International Journal of Advanced Technology and Engineering Exploration, Vol 12(123)
ISSN (Print): 2394-5443 ISSN (Online): 2394-7454
http://dx.doi.org/10.19101/IJATEE.2024.111101058

Research Article

Efficient sequential rule mining in uncertain sequence databases

Imane Seddiki[1*], Farid Nouioua[1,2] and Abdelbasset Barkat[3]
Department of Computer Science, Mohamed El Bachir El Ibrahimi University, Bordj Bou Arreridj 34000, Algeria[1]
LIS UMR-CNRS 7020, Aix-Marseille University, Marseille, France[2]
Laboratory of Informatics and its Applications, Faculty of Mathematics and Computer Science, University of M'sila, M'sila 28000, Algeria[3]

Received: 18-June-2024; Revised: 18-February-2025; Accepted: 20-February-2025

Abstract
As data becomes a crucial resource for powering various real-world applications, the field of data mining encounters numerous challenges, particularly regarding storage and real-time processing. Mining association rules to uncover relationships and patterns in large datasets is a crucial technique. However, the inherent uncertainty and incompleteness of data pose significant difficulties for traditional mining algorithms. To tackle these challenges, a novel method is proposed for mining sequential rules from uncertain sequence databases (SDs). This method involves two primary steps: first, extracting a set of probabilistic rules, and second, filtering these rules based on the sequential information within the data. This approach effectively addresses data uncertainty and incompleteness, enabling the extraction of meaningful sequential rules that are otherwise difficult to identify using conventional methods. This innovative method enhances the capability of mining algorithms to handle uncertain data, offering a robust solution for real-time data processing as well as storage issues in various applications. Experimental results demonstrate the algorithm's efficiency and scalability on both synthetic and real-world datasets. The proposed method achieved superior runtime and memory efficiency as dataset sizes increase.

Keywords
Association rule, Probabilistic database, Sequences database, Sequential rule, Uncertain data.

1.Introduction
Association rule mining is a powerful technique in data mining demonstrating its effectiveness in various contexts, particularly in modelling customer behaviour by analysing market-basket databases that represent customer purchases. In this scenario, each association rule indicates how certain purchases are related to others [1, 2]. This technique was extended to sequential databases, which consist of a series of sequences, where events are required to follow a specific partial order, this type of databases are used in various fields, such as bioinformatics for storing deoxyribonucleic acid (DNA) or protein sequences and marketing for tracking customer purchase sequences [3, 4]. Sequential rules are patterns extracted from the original sequence database (SD), highlighting relevant associations between subsequences of events. These rules are then used to identify and predict the occurrence of specific subsequences within a set of items [5].

In many real-world applications, data might be incomplete or mislabelled. For instance, data collected from wireless sensor networks may be uncertain due to external interference or sensor aging, complicating the extraction of association or sequential rules, and requiring special methods to handle the uncertainty in data. This uncertainty is often represented as an existential probability, which can be associated with each sequence or each item within a sequence [4, 6–8]. Existing techniques for mining sequential rules face limitations in addressing data uncertainty, as the prevalence of such data is increasing daily, making the extraction of sequential rules from such data a critical challenge in many real-world applications, this growth highlights the importance of developing robust methods to handle uncertain data [9–11].

The primary objectives of this paper are to develop a method for mining sequential rules from uncertain SDs and to demonstrate its effectiveness in handling data uncertainty. This paper contributes by proposing a novel approach for extracting probabilistic rules

*Author for correspondence

from uncertain SDs and filtering these rules based on the sequential information present in the data. This approach enhances the capability of mining algorithms to process uncertain sequences more effectively.

The paper is organized as follows: Section 2 reviews related work and highlights existing challenges. Section 3 provides background information and a problem statement to help the reader understand the proposed method in detail. Section 4 presents the experimental results and discussion. Finally, Section 5 provides concluding remarks and discusses potential future work.

2.Literature review

In the rapidly evolving field of data mining, handling uncertainty has emerged as a critical challenge for researchers. Real-world datasets frequently contain missing, noisy, or imprecise information, which limits the effectiveness of traditional mining techniques. To address this issue, researchers have increasingly focused on extracting meaningful sequential rules from uncertain data. This involves carefully modelling and computing the probabilities of item occurrences and sequence patterns to ensure accurate and reliable results.

In this section, the primary related work on mining sequential rules from uncertain data is reviewed. Upon thoroughly examining the literature, the work is categorized into two main areas: the first focuses on finding frequent itemsets from probabilistic data [12–14], and the second addresses mining sequential patterns and rules from sequential data [15, 16]. The first category is further divided into three subcategories based on the method used to find frequent item sets: 1) using the concept of expected support, 2) employing exact methods to compute the exact probability of pattern frequencies, and 3) relying on approximation models [17]. In the following, a set of papers is carefully selected for each category and subcategory. These papers are then analyzed and discussed in detail to provide a comprehensive understanding of their contributions and relevance.

Within the first subcategory, which used expected support as a primary metric, [18] suggested that all proposed algorithms for mining frequent itemsets from uncertain databases relied on support-based constraints to reduce the search space. However, this approach was insufficient, as frequent itemsets could exhibit weak affinity. To address this limitation, a new approach called weighted uncertain interesting pattern mining (WUIPM) was proposed. This method utilized the concept of expected support to identify frequent itemsets and introduced a novel tree structure, the weighted uncertain interesting pattern tree (WUIP-tree), where each item was assigned, a weight representing its importance. Additionally, a new measure called weighted expected support confidence (wUConf) was introduced to mine significant patterns. They claimed that WUIPM outperformed other algorithms. The challenge of extracting frequent patterns and association rules from uncertain data, particularly within probabilistic databases (PDB) that operate under the possible world semantics (PWS) framework, was addressed in [7]. Traditional mining methods struggled due to the exponential number of possible worlds such databases could generate. To overcome this limitation, two efficient algorithms, p-Apriori (a bottom-up approach) and TOp-Down Inheritance of Support pmf (TODIS), were proposed. These methods were designed to compute frequent patterns with probabilistic guarantees and could be easily extended to handle maximal frequent patterns and probabilistic association rules. Bernecker et al. [19] proposed a new approach that utilizes a probability distribution function with two models: the Poisson and normal distribution, classified under the first category that relies on approximation models. Their objective was to identify itemsets with frequentness probabilities above a threshold and to mine itemsets with the highest frequentness probabilities in both tuple and attribute uncertainty models. They defined the concept of frequency probability and used expected support along with the two probability distributions to efficiently approximate itemset support. Their results indicated that models based on expected support and Poisson approximation were easy to implement and compute efficiently due to pruning techniques, whereas the normal approximation model required more effort and did not allow pruning. An algorithm called probabilistic frequent pattern growth (ProFPGrowth) was introduced, based on the classical frequent pattern growth (FP-Growth) algorithm, to mine a PDB and extract probabilistic frequent itemsets [8]. Additionally, the ProFP-Tree was introduced, transforming the probabilistic database into a tree structure to facilitate information retrieval and minimize database access. The identification of probabilistic frequent itemsets was based on generating functions, which reduced computational complexity to O(N).

Under the second category, which focuses on mining sequential patterns and rules from sequential data, several key approaches have been considered.

