SPoID: Do Not Throw Meaningful Incomplete Sequences Away!

Céline Fiot, Anne Laurent, Maguelonne Teisseire

SPoID: Do Not Throw Meaningful Incomplete Sequences Away!

Céline Fiot
LIRMM - CNRS
161 rue Ada
34392 Montpellier, France
fiot@lirmm.fr

Anne Laurent
LIRMM - CNRS
161 rue Ada
34392 Montpellier, France
laurent@lirmm.fr

Maguelonne Teisseire
LIRMM - CNRS
161 rue Ada
34392 Montpellier, France
teisseire@lirmm.fr

Abstract

Industrial databases often contain a large amount of unfilled information. During the knowledge discovery process one processing step is often necessary in order to remove these incomplete data either by deleting or assessing them. When the data mining task consists in mining 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. Therefore we propose a method that uses the partial information contained in incomplete records, only temporary ignoring the missing part of the record. Experiments run on various synthetic datasets show the validity of our proposal as well in terms of quality as in terms of the robustness to the rate of missing values.

Keywords: Data mining, sequential patterns, missing values, incomplete data.

1 Introduction

For the last decade, data mining application fields have widened. Approaches which were designed for specific applications are now applied to other ones. Especially it is the case of sequential patterns. This data mining technique aims at discovering knowlegde from temporal databases. First designed to analyse customer behavior, they are now used in various industrial or biological fields. The searched datasets are sets of time-stamped records. Each of them is constituted of a set of values.

But sequential pattern approaches must be adapted, these kinds of data having different format or some of them being imperfect. In particular, industrial databases often contain incomplete data (records containing unfilled attributes) due to breakdowns or errors, for instance. However existing 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. For this reason it could be useful to implement a method for mining sequential patterns within incomplete databases.

In this paper we propose an extension of the principles originally developed by [2] in order to discover frequent sequences within databases containing values missing at random. This approach was designed on the basis of an association rule method for mining incomplete databases [14] and of a technique often used in machine learning [8]: ignoring missing values without ignoring the whole involved record. This principle consists in only making use of available information (i.e. filled-in attributes) and ignoring missing information. Thus only partial complete databases are mined for each pattern and the whole dataset is used to discover all the patterns.
We here redefine this principle for sequential pattern mining, thus proposing to discover maximal frequent sequences within time-stamped incomplete databases. For this reason, we adapt the concepts linked to sequential pattern discovery, leading to the SPoID (Sequential Patterns over Incomplete Data) algorithm. This algorithm uses a well-known and efficient sequential pattern mining algorithm, PSP [9]. It has been tested on synthetic datasets to show the validity of our approach.

The remainder of the paper is organized as follows. Section 2 introduces the methods for association rule mining in the presence of missing values, and the concepts linked to sequential pattern discovery. Section 3 details our approach to mine for sequential patterns within incomplete data, and we describe our algorithm in Section 4. Section 5 is then dedicated to experiments that show the validity of our approach. Finally,

we discuss in Section 6 further work opened by this proposal and we conclude in Section 7.

2 Sequential Patterns and Incompleteness

Sequential patterns are often introduced as an extension of association rules, initially proposed in [1]. 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.

2.1 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-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 ($minFreq$) [2]. 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 [10], to handle numerical and quantitative values [7, 3, 6] 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...) [15, 11, 5]. 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.

2.2 Motivations

Let us consider the database given by Table 1. The goal is to extract the sequential patterns with a minimum frequency equal to 50%.

Table 1: Complete database.

O.	Seq.
O1	(a b) (b c d) (b c e)
O2	(a) (b c) (b d)
O3	(a b) (b c) (b c d)

Table 2: Incomplete database.

O.	Seq.
O1	(a b) (? ? c) (? b c)
O2	(a) (? c) (b d)
O3	(a b) (? c) (? ? c)

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 2.

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.

During the pre-processing step, incomplete records may either be completely deleted, which leads to the dataset in Table 3, or be partially but definitively deleted (only missing values are removed), as shown

by Table 4.

Table 3: After incomplete record deletion.

O.	Seq.
O1	$(a\ b)$
O2	$(a)\ (b\ d)$
O3	$(a\ b)$

Table 4: After missing value deletion.

