotallyFuzzy keeps in memory the most certain occurrence of a sequence and each possible start while scanning the database.

V. Experiments

These experiments were carried out on a PC - Linux 2.6.7 OS, CPU 2.8 GHz with 512 MB of memory. The algorithm was implemented in Java on the PSP principle. In particular, the Prefix-Tree structure was used to store the candidate and frequent sequences.
We used synthetic datasets randomly generated by a normal distribution. Then some items were randomly replaced by missing values. Sequential patterns were extracted from the complete database and from the preprocessed incomplete ones (i.e. incomplete databases in which incomplete records have been deleted). Then those patterns are compared to the one discovered by our algorithm SPoID. Results here detailed were obtained from several synthetic datasets containing around 2000 sequences of 20 transactions in average. Each transaction contains around 10 items chosen among 100.

Our analysis is based on the several counting:

- the total number of sequential patterns discovered by SPoID,
- the number of sequential patterns discovered by SPoID, that are discovered in the complete database,
- the number of wrong sequential patterns discovered by SPoID (that groups together the patterns that are not discovered in the complete database and the one not found by SPoID but should be).

Table V sums up these notations.

A. ApSPoID vs. SPoID

Following figures give a comparison between patterns discovered by ApSPoID and those discovered by PSP after pre-processing (deletion of incomplete data) and also those discovered by SPoID ([14], an existing approach for mining incomplete datasets.

TABLE V
NOTATIONS FOR THE DIFFERENT KINDS OF PATTERNS.

β	# sequential patterns discovered by SPoID, also contained in the complete dataset
δ	# different sequential patterns
θ	# sequential patterns discovered by SPoID in the incomplete database
τ	# sequential patterns discovered in the complete dataset

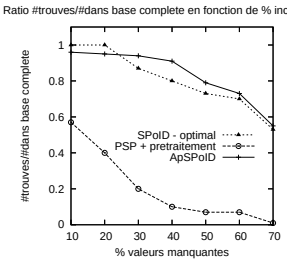
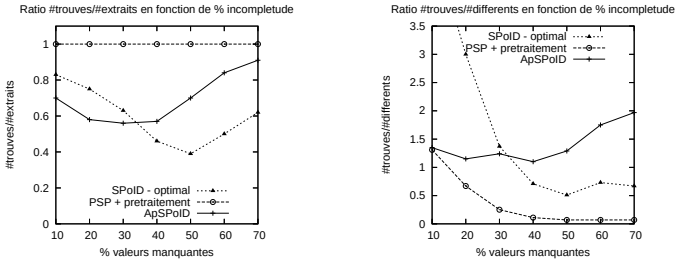


Fig. 1. β/τ rate according to rate of database incompleteness.

First, Figure 1 shows the evolution of the ratio β/τ . ApSPoID and SPoID have similar behaviors regarding the number of right discovered patterns. More especially, ApSPoID gives better results fronm 40% to 70% of incompleteness. On this interval, the results of SPoID are less interesting than those obtained on the range 10 to 20%, where SPoID discovers all the patterns of te complete database. It can also be noted that wathever the percentage of incompleteness, the pre-processing method is not powerful.



(a) (b)

Fig. 2. (b): β/θ rate according to rate of database incompleteness;(b): β/δ rate according to rate of database incompleteness.

Then figures 2(a) and 2(b) show a comparison of parasitic and missing patterns. Figure 2(a) presents the proportion of right patterns with respect to discovered patterns , according to the incompleteness percentage. This ratio is considered as good when it is close to 1 (each discovered patterns is also discovered in the complete database). However, these results should be compared to Figure 1. All the extacted patterns may indeed correspond to the complete database but an important amout of them may be missing. In our experiments, it is the case of PSP with pre-processing.