The common sequential rules (CMRules) algorithm [4] is widely used in applications such as stock market analysis, weather observation, drought management, and e-learning. This algorithm converts a SD into a transaction database (TDB) to optimize the search space, then extracts association rules that meet minimum support and confidence thresholds. These rules are subsequently filtered to retain only those that respect the original sequential ordering. The authors evaluated CMRules in terms of time complexity, performance on real datasets, and its application in an intelligent agent aboard the International Space Station, demonstrating promising results across various contexts. In [20], the problem of high-utility sequential rule mining (HUSRM) was addressed, focusing on discovering high-utility sequential rules (HUSRs) from SDs based on minimum utility (minutil) and minimum confidence (minconf) thresholds. The objective was to efficiently identify these rules to support decision-making in various applications. To tackle this challenge, the authors proposed a novel approach called utility sequential rule algorithm (US-Rule), incorporating several optimization strategies. These strategies included early removal of unpromising items and rules, utilization of bitmap and compact utility-tables for enhanced memory efficiency, and the application of tighter upper bounds for rule expansion. The US-Rule algorithm was evaluated on multiple datasets, outperforming existing HUSRM methods, especially on dense and long sequences. It improved efficiency, memory usage, and scalability. However, its effectiveness varied by dataset type, requiring careful selection and tuning for optimal performance.

Wu et al. [21] addressed the inefficiency of existing methods in mining frequent order-preserving patterns (OPPs) from time series data, particularly in non-stationary data, where patterns repeat with different mean values. To overcome this limitation, they proposed an enhanced algorithm, efficient frequent order-preserving pattern miner (EFO-Miner), which leverages techniques such as pattern fusion, support-based pattern fusion, screening, and pruning to improve efficiency. Additionally, they introduced the order-preserving rule miner (OPR-Miner) to discover strong rules and mine relationships between OPPs. The study introduced EFO-Miner and OPR-Miner, which enhance efficiency, scalability, and rule mining effectiveness. EFO-Miner outperformed existing methods in speed, while OPR-Miner produced fewer but more meaningful rules. However, challenges remain in implementation complexity, scalability, and generalization across diverse time series data. In a recent study [22], researchers addressed the limitations of partially-ordered sequential rule mining by introducing methods for mining high-utility totally-ordered sequential rules (HTSRs). Traditional SRM and HUSRM approaches focus on partially-ordered rules, often ignoring the strict order of events, which can lead to inaccuracies in applications where sequence order is crucial, such as medical diagnoses, market predictions, and recommendation systems. To bridge this gap, they proposed two algorithms: totally-ordered sequential rules (TotalSR) and its optimized version, TotalSR+, designed to efficiently extract HTSRs from SDs. TotalSR introduces advanced techniques like utility tables, prefix sum lists, left-first expansion, and anti-monotonic pruning for efficient sequential rule mining. An auxiliary antecedent record table enhances performance, reducing execution time and improving scalability. TotalSR and TotalSR+ outperform existing methods by capturing strictly ordered rules and optimizing computational efficiency. While requiring dataset-specific tuning, these algorithms significantly advance sequential rule mining by improving accuracy, scalability, and efficiency.

In [4], targeted sequential pattern mining (TaSPM) was introduced as an algorithm designed to improve sequential pattern mining (SPM) by focusing on specific query sequences. Traditional SPM methods generate excessive, often irrelevant patterns, while existing targeted mining approaches lack generality and are constrained by specific application scenarios. TaSPM employed four innovative pruning strategies: unpromising transaction filter pruning (UTFP), unpromising prefix item pruning (UPIP), unpromising s-extension item pruning (USIP), and unpromising i-extension item pruning (UIIP) to eliminate unnecessary operations and enhance efficiency. A post-processing technique further optimizes the mining process.

Experiments on real and synthetic datasets demonstrate that TaSPM significantly outperforms baseline SPM algorithms in runtime, memory efficiency, and scalability, reducing redundant outputs while improving pattern relevance. However, its performance depends on predefined support thresholds and may face increased complexity with highly specific queries. *Table 1* provides a

comparative summary of key existing studies related to mining sequential rules from uncertain data, highlighting the most important characteristics of each work, such as the type of data used, algorithms, and datasets employed.

Table 1 A comparative summary of rule mining related work

Source	Works	Type of data	Target	Dataset	Algorithm(s)	Programming language
[4]	OPR-Miner	Time series data	Frequent OPPs	The stock and oil datasets	Efficient frequent OPP miner	C++
[5]	CMRules	Sequences	Sequential rules	Kosarak dataset	Apriori	Unknown
[7]	p-Apriori and TODIS	Probabilistic	Probabilistic association rules	data generator the accidents dataset (IBM)	Apriori with dynamic programming	C++
[8]	ProFP-Growth	Probabilistic	Probabilistic frequent patterns	artificial dataset (the accidents dataset)	FP-Growth	Unknown
[20]	WUIPM algorithm	Probabilistic	Frequent itemsets	Frequent itemset mining implementations dataset repository	Expected support WUIP-tree structure	C/C++
[21]	Approximation model	Probabilistic	Frequent itemsets	accidents T10I4D100k IBM available at http://fimi.uantwerpen.be/data/	Poisson and normal distribution	C
[22]	US-Rule	SDs	Sequential rules	Bible, Kosarak, Leviathan, Sign, Syn10k and Syn20k	US-Rule algorithms	Java
[23]	Equivalence class based sequential rule miner (ERMiner)	Sequences	Sequential association rules	Sign, Snake, FIFA BMS and Kosarak10k	Equivalence Classes	Java
[24]	Temporal probabilistic rules (TP-rules)	Relational	Association rules	Boko Haram Bdataset	Equivalence Classes	unknown
[25]	Common sequential Pattern Mining (CM-SPAM) algorithm	Sequences	Frequent sequential pattern	China pollution data PM2.5	SPM	unknown
[26]	Context-based association rule mining	Conventional data	Association rules	Online energy data portals	Context based association rule mining	Python
[27]	Top-K uncertain frequent pattern algorithm	Probabilistic	Top-k frequent patterns	Chainstore, Foodmart, Retail and T10I4D100K	Top-K uncertain frequent pattern algorithm	C
[28]	Weighted	Sequences	Frequent	**SIGN**: A dataset	Maximum Possible	python

Source	Works	Type of data	Target	Dataset	Algorithm(s)	Programming language
	frequent sequential pattern mining(WFSPM) framework		sequential pattern	of sign language utterances. **LEVIATHAN**: A conversion of the novel Leviathan by Thomas Hobbes (1651) as a SDs (each word is an item).	Weighted Support pruning	

The problem of mining uncertain sequences has been extensively studied, with numerous research efforts dedicated to addressing its challenges. Several comprehensive surveys have synthesized and evaluated the existing body of research, providing a broad overview of methodologies, findings, and emerging trends in the field. Similarly, studies featuring reproducible frameworks have also been discussed [29–33].

3.Materials and methods

This section introduces and defines fundamental concepts essential to this work, categorized by the type of data: transactional, sequential, and uncertain, while presenting the theoretical foundations and definitions of the specific problem addressed in this paper.

3.1Transactional data

Association rule mining is a significant and extensively studied method within the field of data mining. Its objective is to discover compelling correlations, frequent patterns, associations, or causal relationships among item sets in TDBs and other data stores [6, 34, 35].