O.	Seq.
O1	$(a\ b)\ (c)\ (b\ c)$
O2	$(a)\ (c)\ (b\ d)$
O3	$(a\ b)\ (c)\ (c)$

Considering the first pre-processing, discovered pattern $minFreq = 50\%$ is: <(a b)>. In the best case, Table 4, 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.
As deleting missing values or incomplete records leads to an important loss of information, using the whole dataset without deletion appears essential. In Section 3, we thus propose an approach that uses partial and temporary disabling of incomplete data. All the records will then be used for discovering all the sequential patterns, but each of them is extracted from a partial dataset. This method has been instigated from an association rule approach.

2.3 Association Rules and Missing Values

Some works were proposed to discover association rules within incomplete databases. Especially, [12, 13] implement an assessment systems based on a probabilistic distribution. With this approach one missing value can represent various values while mining for association rules. These methods are particularly well-suited to classical non-historized relational databases, but they are not easily adaptable to the specific format of data sequence detailed in section 2.1.

That is the reason why we chose to start our approach from the RAR algorithm (Robust Association Rules), proposed by [14]. This method, fully compatible with the original proposal [1], 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 main idea consists in only taking into account filled-in attributes in incomplete records. The whole database is not used to discover each rule but to generate the whole set of rules. This technique is based on the valid database concept, which is a complete dataset for a given itemset. The remaining part of the database is temporary ignored.
In order to consider this dataset partitioning, definitions of support (percentage of records in database that include the rule items) and confidence (probability for a record to contain the right part of the rule knowing it contains the left part) were reformulated. Furthermore, a new concept was introduced to take into account the size of the complete sample used to compute the rule support. This representativity measure also allow to prune rules that are slightly significant with respect to the initial database.

From this principle we redefine in the next section the definitions linked to sequential pattern mining previously detailed. These new definitions are then used by our algorithm to mine incomplete sequence database.

3 SPoID: Dealing with Incomplete Data

3.1 Sequential Patterns over Incomplete Data (SPoID)

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, SPoID (Sequential Patterns over Incomplete Data), based on the principles of the RAR algorithm proposed by [14].

The main idea of our approach, as the one of the RAR method, is incomplete elements disabling, within our context, incomplete sequences. While the RAR algorithm only regards complete records for association rules mining, we propose to only take into account the complete data sequences for each candidate sequence. In other words, when an incomplete data sequence is scanned, only filled-in time-stamped attributes will be considered for frequency calculation. Thus each candidate sequence will be considered as a frequent sequence on a partial database, but the whole dataset will be used to find the whole set of frequent sequences.

Let us consider a candidate sequence S, the set $\mathcal{O}$ of objects in the database can be divided into three disjoint subsets (Figure 1):the set of data sequences supporting S, denoted by $\mathcal{O}_S$, the set of data sequences not supporting S, denoted by $\mathcal{O}_{\overline{S}}$, and the set of data sequences that we do not know if they support S or not, denoted by $\mathcal{O}^*_S$.

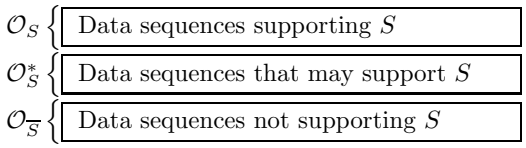

Figure 1: Partition of the database depending on S inclusion.

3.2 Definitions

For each candidate sequence S, only the subsets $\mathcal{O}_{\overline{S}} \cup \mathcal{O}_S$ will be kept to determine wether the sequence S is frequent or not. This data sequence set represents the *valid database* for S.

Definition 1. *A* valid database *is a database only containing complete data sequences for a given candidate sequence, i.e. each value of each record in the data sequence corresponds to an identified item i of $\mathcal{I}$, the set of items in the database.*

Constitution of a valid database leans on temporary disabling data sequences that contain missing values for items in the candidate sequence. This implies to redefine the frequency calculation to take into account the database partial deactivation.

Definition 2. *A data sequence is* disabled *for a candidate sequence S if it is incomplete for S (i.e. we cannot decide whether it supports S or not). The set of data sequence disabled for a candidate sequence S is denoted by $Dis(S)$.*

The frequency definition given in section 2.1 is then modified in order to consider the valid database concept, and thus that only one part of the dataset is used for frequency calculation.