Note that ApSPoID results are better than those of SPoID from 40% of incompleteness. At this rate ApSPoID indeed discovers 90% of the patterns found in the complete database

and those patterns represent around 60% of discovered patterns.
Furthermore, when the database contains 50% of missing values, ApSPoID discovers 80% of the squential patterns of the complete database and those patterns represent 70% of discovered patterns while SPoID only find 70% of the complete database patetrns, only representing 40% of discovered patterns.

These observations are confirmed by Figure 2(b) that shows the evolution of $\beta/delta$. It shows that form 35% of incompleteness, the proportion of right patterns according to the different ones is clearly more important for ApSPoID than for SPoID. Moreover, the number of right pattern is still higher than the part of different patterns (this ratio should remain greater than 1), on the contrary to SPoID for which results are good until 30% of incomlpeteness and then collapse.

B. Choosing a distribution

We analyse the behavior of ApSPoID according to the frequency distribution for incomplete records. We show that it has an impact on obtained results.

We here present a comparison of ApSPoID performances used with two different distributions:

- One distribution is computed using the RAR algorithm (as in example, Section IV). It is used by ApSPoID - Dist 1 on Fig. 4.
- A second distribution is found by considering the whole set of records to compute the frequency. It is used by ApSPoID - Dist 2 on Fig. 4.

While the first distribution is rather optimistic (each missing value is ignored for computing the distribution), the second one is more pessimistic (each missing value is considered as different from the item of which the frequency is computed). Choosing one or the other has an impact on the final results.

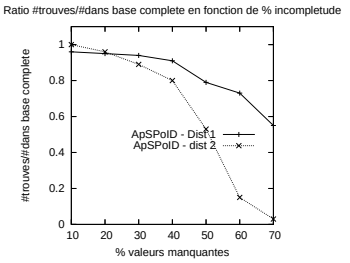


Fig. 3. β/τ rate according to rate of database incompleteness.

Fig. 3 shows that the second distribution does not allow ApSPoID to find the patterns of the complete database from 40% of incompleteness, whereas ApSPoID with the optimistic distribution finds an important amount of patterns until 60% of incompleteness. On the other hand, Fig. 4(a) shows that ApSPoID with the second distribution discovers less parasitic patterns.

From Fig. 4(b) we can conclude that the pessimistic distribution gives better results than the optimistic one until

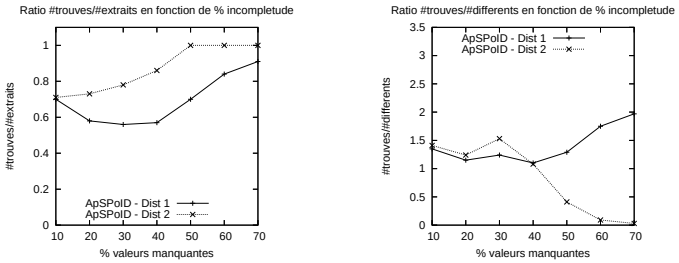


Fig. 4. (a): β/θ rate according to rate of database incompleteness;(b): β/δ rate according to rate of database incompleteness.

40% of incompleteness. On the range 20 to 40% of incomplete data, results obtained by ApSPoID with the second distribution are similar to those of SPoID.

Thus, choosing a distirbution more or less optimistic determines which quantity of sequential patterns found in the complete database will be retrieved by ApSPoID, but also the amount of parasitic patterns. A trade-off should then be made between the percentage of right patterns found and the percentage of parasitic patterns.

These experiments show that in general , ApSPoID is more reliable than SPoID from 30% of missing values and that these results are interesting until 60% of incomplete data. On the other hand patterns discovered by ApSPoID - Dist 1 include 80 to 90% of patterns contained in the complete database for an incompleteness rate between 10 to 40%. These results are almost as good as those of SPoID. However, using ApSPoID does not require to specify other parameter than $minFreq$. Its use is then easier than the use of SPoID. Furthermore, results obtained by ApSPoID could be improved by refining the frequency distribution at each iteration, considering more and more precisely the whole records. Lastly, regarding the comparison of runtime, ApSPoID behaves similarly to SPoID.