A TDB is defined as a set of transactions, where each transaction is a set of items, formally: $TDB = \{t_1, t_2, t_3...t_n\}$ where $t_x \subseteq I$ is a transaction and $I = \{i_1, i_2, i_3....i_m\}$ is the set of all items.
An association rule between X and Y such *that* $X, Y \subseteq I$ and $X \cap Y = \emptyset$ is denoted by $X \rightarrow Y$. Recall that the support of a subset of items $X \subseteq I$ is defined as the number of transactions that contain this subset as shown in Equation 1.

$$Supp(X) = Card\{t \in TDB : X \subseteq t\} \quad (1)$$

Where Card refers to the cardinality of a set, which measures the number of elements in the set or describes its size in terms of its elements. The support of a rule X → Y is defined as the support of the union of the left side and the right side of the rule as shown in Equation 2.

$$Supp(X \rightarrow Y) = Supp(X \sqcup Y) \quad (2)$$

The confidence of a rule is defined as the ratio between its support and the support of the left side of the rule as shown in Equation 3.

$$Conf(X \rightarrow Y) = Supp(X \sqcup Y)/Supp(X) \quad (3)$$

An association rule $X \rightarrow Y$ is valid if and only if:
1. $Supp(X \rightarrow Y) \geq minSup$
2. $Conf(X \rightarrow Y) \geq minConf$

Where minSup (respectively, minConf) represents the minimum support and confidence threshold, respectively.

3.2Sequential data

A SDs is a database that contains a set of sequences where each sequence is constituted from n ordered set of transactions [36, 37]. Formally, let $I = \{i_1, i_2,...,i_n\}$ be a set of items. A SDB is defined as a set of sequences $S = \{s_1, s_2,...,s_n\}$ where each sequence s_x is an ordered list of transactions (itemsets): $s_x = \{X_1,X_2,...,X_m\}$ such that $X_1, X_2,...,X_n \subseteq I$ [1]. *Table 2* shows a sample of a SDB containing five sequences.

Table 1 A sequence database

Sequence ID	Sequence Content
Sequence 1:	[{e,f},{d},{a},{b}]
Sequence 2:	[{a,b},{d},{e},{c}]
Sequence 3:	[{f},{c,d}]
Sequence 4:	[{c,d},{f},{a,e}]
Sequence 5:	[{d},{e},{b,c},{a}]

Sequential rules emerge due to the unique characteristics of SDBs. They define a new type of rule, introduced in [4, 38], where a sequential rule $X \rightarrow Y$ denotes a relationship between two itemsets X and Y with no common items $(X \cap Y = \emptyset)$. This rule suggests that if items in X appear in some transactions within a sequence, items in Y will appear subsequently in the same sequence, following all items of X. It's important to note that the items in X and Y are not required to appear in the same

transaction within the sequence and their order is not constrained. To identify relevant sequential rules, researchers adapted two measures commonly used in finding association rules. The first measure, called sequential support, is computed as the number of sequences where all items in *X* precede all items in *Y* (denoted by *Supp(X•Y)* in Equation 4) divided by the total number of sequences [39].

$$SeqSupp(X \rightarrow y) = Supp(X \bullet Y)/Card(S) \quad (4)$$

The second measure, called sequential confidence, as shown in Equation 5, is calculated as the ratio of the number of sequences in which all items in X precede all items in Y to the number of sequences in which X appears [39].

$$SeqConf(X \rightarrow Y) = Supp(X \bullet Y)/Supp(X) \quad (5)$$

3.3Uncertain data

According to [40, 41], a PDB involves two types of uncertainty:

- Tuple-level uncertainty: In this model, each tuple in the database includes an existential probability attribute. This approach is noted for its straightforward design and query process. *Table 3* illustrates an instance of such a database [7].
- Attribute-level uncertainty: This form of uncertainty pertains to the uncertainty surrounding the values of transaction attributes. Here, each attribute carries its own existential probability value, indicating the level of certainty associated with its value. *Table 4* provides an example of this database type [8].

Table 3 A probabilistic database example (Tuple level uncertainty)

ID	Location	Time	Speed	Trac	Weather	Probability
	X	8-9pm	30-40	High	Rain	0.1
	X	7-8am	80-90	Low	Null	1.0
t_1 t_2 t_3 t_4 t_5	X	8-9pm	80-90	Low	Foggy	0.5
	X	8-9pm	30-40	high	Rain	0.2
	Y	2-3pm	50-60	low	Sunny	1.0

Table 4 A probabilistic database example (Attribute level uncertainty)

TID	Transaction content
1	(A, 1.0) (B, 0.2), (C, 0.5)
2	(A, 0.1), (D, 1.0))
3	(A, 1.0), (B, 1.0), (C, 1.0), (D, 0.4)
4	(A, 1.0), (B, 1.0), (D, 0.5)
5	(B, 0.1), (C, 1.0)

3.4Problem definition

This paper focuses on discovering sequential rules from SDBs that include uncertain data. The uncertainty specifically pertains to the attribute level, where it is represented by existential probability values associated with items.

A probabilistic sequence database (PSDB) is defined by a set of sequences $S=\{s_1, s_{2,\ldots},s_n\}$, where each sequence s_x consists of an ordered list of transactions (itemsets), $s_x=\{X_1, X_{2,\ldots},X_m\}$, with each $X_i \subseteq I$ representing a transaction containing items from the set $I=\{i_1, i_{2,\ldots},i_n\}$ [42, 43].

In this database, transactions may include items that are uncertain, where uncertainty is characterized by existential probabilities such that $0 <= p\ (item) <= 1$, where an item is considered certain if its existential probability is equal to 1.

Table 5 represents a PSDB comprising eight sequences, where each sequence includes both certain and uncertain items. This PSDB serves as a running example in the whole the paper. The goal is to identify a set of sequential rules that are both strong and useful. Specifically, the focus is on rules that are sufficiently probable and adhere to the sequential order observed in sequences within the PSDB. Various algorithms exist for extracting rules from different data types: association rules from transactional data, sequential rules from sequential data, and probabilistic rules from uncertain data. However, this paper aims to integrate these approaches to derive rules from uncertain and sequential data, referred to as probabilistic sequential rules.

Table 5 Example of PSDB

Sequence ID	Sequence Content (items)
1	(Z, 0.2), (W, 0.6)
2	(X, 1.0), (Y, 1.0), (Z,1.0), (W, 0.4)
3	(X, 1.0), (Y, 0.3), (Z,0.8)
4	(X, 0.4), (Y, 0.7)
5	(Y, 0.4), (Z, 1.0))
6	(X, 0.1), (Y, 1.0), (Z,1.0)
7	(X, 1.0), (Y, 1.0), (W,1.0)
8	(Y, 0.3), (Z, 1.0)

3.5Proposed approach

In this work, the following assumptions are made:

- The data is stored in a PSDB.
- Each item is a singleton item, meaning there are no nested sequences.
- Each item within a sequence has an existential probability ranging between 0 and 1 (items with a probability of 0 do not occur and are, therefore, absent from the database).

The keys for our approach are as under:

- To handle data uncertainty, the expected support measure is used to assess the probabilistic frequency of subsequences and sequential rules.
- Considering the ordering constraints inherent in sequences, the methodology operates in two phases: first, ignoring temporal information to generate a set of classical probabilistic rules, and then filtering these rules in the second phase to ensure adherence to temporal constraints.
- The core of the proposed algorithm is an FP-Growth-style algorithm, specifically customized for mining sequential rules.

Given this problem definition and the outlined assumptions, a methodology was proposed, structured into six steps and partially based on previous work on extracting frequent itemsets from a PSDB [44].