Definition 3. *The* frequency *of a sequence S is the appearance rate of this sequence among the data sequences that can support it. It is defined as the ratio of the number of data sequences supporting S by the number of data sequences that surely include S or not (complete data sequences for S). It is given by:*

$$Supp(S) = \frac{|\mathcal{O}_S|}{|\mathcal{O}| - |Dis(S)|}$$

Property 1. *Considering minor restrictions, this definition holds the antimonotonicity property of the support definition enonciated by [2].*

Proof.

$$Supp(S) = \frac{|\mathcal{O}_S|}{|\mathcal{O}| - |Dis(S)|} = \frac{|\mathcal{O}_S|}{|\mathcal{O}_S| + |\mathcal{O}_{\overline{S}}|} = \frac{1}{1 + \frac{|\mathcal{O}_{\overline{S}}|}{|\mathcal{O}_S|}}$$

However, if $S' \subseteq S$, then $|\mathcal{O}_{S'}| \geq |\mathcal{O}_S|$. Indeed, if a sequence S' is supported by the data sequence of an object o, then either this object supports its supersequence S or it does not support it, or we do not know. But in any case, any object that does not support S' cannot include its supersequence S. Moreover, if $S' \subseteq S$, then $|\mathcal{O}_{\overline{S}}| \geq |\mathcal{O}_{\overline{S'}}|$. Indeed, $o \in \mathcal{O}_{\overline{S'}} \Rightarrow \exists i \backslash s'_i \not\subset o$, however $S' \subseteq S \Rightarrow \forall i, \exists s_k \backslash s'_i \subseteq s_k$, then $\exists k \backslash s_k \not\subset o$ and, in that case, o does not include S.
Considering that none of these cardinalities is null, both inequalities 3.2 and 3.2 can be multiplied member to member. That leading to:

$$\begin{aligned} |\mathcal{O}_{\overline{S}}||\mathcal{O}_{S'}| \geq |\mathcal{O}_S||\mathcal{O}_{\overline{S'}}| &\Rightarrow 1 + \frac{|\mathcal{O}_{\overline{S}}|}{|\mathcal{O}_S|} \geq 1 + \frac{|\mathcal{O}_{\overline{S'}}|}{|\mathcal{O}_{S'}|} \\ &\Rightarrow \frac{1}{1+\frac{|\mathcal{O}_{\overline{S'}}|}{|\mathcal{O}_{S'}|}} \geq \frac{1}{1+\frac{|\mathcal{O}_{\overline{S}}|}{|\mathcal{O}_S|}} \\ &\Rightarrow Supp(S') \geq Supp(S) \end{aligned}$$

Then, the support definition for valid databases holds the antimonotony property. □

As the new support definition is antimonotonic, we can use the various properties described in [2] in order to implement the sequential mining algorithm within incomplete databases. However, the *frequency* concept must also be regarded taking into account the size of the valid database used to compute it. Therefore we define a *representativity* criteria and a minimum representativity threshold *minRep*, that must be satisfied: a valid database must be a significative sample of the whole dataset for a sequence S to be frequent, even if the condition $freq(S) \geq minFreq$ holds.

Definition 4. *The* representativity *$Rep(S)$ of a sequence S is defined as the ratio of the number of data sequences including S or that cannot include it by the total number of data sequences in the whole dataset. It is given by:*

$$Rep(S) = \frac{|\mathcal{O}| - |Dis(S)|}{|\mathcal{O}|}$$

Definition 5. *A sequence is said to be* representative *if its representativity is greater than a minimum representativity value minRep.*

In other words, to be kept as frequent, a candidate sequence must have a representativity greater than the minimum representativity threshold *minRep* and its frequency must be no less than the user-defined minimum threshold $minFreq$.

3.3 Representativity Threshold and Margin of Error

Statistics use sampling techniques that allow to only consider a population subset to assess a proportion, satisfying an error interval with a sufficient confidence. These tools help to determine the optimal sampling size depending on the data distribution. Thus considering a random data distribution, [16] uses the Chernoff bound to set the minimal size of a random sample for association rule mining. This result was also proved theoretically and experimentally by [17].