VI. Further Work

Sequential pattern discovery is an interesting data mining method while learning knowledge from large time-stamped databases, such as industrial process databases. But this kind of databases often contains missing values. However there is only one mining techniques for discovering frequent sequences from incomplete databases. Therefore in this paper we proposed to adapt the original definitions related to sequential pattern mining in order to handle random incompleteness in data sequences. These new definitions enable the user to manage missing values directly during the mining task, then avoiding an heavy preprocessing step. Our method and algorithm ApSPoID has been implemented and tested on synthetic datasets. We have thus shown the robustness of our approach until an incompleteness rate around 50% , while the existing approaches give erroneous results since 10% of missing values. We now work on extending this approach in

order to regard other kind of missing values (not randomly spread, for instance), after having detected the different kind of incomplete information. It seems also necessary to detect the attributes that should not be considered as incomplete but not existing. Lastly noise is also a common imperfection in real-life databases. It could then be interesting to consider it in a further version of our algorithm.

REFERENCES

[1] R. Agrawal and R. Srikant, "Mining sequential patterns," in *Eleventh International Conference on Data Engineering*. Taipei, Taiwan: IEEE Computer Society Press, 1995, pp. 3–14.

[2] J. Nayak and D. Cook, "Approximate association rule mining," in *Florida Artificial Intelligence Research Symposium*, 2001.

[3] V. Ng and J. Lee, "Quantitative association rules over incomplete data," in *IEEE International Conference*, 1998, pp. 2821–2826.

[4] F. Masseglia, F. Cathala, and P. Poncelet, "The PSP Approach for Mining Sequential Patterns," in *Principles of Data Mining and Knowledge Discovery*, 1998, pp. 176–184.

[5] R. Agrawal, T. Imielinski, and A. N. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*. Peter Buneman and Sushil Jajodia, 26–28 1993, pp. 207–216.

[6] F. Masseglia, P. Poncelet, and M. Teisseire, "Incremental mining of sequential patterns in large databases," LIRMM, France, Tech. Rep., 01 2000.

[7] T. Hong, K. Lin, and S. Wang, "Mining Fuzzy Sequential Patterns from Multiple-Items Transactions," in *Joint 9th IFSA World Congress and 20th NAFIPS Int. Conf.*, 2001, pp. 1317–1321.

[8] R.-S. Chen, G.-H. Tzeng, C.-C. Chen, and Y.-C. Hu, "Discovery of Fuzzy Sequential Patterns for Fuzzy Partitions in Quantitative Attributes," in *ACS/IEEE Int. Conf. on Computer Systems and Applications*, 2001, pp. 144–150.

[9] C. Fiot, A. Laurent, and M. Teisseire, "From crispness to fuzziness: Three algorithms for soft sequential pattern mining," *IEEE Transactions on Fuzzy Systems*, à paraître.

[10] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *5th International Conference on Extending Database Technology (EDBT '96)*. London, UK: Springer-Verlag, 1996, pp. 3–17.

[11] F. Masseglia, P. Poncelet, and M. Teisseire, "Pre-Processing Time Constraints for Efficiently Mining Generalized Sequential Patterns," in *11th Int. Symp. on Temporal Representation and Reasoning (TIME '04)*, 2004, pp. 87–95.

[12] C. Fiot, A. Laurent, and M. Teisseire, "Des motifs séquentiels généralisés aux contraintes de temps étendues," in *6èmes journées d'Extraction et Gestion des Connaissances*, 2006, pp. 603–614.

[13] A. Ragel and B. Cremilleux, "Treatment of missing values for association rules," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, 1998, pp. 258–270.

[14] C. Fiot, A. Laurent, and M. Teisseire, "Spoid: Extraction de motifs séquentiels pour les bases de données incomplètes." in *7èmes journées d'Extraction et Gestion des Connaissances*, 2007.