1. **Transformation of the PSDB**: convert the PSDB into a PDB.
2. **Extraction of singleton frequent items**: identify and extract the set of singleton frequent items.
3. **Construction of probabilistic frequent pattern tree (PFPTree)**: generate a specialized structure called the PFPTree to efficiently represent and extract probabilistic frequent patterns.
4. **Mining probabilistic frequent patterns**: extract the set of probabilistic frequent items using the PFPTree.
5. **Generation of probabilistic rules**: derive all possible probabilistic rules from the mined frequent patterns.
6. **Temporal filtering of rules**: apply a filtering mechanism to retain only those rules that satisfy the temporal constraints present in the SD.

A detailed discussion of these steps is provided in the following sub-sections.

3.6Transforming PSDB into PDB

A strategy inspired by [4] is employed to convert a PSDB into a PDB, which involves ignoring the sequential order of items, effectively treating sequences as unordered sets of items. This approach enables the transformation into a standard PDB.

3.7Extracting the set of singleton frequent patterns

To determine the set of singleton frequent patterns from the PDB, all possible scenarios or "worlds" that can arise from this database must be considered. Each possible world represents a distinct realization where decisions are made for uncertain items, determining whether they occur or not. Hence, a possible world wi (the ith possible world) signifies a specific certain configuration of the PDB, characterized by a probability denoted as P(wi). Therefore, using these possible worlds, the expected support of an item A in the PDB is calculated by Equation 6.

$$ExpSupp(A) = \sum_{i=1}^{w} p(w_i) A_{wi} \quad (6)$$

The expected support of A is calculated as the sum of products of the probability of each possible world and the count of occurrences of A in that world. However, this computation becomes very expensive due to the exponential growth in the number of possible worlds generated by a PDB regarding the number of uncertain items (the total number of possible worlds is $2^{|\text{uncertain items}|}$). The expected support of an itemset A is equal to the sum of the products of the probabilities assigned to the items belonging to A [45]. This relationship is expressed mathematically in Equation 7.

$$ExpSupp(A) = \sum_{j=1}^{|PDB|} \prod_{x \in A} P(x) \quad (7)$$

According to Equation 7, and as illustrated in *Figure 1*, which presents the pseudocode for calculating the expected support of all singleton items in the PDB, computing the expected support of A requires scanning the PDB transaction by transaction.

For each transaction, the product of the probabilities of the singleton items belonging to A is calculated, and these values are then summed over the entire PDB. Since all items are assumed to be singleton items (i.e., no nested sequences), Equation 7 can be simplified as shown in Equation 8.

$$ExpSupp(A) = \sum_{j=1}^{|PDB|} P(A) \quad (8)$$

Applying the aforementioned mathematical relationship to the running example (*Table 5*), the expected support values for each singleton item are computed as follows: *ExpSupp(X)=3.5, ExpSupp(Y)=4.7, ExpSupp(Z)=5, ExpSupp(W)=2.*

3.8Construction of the probabilistic frequent pattern tree (PFPTree)

The method for constructing the PFP-tree used in this work is inspired by the approach in [6]. Their algorithm was adapted to create a customized version of the PFP-tree. In this approach, constructing the PFP-tree requires only a single scan of the PDB, processing each transaction sequentially and inserting it into the PFP-tree. Each transaction is represented by a path starting from the root, with each node representing a single item in the transaction. The structure of the PFP-tree is as follows (consider a node *n* in the PFP-tree):

- *n.item*: a label used to denote the node, representing one item.
- *n.list* : a list of transactions where the item of this node appears, consisting of pairs: transaction label and the existential probability of the item in this transaction (if the item is certain, the existential probability is 1).
- *n.next*: a link pointing to the next node in the tree, which can be a list of nodes or null if the current node is a leaf.

The algorithm begins with a tree that contains only the root element, which is initialized with an empty label and an empty list, except for the *next* field, which is not empty and contains a link to the set of the root's child elements. It then processes each transaction individually, starting with transaction t_1, it inserts it directly as a path in the tree (path = root → $t_1.item_1$ → $t_1.item_2$ → ... → $t_1.item_n$) as illustrated in *Figure 1*. For the remaining transactions, there are two possibilities: if the path corresponding to the current transaction already exists in the tree, an update is performed by adding the transaction label and the existential probability of the current item to the transactions list field of the corresponding node. If the path does not exist in the tree, it is created entirely, before processing a transaction, its items are ordered based on their expected support calculated in the previous step. *Figure 1* represents the complete PFP-tree generated after the algorithm runs on the provided example.

Input: PDB: Probabilistic Database (set of transactions).
Output: ExpSupp: Expected support of each singleton item in PDB.
1: ExpSupp = 0
2: **For each** transaction t_i in PDB
3: A = the first singleton item in t_i
4: Prod = 1
5: **For each** singleton item (currentItem) in t_i
6: **If** currentItem = A **then**
7: Prod = Prod * P(A)
8: **End for.**
9: ExpSupp(A) = ExpSupp(A) + Prod
10: A = the next singleton item not treated in t_i, goto 4
11: **End for.**
12: **Return** ExpSupp.

Figure 1 Pseudo code of the algorithm for extracting the set of singleton frequent patterns

As shown in *Figure 2*, which presents the pseudocode of the algorithm responsible for constructing the PFPTree, it is evident that if the algorithm encounters a new transaction not yet included in the tree, it creates a new path for this transaction. However, if the transaction already exists in the tree, the algorithm updates the lists of nodes belonging to the path constructed based on the items in this transaction. Let's consider a transaction t_x= *(X:0.6, W:0.7, Y:1)* that needs to be added to the tree. The first step is to order the set of items according to their expected support, resulting in t_x= *(Y:1, X:0.6, W:0.7)*. Next, the children of the root node are explored to check for the presence of a child representing Y. If it does not exist, a new one is created. The value t_x = 1 is then added to the list associated with this node (Y.list). Next, the item X is processed. The child nodes of Y are explored to check for a child representing X. If it does not exist, a new one is created. The value t_x = 0.6 is then added to the list associated with this node (X.list). This process is repeated for the remaining items in the transaction. *Figure 3* illustrates the PFP-Tree obtained after applying the proposed algorithm to the running example previously presented in *Table 5*.

Input: PDB: Probabilistic database (set of transactions).
Output: PFP-tree: Probabilistic frequent pattern tree

```
1: root = null
2: For each transaction ti in PDB
3:         newPath = false;
4:         to= Ordered ti according to the value of expected support compute in step one;
5:         currentItem = next singleton item in to;
6:         For each child of root
7:         |       if child.label == currentItem and not (newPath) then
8:         |       |       add (ti = P(currentItem)) to child.list;
9:         |       |       child = child.next;
10:        |       |       For each currentItem in to.
11:        |       |       |       if child.label == currentItem and not (newPath) then
12:        |       |       |       |       add (ti = P(currentItem)) to child.list;
13:        |       |       |       Else
14:        |       |       |               newPath = true;
15:        |       |       |               break;
16:        |       |       endfor
17:        |       else
18:        |               newPath = true;
19:        |               break;
20:        endfor
21:        Create new child of the node root;
22:                child.label = currentItem;
23:                child.next = null;
24:                child.list = child.list+(ti = P(currentItem));
25: endfor
26: Return root.
```