We thus propose to use two kinds of representativity depending on the user needs: the minimum representativity threshold can be defined either by the

user as a percentage of the dataset size, or it can be an absolute number of data sequences computed from statistics formula related to the data distribution and user-defined parameters for error and confidence level. However our experiments show that the optimal representativity threshold is not an absolute value but rather depends on the missing value rate of datasets.

4 Implementation

4.1 An Example

The incomplete database used in this example is given by Table 2. Let $minFreq$ be 50%. First support and representativity are computed for each item to determine which one are frequent. Item a is certainly supported by the three objects, then its frequency is $freq(a) = 3/3 = 100\%$ and its represetativity is equal to 1. It is the same for items b and c.
For item d, $\mathcal{O}_{<d>} = \{O2\}$ and $Dis(< d >) = \{O1, O3\}$, then $freq(d) = 1/(3\text{-}2) = 1$ and $rep(d) = (3\text{-}2)/3 = 0.33$. If $minRep$ is 0.3, then $rep(d) > minRep$ and d is a frequent item. On the other hand, if $minRep = 0.4$, then $rep(d) < minRep$ and d is not a frequent item because the valid database regarded to compute its frequency is not significant enough.

Let us consider $minRep$=0.3. Now we consider the candidate sequence $S =< (a\ b)(a\ b\ c\ d) >$. This sequence cannot be supported by one of the data sequence, because none of them contains an itemset composed of 4 items, either complete or not. Then $\mathcal{O}_S = \{\}$, $Dis(S) = \{\}$ and $\mathcal{O}_{\overline{S}} = \{O1, O2, O3\}$ and $freq(S)$=0. S is not frequent. Now we consider $S' =< (a\ b)(b\ c) >$. It is supported by $O1$, it cannot be supported by $O2$ but maybe by $O3$. Then, $\mathcal{O}_{S'} = \{O1\}$, $Dis(S') = \{O3\}$ and $\mathcal{O}_{\overline{S'}} = \{O2\}$. That leads to $freq(S')$=1/(3-1)=50% and $rep(S')$=(3-1)/3 = 0.67. S' is then both representative and frequent.

Applying this method, discovered patterns for $minFreq$=50% and $minRep$ = 0.3 are: <(a b)(c)(b c)> and <(a)(c)(b d)>. 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 5 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.

4.2 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 is described by the algorithm Alg. 1: for each candidate sequence, for each object,

- if the candidate sequence is found, the absolute value of the frequency is incremented,
- if the candidate sequence is not found nor a sequence with missing values that could be replaced to complete the candidate sequence, then the object does not support the candidate sequence. The absolute frequency is not incremented.
- an incomplete data sequence in which missing values could be replaced by items of the candidate sequence is found. In that case, the object is added to the disabled object set.

Once the whole dataset scanned, the absolute value of the frequency is divided by the substraction of the number of disabled objects to the number of objects in the database. The representativity is also computed. Then the pruning step is run to delete candidate sequences that are neither frequent nor representative.

SPoID - **Input:** $|\mathcal{O}|$, sequence database,
$minSup$, minimum support
$minRep$, minimum representativity
(user-defined or computed)
Ouput: $SPList$, frequent sequence list

$C \leftarrow \{i \in \mathcal{I}\}$; $k = 1$;
$F \leftarrow$ getFrepnRep(C,$minFreq$,$minRep$) ;
$SPList$.add(F) ;
While ($C \neq \varnothing$) **do**
  k++ ; $C \leftarrow$ generate(F,k) ;
  For each candidate sequence $s \in C$ **do**
    For each object $o \in \mathcal{O}$ **do**
      [Search for s within S_o]
      If ($s \in S_o$) **Then**
        $support(s)$++ ;
        $Dis(s) \leftarrow Dis(s) \backslash$o ;
      Else
        If ($\tilde{s} \in S_o / \tilde{s}$ may be s) **Then**
          $Dis(s) \leftarrow Dis(s) \cup o$;
        End If
      End If
    End For
    $Sup(s) \leftarrow support(s)/(|\mathcal{O}| - |Dis(s)|)$;
    $Rep(s) \leftarrow |\mathcal{O}| - |Dis(s)|/|\mathcal{O}|$;
    If (($Sup(s) < minSup$)||
      ($Rep(s) < minRep$)) **Then**
      prune(s) ;
    End If
  End For
End While
return SPList ;

Algorithm 1 – *SPoID - main algorithm.*

The temporal complexity of this algorithm is, in the worse case, the same as the one of the algorithm TOTALLYFUZZY presented by [6]. We use the same kind of optimizations to reduce the number of database scans. On the other hand, the space complexity is much less than the one of TOTALLYFUZZY, as it is similar to the one of PSP.