Figure 2 Pseudocode of the algorithm for constructing the PFPTree

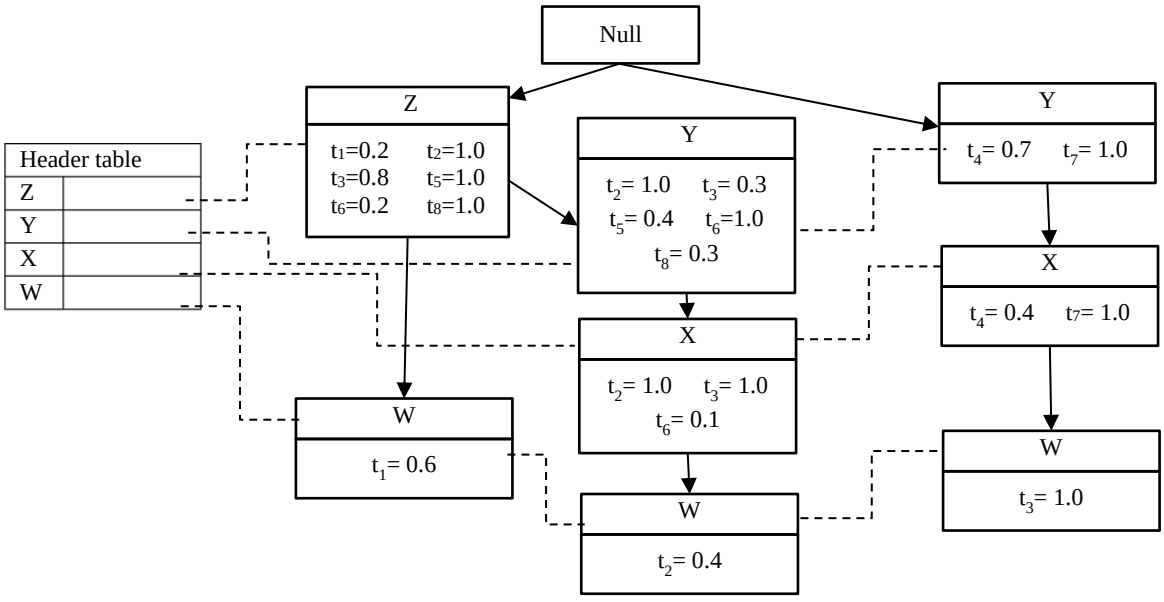

Figure 3 The PFP-Tree after scanning the whole PDB

3.9Mining probabilistic frequent patterns

In this step, the set of probabilistic frequent itemsets is extracted based on the singleton frequent items identified in the previous step. Given the uncertainty in the data, the concept of expected support is used for singleton frequent items, as defined by Equation

8. By applying Equation 1 to the running example (*Table 5*) to calculate the expected support for the itemset {Z, W}, the result obtained is ExpSupp({Z, W}) = 0.52.

3.10Generation of probabilistic rules

The set of probabilistic rules is generated from the set of probabilistic frequent itemsets, where for each frequent itemset all its nonempty subsets are generated. Then the rule is generated as follow:

- The left side of the rule is any subset of the treated itemset.
- The right side of the rule is the remaining part of the itemset, excluding the elements on the left side.

For example, if *s* is a subset of *I*, the rule considered is *R: s* → *I* – *s*. However, if its confidence does not meet or exceed the minimum confidence threshold defined for to the application, rule is ignored and excluded from further consideration. The confidence of the rule can be calculated using Equation 9.

$$Conf(R) = ExpSupp(I^f)/ExpSupp(S) \quad (9)$$

3.11Temporal filtering of rules

After extracting the set of probabilistic rules, the next step is sequential filtering. For each rule identified in the previous step, sequential support and confidence are calculated. Only rules with a sequential confidence greater than or equal to a predefined threshold, referred to as the minimum sequential confidence, are selected. If $x \rightarrow y$ is a rule identified in the previous step, two measures previously defined in this paper are used: sequential support, as described in Equation 4, and sequential confidence, as described in Equation 5. Only the rules with sequential confidence that meet or exceed a predefined threshold are retained, forming the set of probabilistic sequential rules.

4.Results and discussion

The proposed algorithm was implemented using Python on a Windows machine with an Intel Core i5 processor and 16 GB of RAM. For testing, Mushroom dataset [46] and synthetic datasets were used.

4.1Testing the algorithm on synthetic data

Before presenting the results, an overview of the synthetic data is provided. The data were randomly generated by simulating a PSDB and creating sequences containing various items, each assigned an existential probability between zero and one. A set of ten files was then created, each containing a different number of sequences, with the first file containing 500 sequences and the last one containing 8,000 sequences. *Figure 4* illustrates the time taken by the proposed algorithm to generate the set of sequential rules, measured in seconds. The processing time depends on the number of sequences in the data. The figure includes three plots: one for the total time taken by the entire algorithm, another for the time taken to generate the set of probabilistic rules, and the third plot shows the time taken for sequential filtering, with the first plot representing the total time, which equals the sum of the other two times. Clearly, the execution time increases as the number of sequences grows. However, in terms of time complexity, the proposed algorithm maintains a reasonable complexity, typically falling within O(n log(n)) and, in the worst-case scenario, $O(n^2)$, as measured using Big O notation to assess algorithmic complexity.

Figure 5 illustrates the space complexity of the proposed approach. On the x-axis, the number of sequences in the synthetic data is represented, while the y-axis indicates the memory required to execute the algorithm in megabytes. This analysis shows that the algorithm's memory usage increases as the input data grows. However, it performs optimally with larger datasets, where memory consumption stabilizes and continues to increase at a negligible rate.

4.2Evaluating algorithm performance with real-world data

The mushroom dataset [45], used in this study, comprises descriptions of samples from 23 species of gilled mushrooms belonging to the Agaricus and Lepiota families. Each species is classified as either definitely edible, definitely poisonous, edible but with uncertain recommendation, or inedible. The latter category includes mushrooms with toxic properties. The guide emphasizes that determining a mushroom's edibility is not straightforward.

In this phase of the research, no existing PSDB suitable for benchmarking this type of study was found. However, a SD can be derived from a transactional database by considering the order of items within each transaction. In contrast, a PDB is a standard database where each item is associated with its own existential probability.

Figures 6 and *7* present the temporal and spatial complexities of the proposed algorithm compared to the naive algorithm when tested on real-world data, specifically the Mushroom dataset. The naive algorithm's results serve as a baseline for evaluating

the performance of the proposed algorithm. These experiments were conducted with a minimum support threshold, where the minimum support divided by the number of sequences equals 0.5.

Figure 6 displays the space complexity of the proposed algorithm. The x-axis represents the dataset size in terms of the number of sequences, while the y-axis indicates the amount of memory used in each experiment. The graph confirms that the algorithm exhibits quadratic space complexity relative to the dataset size.

Figure 7 illustrates the runtime of the proposed algorithm relative to the number of sequences in the dataset. The algorithm demonstrates quadratic behavior, with runtime increasing as the number of sequences grows. Interestingly, in some instances, the execution time decreases between two experiments. This occurs because, although the dataset size in the first experiment is larger than in the second, the number of rules discovered in the second experiment is greater. The increased number of rules reduces processing time per rule, as shown in the graph.

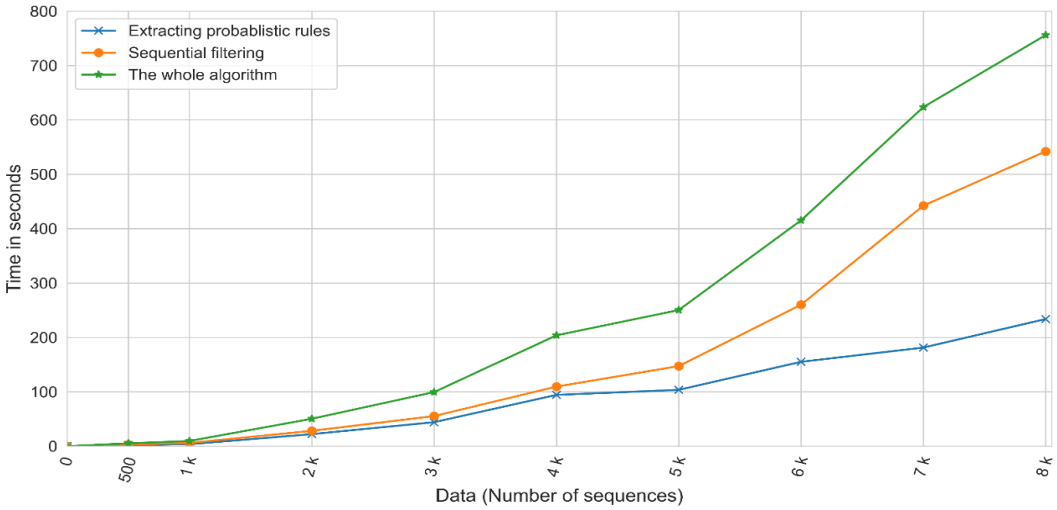

Figure 4 Temporal Complexity of the proposed algorithm tested on synthetic data

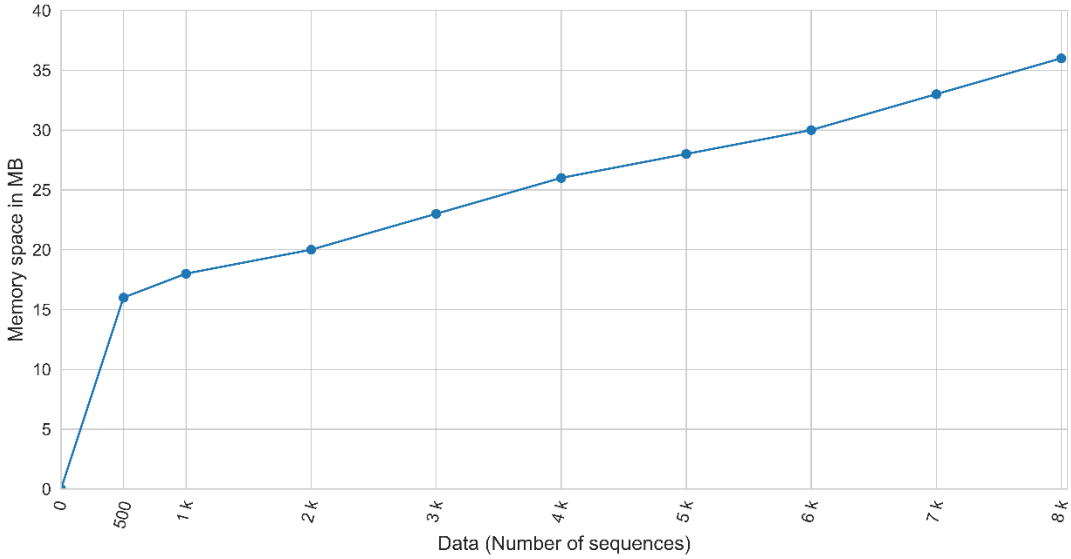

Figure 5 Spatial Complexity of the proposed algorithm tested on synthetic data

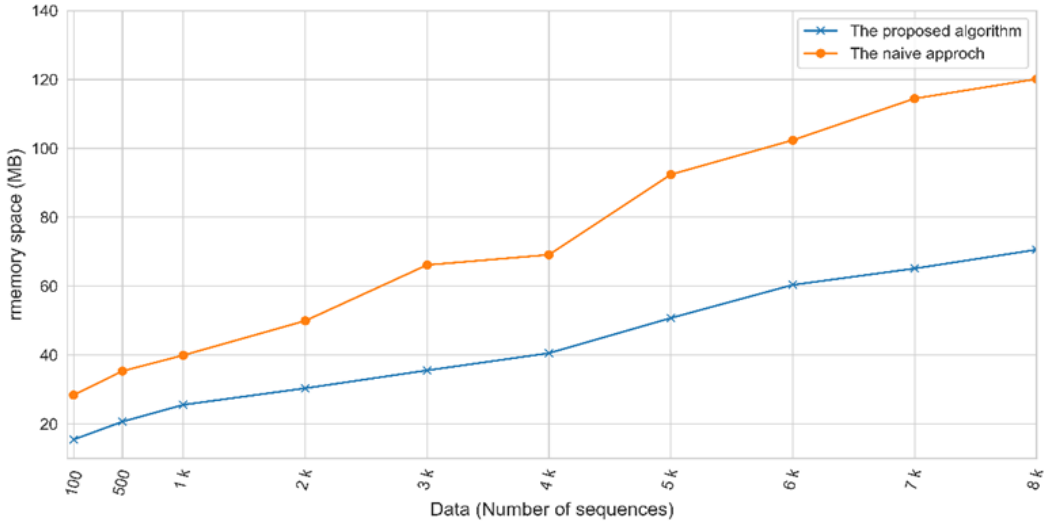

Figure 6 Spatial Complexity of the proposed algorithm tested on real data compared to the naive algorithm

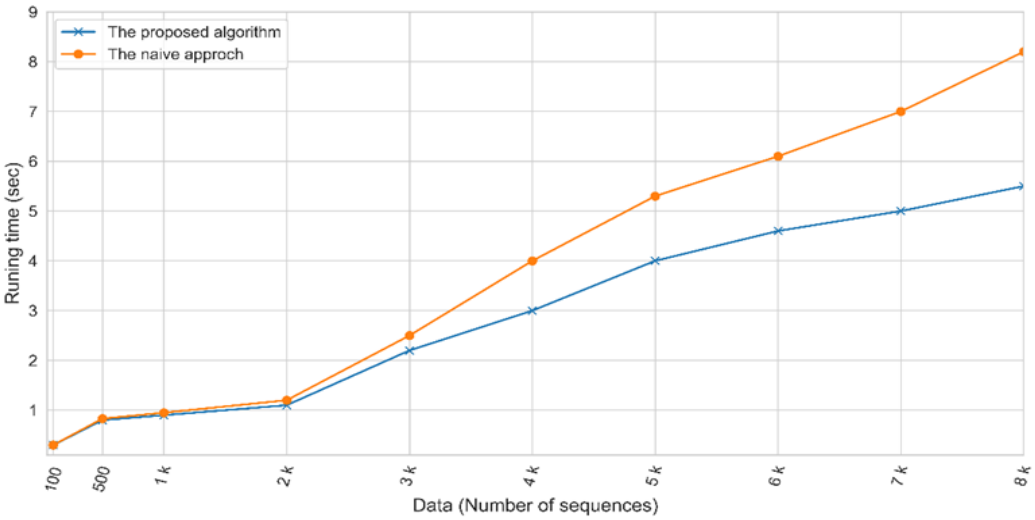

Figure 7 Temporal Complexity of the proposed algorithm tested on real data compared to the naive algorithm

4.3Discussion

The experimental results of the proposed algorithm, implemented in Python and tested on both synthetic and real-world datasets, provide critical insights into its performance and practicality.