5 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. 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 5 sums up these notations.

Table 5: 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

First, Figure 2 shows the evolution of the ratio β/θ, with respect to the minimum representativity threshold. It can be noted that this rate increases according to *minRep*. It means that among sequential patterns discovered by SPoID, the proportion of sequential patterns obtained on the complete dataset increases with the *minRep* threshold.

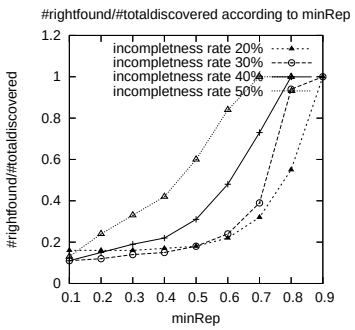

Figure 2: β/θ rate according to *minRep*.

This observation can be completed by analysing Figure 3, which represents evolution of the ratio β/τ (number of discovered sequential patterns with respect to sequential patterns that should be discovered) according to the minimum representativity threshold. This ratio decreases while *minRep* increases. It means that the minimum representativity threshold should be low enough to allow the discovery of all the sequential patterns obtained in the complete database.

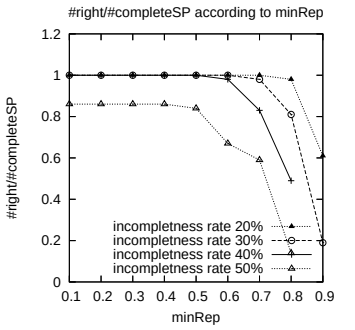

Figure 3: β/τ rate according to *minRep*.

Then we show that there exists an optimal value of the representativity threshold for which the ratio β/θ and β/τ are the closest to 1. This value is the threshold at which the right patterns discovered in the incomplete database are the most numerous compared to the number of wrong patterns. Figure 4 focuses on this optimal value *minRep*. This graph describes the evolution of the ratio β/δ according to the minimum representativity. It can be noted that there is not an absolute value for the minimum representativity, that would be common to every database independently from the incompleteness rate and only depending on an error margin. From these results, the minimum representativity threshold only depends on the incompleteness rate of the database.
Whatever the proportion of missing values in the incomplete database, the overall behavior of the ratio

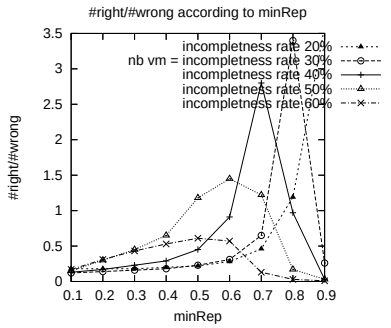

Figure 4: β/δ rate according to *minRep*.

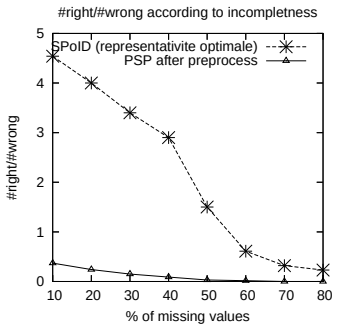

Figure 5: β/δ rate according to the missing value proportion.

β/δ is similar: it increases until it reaches a maximum before decreasing. This maximal point corresponds to the average optimal representativity, for which the number of right patterns discoverd by SPoID is the highest and the number of wrong patterns is the lowest. Table 6 gives the value of optimal representativity empirically found, for each incompleteness rate in datasets.

Table 6: Average optimal representativity according to the missing value proportion in the database.