Testing the algorithm on synthetic data provided a controlled environment to evaluate its performance across various scenarios. The synthetic data, generated by simulating a PSDB, varied in size from 100 to 8000 sequences. The results demonstrated a clear increase in execution time as the number of sequences grew, which aligns with expectations given the algorithm's time complexity. Specifically, the temporal complexity of the algorithm was found to be quite reasonable, falling within O(n log(n)) or $O(n^2)$ in the worst-case scenario. This finding is significant as it confirms the algorithm's efficiency in handling larger datasets, which is a critical requirement for practical applications.

The analysis of space complexity revealed that memory consumption also increased with larger input data, but at a diminishing rate. This suggests that the algorithm performs optimally with larger datasets, where the memory consumption stabilizes. The ability to maintain a manageable increase in memory usage while scaling up the dataset size indicates the algorithm's robustness and efficiency in terms of resource utilization.

The evaluation of the algorithm using the mushroom dataset provided insights into its performance in real-world scenarios. The mushroom dataset, comprising descriptions of samples from various species of gilled mushrooms, presented a more complex and nuanced dataset compared to the synthetic data.

The spatial complexity analysis confirmed that the algorithm exhibits quadratic space complexity relative to the dataset size. This result is consistent with the synthetic data analysis and reinforces the algorithm's capability to manage memory effectively even as the dataset size increases. The temporal complexity analysis showed quadratic behavior with runtime increasing as the number of sequences grew. Interestingly, there were instances where the execution time decreased between two experiments despite an increase in dataset size. This phenomenon can be attributed to the nature of the rules discovered, where a larger number of rules in smaller datasets reduced the processing time per rule.

The results from both synthetic and real-world data highlight the algorithm's scalability and efficiency, the reasonable time complexity and stabilized memory usage with larger datasets make it suitable for practical applications involving large-scale data. Additionally, the observed quadratic behavior in both temporal and spatial complexities indicates that the algorithm can handle substantial increases in data size without exponential growth in resource requirements.

Comparing new scientific work with existing research is a cornerstone of scientific inquiry. However, finding prior studies that directly align with new research is not always feasible. In such cases, a practical alternative is to benchmark the work against a naive method, which provides a simple baseline for evaluation. In this study, the naive algorithm used as a baseline consists of three steps:

- Extract the set of probabilistic rules based on the expected support and confidence as defined in equations 7 and 9.
- Extract the set of sequential rules using the sequential support and confidence defined in equation 4 and 5.
- Determine the intersection of the two sets to identify the sequential probabilistic rules.

The comparison between the proposed algorithm and the naive approach as shown in *Figures 6* and *7* reveals significant differences in performance and scalability. The proposed algorithm consistently outperforms the naive approach in terms of running time, requiring less time to process the same number of sequences. This performance gap becomes more noticeable as the dataset size increases. For smaller datasets (up to 1k sequences), the running times of both methods are relatively close, with the naive approach slightly outperforming the proposed algorithm at the smallest dataset size. However, as the number of sequences grows, the proposed algorithm demonstrates superior scalability, with its runtime increasing at a much slower rate compared to the naive approach. In contrast, the naive approach exhibits a steeper growth in runtime, indicating inefficiency in handling larger datasets. By the time the dataset size reaches 8k sequences, the naive approach is significantly slower than the proposed algorithm. Overall, the proposed algorithm proves to be more efficient and scalable, particularly for larger datasets, making it a more effective solution in data-intensive scenarios.

4.4Limitations of the proposed algorithm

After examining the proposed algorithm for mining sequential probabilistic rules from uncertain SDs, the following limitations have been identified:

- **Handling of large datasets**: The algorithm exhibits quadratic complexity in both time and space, which may not be optimal for very large datasets. Although its scalability is reasonable, it could face challenges when processing extensive real-world datasets due to memory and time constraints.
- **Dependency on existential probabilities**: The algorithm relies heavily on the accurate assignment of existential probabilities to items within sequences. Any errors or biases in these probabilities can directly affect the quality of the mined rules, leading to less reliable results.
- **Sequential filtering approach**: The two-phase approach (first generating probabilistic rules and then applying temporal filtering) might introduce inefficiencies. If the initial rule set is large, the filtering step could become computationally expensive.
- **Lack of parallelization**: The algorithm does not appear to leverage parallel processing techniques. With modern hardware capabilities, parallelization could significantly reduce runtime, particularly for steps like PFPTree construction and probabilistic rule generation.
- **Assumptions on data structure**: The method assumes that the data is well-structured as a PSDB with clearly defined existential probabilities. This

assumption may limit its applicability to datasets with incomplete or ambiguous structures.

- **Unavailability of suitable datasets**: A significant limitation is the lack of publicly available datasets that simultaneously incorporate uncertain and sequential data. This restricts the evaluation of the algorithm to synthetic data or datasets that must be preprocessed to fit the required format. This limitation challenges the generalizability and validation of the algorithm's performance across diverse real-world scenarios.
- **Increased computational complexity**: Highly uncertain data increases the number of possible worlds that need to be considered during the computation of expected support. As the uncertainty grows, the algorithm must account for exponentially more configurations, which can lead to significant increases in computation time.

4.5Potential improvements

To reduce computational costs and optimize memory usage for the proposed algorithm, especially in real-world applications with high data volumes, the following improvements and optimizations can be considered:

- Parallel Processing: Split the dataset into smaller partitions and process them independently for tasks such as PFP-Tree construction, rule generation, and filtering. This approach can significantly reduce runtime by utilizing multiple cores or processors.
- Distributed Systems: Implement the algorithm on distributed platforms like Hadoop or Spark to efficiently handle large-scale datasets by distributing the workload across multiple nodes.

A complete list of abbreviations is listed in *Appendix I.*

5.Conclusion and future work

This study introduced a novel approach for mining sequential probabilistic rules from uncertain SD, addressing a significant gap in data mining. The proposed algorithm effectively handles data uncertainty by associating each item with an existential probability and leveraging expected support measures for probabilistic rule extraction. The methodology, structured in six key steps, ensures efficient transformation, rule generation, and temporal filtering, enhancing accuracy while maintaining computational efficiency. The algorithm was tested on synthetic and real-world datasets, demonstrating superior performance compared to traditional approaches in terms of execution time, memory efficiency, and scalability. Results confirm that the algorithm exhibits quadratic complexity, with optimized performance as dataset size increases.

Future work will focus on enhancing the capabilities of the proposed approach. Specifically, efforts will be directed toward developing a dedicated PSDB to further test and refine the proposed algorithms. Additionally, the methodology will be extended to incorporate tuple-level uncertainty models, broadening its applicability across diverse domains. To improve computational efficiency, future optimizations will aim to reduce time complexity, particularly in worst-case scenarios. Implementing parallel and distributed processing techniques will be explored to enhance scalability and real-time performance. Furthermore, extensive testing on diverse real-world datasets will be conducted to assess the algorithm’s generalizability and robustness. These advancements will ensure that the proposed approach evolves into a versatile and efficient tool for mining uncertain sequential data, making it more applicable to real-world scenarios such as cybersecurity, healthcare, and market analysis.

Acknowledgment

None.

Conflicts of interest

The authors have no conflicts of interest to declare.