% of missing values	**optimal** *minRep*
10%	0.97
20%	0.9
30%	0.81
40%	0.74
50%	0.6
60%	0.48
70%	0.39
80%	0.22

Figure 4 also shows the SPoID algorithm performances depending on the missing value proportion in the database. A difference can be noted between the overall evolution of the ratio β/δ for databases containing less than 40% of missing values and the one containing 50% of incomplete records or more.
Thus Figure 5 gives the comparison of SPoID success rate with results obtained on a preprocessed incomplete database. This figure shows that some right patterns, discovered by SPoID are not found using a deletion preprocessing step. It can be noted that the ratio of right patterns strongly decreases between 40 and 50% of missing values: the number of wrong patterns becomes proportionnaly slightly higher compared to the number of right patterns.
It can also be noted that this ratio becomes less than 1 when the missing value percentage exceeds 50%. SPoID can then discover sequential patterns within incomplete database if at least half of the records in the dataset are complete, while the former methods requiring a preprocessing step do not find all the frequent patterns since 10% of missing values.

Lastly, the analysis of runtime performances shows that at constant incompleteness rate, runtime of SPoID is slightly constant while the *minRep* threshold decreases.The qualitative analysis of corresponding candidate and frequent sequences has shown that the number of candidate sequences increases but, as the minimum representativity is lower, the time spent for scanning the database decreases. We also noted that runtime increases with the incompleteness rate.

6 Further Work

Results detailed in these experiments show that the method SPoID is robust until around 40% of incompleteness. But these interesting results could be improved using another approach. Indeed, as we introduced it in section 2.3, some work were done for mining association rules using probabilities to assess missing values. These proposals [12, 13] cannot easily be adapted for mining sequences. However we are currently working on using a frequency distribution for modelizing incomplete data.
This frequency distribution is not a probability distribution and does not have the properties of a possibility distribution. We rather designed it as a fuzzy set, of which the membership function represents the level of certainty of a missing value to be for each possible value present in the database. Some experiments are currently carried out. First results are available in [4]. Next step will then consist in detecting the different kinds of incomplete information including the attributes that should not be considered as incomplete even if unfilled. Lastly we think about regarding noise in a further version of our algorithm, as it is a common imperfection in real-life databases.

7 Conclusion

Temporal databases available from many fields such as biological data or industrial process data most of the time contain imperfect data. More especially they may contain a lot of incomplete data. But the most adapted data mining technique to analyse such time-stamped datasets, i.e. sequential pattern discovery, cannot be easily applied to incomplete data. There is indeed no mining technique for discovering frequent sequences from incomplete databases. Therefore in this paper we proposed new definitions for 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 a heavy preprocessing step. Our method and algorithm SPoID has been implemented and tested on synthetic datasets. We have thus shown the robustness of our approach until an incompleteness rate around 40% , while the existing approaches give erroneous results since 10% of missing values. We now work on extending this approach in order to take into account other types of missing values such as data not randomly spread.

References

[1] 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*, pages 207–216. Peter Buneman and Sushil Jajodia, 26–28 1993.

[2] R. Agrawal and R. Srikant. Mining sequential patterns. In *Eleventh International Conference on Data Engineering*, pages 3–14, Taipei, Taiwan, 1995. IEEE Computer Society Press.

[3] 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*, pages 144–150, 2001.

[4] C. Fiot. Approximate sequential patterns for incomplete databases. Technical Report RR-07003, LIRMM, France, 02 2007.

[5] 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*, pages 603–614, 2006.

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

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

[8] W. Z. Liu, A. P. White, S. G. Thompson, and M. A. Bramer. Techniques for dealing with missing values in classification. *Lecture Notes in Computer Science*, 1280, 1997.

[9] F. Masseglia, F. Cathala, and P. Poncelet. The PSP Approach for Mining Sequential Patterns. In *Principles of Data Mining and Knowledge Discovery*, pages 176–184, 1998.

[10] F. Masseglia, P. Poncelet, and M. Teisseire. Incremental mining of sequential patterns in large databases. Technical report, LIRMM, France, 01 2000.

[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)*, pages 87–95, 2004.

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

[13] V. Ng and J. Lee. Quantitative association rules over incomplete data. In *IEEE International Conference*, pages 2821–2826, 1998.

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

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

[16] H. Toivonen. Sampling large databases for association rules. In *VLDB '96: Proceedings of the 22th International Conference on Very Large Data Bases*, pages 134–145, 1996.

[17] M. J. Zaki, S. Parthasarathy, W. Li, and M. Ogihara. Evaluation of sampling for data mining of association rules. Technical report, Rochester, NY, USA, 1996.