Data availability

The mushroom dataset utilized in this study is publicly accessible and can be found at http://archive.ics.uci.edu/ml

Author’s contribution statement

Imane Seddiki: Conceptualization, Investigation, Data collection, Implementation, Writing – original draft, Writing – review and editing. **Farid Nouioua**: Conceptualization, Investigation, Data collection, Writing – review and editing, Supervision. **Abdelbasset Barkat**: Supervision, Investigation on challenges and Draft manuscript preparation.

References

[1] Agrawal R, Imieliński T, Swami A. Mining association rules between sets of items in large databases. In proceedings of the 1993 SIGMOD international conference on management of data 1993 (pp. 207-16). ACM.

[2] Jashma SPP, Dinesh AU, Reddy NS. Mining frequent itemsets from transaction databases using hybrid switching framework. Multimedia Tools and Applications. 2023; 82(18):27571-91.

[3] Islam MS, Kar PC, Samiullah M, Ahmed CF, Leung CK. Discovering probabilistically weighted sequential

patterns in uncertain databases. Applied Intelligence. 2023; 53(6):6525-53.
[4] Huang G, Gan W, Yu PS. TaSPM: targeted sequential pattern mining. ACM Transactions on Knowledge Discovery from Data. 2024; 18(5):1-8.
[5] Fournier-viger P, Faghihi U, Nkambou R, Nguifo EM. CMRules: mining sequential rules common to several sequences. Knowledge-Based Systems. 2012; 25(1):63-76.
[6] Zhao Q, Bhowmick SS. Association rule mining: a survey. Nanyang Technological University, Singapore. 2003; 135:1-20.
[7] Sun L, Cheng R, Cheung DW, Cheng J. Mining uncertain data with probabilistic guarantees. In proceedings of the 16th SIGKDD international conference on knowledge discovery and data mining 2010 (pp. 273-82). ACM.
[8] Bernecker T, Kriegel HP, Renz M, Verhein F, Züfle A. Probabilistic frequent pattern growth for itemset mining in uncertain databases. In international conference on scientific and statistical database management 2012 (pp. 38-55). Berlin, Heidelberg: Springer Berlin Heidelberg.
[9] Leemans SJ, Van ZSJ, Lu X. Partial-order-based process mining: a survey and outlook. Knowledge and Information Systems. 2023; 65(1):1-29.
[10] Fister JI, Fister I, Fister D, Podgorelec V, Salcedo-sanz S. A comprehensive review of visualization methods for association rule mining: taxonomy, challenges, open problems and future ideas. Expert Systems with Applications. 2023; 233:120901.
[11] Wael M, Kassem G. A systematic literature review toward standardization of business rules discovery in the context of process mining. In international conference on technological advancement in embedded and mobile systems 2024 (pp. 33-42). Springer, Cham.
[12] Wang J, Wang C, Huang J, Gao M, Zhou A. Uncertainty-aware self-training for low-resource neural sequence labeling. In proceedings of the AAAI conference on artificial intelligence 2023 (pp. 13682-90). AAAI.
[13] Chen CM, Zhang Z, Ming-tai WJ, Lakshmanna K. High utility periodic frequent pattern mining in multiple sequences. CMES-Computer Modeling in Engineering & Sciences. 2023; 137(1):733-59.
[14] Gao D, Zhu Y, Soares CG. Uncertainty modelling and dynamic risk assessment for long-sequence AIS trajectory based on multivariate gaussian process. Reliability Engineering & System Safety. 2023; 230:108963.
[15] Yeshchenko A, Mendling J. A survey of approaches for event sequence analysis and visualization. Information Systems. 2024; 120:102283.
[16] Zhang Y, Paquette L. Sequential pattern mining in educational data: the application context, potential, strengths, and limitations. In educational data science: essentials, approaches, and tendencies: proactive education based on empirical big data evidence 2023 (pp. 219-54). Singapore: Springer Nature Singapore.
[17] Tong Y, Chen L, Cheng Y, Yu PS. Mining frequent itemsets over uncertain databases. Proceedings of the VLDB Endowment. 2012; 5(11):1650-61.
[18] Ahmed AU, Ahmed CF, Samiullah M, Adnan N, Leung CK. Mining interesting patterns from uncertain databases. Information Sciences. 2016; 354:60-85.
[19] Bernecker T, Cheng R, Cheung DW, Kriegel HP, Lee SD, Renz M, et al. Model-based probabilistic frequent itemset mining. Knowledge and Information Systems. 2013; 37:181-217.
[20] Huang G, Gan W, Weng J, Yu PS. US-Rule: discovering utility-driven sequential rules. ACM Transactions on Knowledge Discovery from Data. 2023; 17(1):1-22.
[21] Wu Y, Zhao X, Li Y, Guo L, Zhu X, Fournier-viger P, et al. OPR-miner: order-preserving rule mining for time series. IEEE Transactions on Knowledge and Data Engineering. 2023; 35(11):11722-35.
[22] Zhang C, Lyu M, Gan W, Yu PS. Totally-ordered sequential rules for utility maximization. ACM Transactions on Knowledge Discovery from Data. 2024; 18(4):1-23.
[23] Fournier-viger P, Gueniche T, Zida S, Tseng VS. ERMiner: sequential rule mining using equivalence classes. In advances in intelligent data analysis XIII: 13th international symposium, Leuven, Belgium, 2014 (pp. 108-19). Springer International Publishing.
[24] Subrahmanian VS, Pulice C, Brown JF, Bonen-clark J, Subrahmanian VS, Pulice C, et al. Temporal probabilistic rules and policy computation algorithms. A Machine Learning Based Model of Boko Haram. 2021:43-52.
[25] Zhang L, Yang G, Li X. Mining sequential patterns of PM2.5 pollution between 338 cities in China. Journal of Environmental Management. 2020; 262:110341.
[26] Shaheen M, Abdullah U. CARM: context based association rule mining for conventional data. Computers, Materials & Continua. 2021; 68(3):3305-22.
[27] Le T, Vo B, Huynh VN, Nguyen NT, Baik SW. Mining top-k frequent patterns from uncertain databases. Applied Intelligence. 2020; 50:1487-97.
[28] Islam MA, Rafi MR, Azad AA, Ovi JA. Weighted frequent sequential pattern mining. Applied Intelligence. 2022; 52(1):254-81.
[29] Leung CK. Mining uncertain data. Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery. 2011; 1(4):316-29.
[30] Modi G, Bansal S, Patidar MA. A survey on sequential rule mining techniques. International Journal for Technological Research in Engineering. 2018; 6(3):4825-8.
[31] Diaz-garcia JA, Ruiz MD, Martin-bautista MJ. A survey on the use of association rules mining techniques in textual social media. Artificial Intelligence Review. 2023; 56(2):1175-200.
[32] Aguiar G, Krawczyk B, Cano A. A survey on learning from imbalanced data streams: taxonomy, challenges, empirical study, and reproducible experimental

framework. Machine Learning. 2024; 113(7):4165-243.
[33] Khan S, Shaheen M. From data mining to wisdom mining. Journal of Information Science. 2023; 49(4):952-75.
[34] Zhao X, Zhou X, Li G. Automatic database knob tuning: a survey. IEEE Transactions on Knowledge and Data Engineering. 2023; 35(12):12470-90.
[35] Quvvatov B. SQL databases and big data analytics: navigating the data management landscape. Development of Pedagogical Technologies in Modern Sciences. 2024; 3(1):117-24.
[36] Bu C, Zheng X, Zhao X, Xu T, Bai X, Jia Y, et al. GenBase: a nucleotide sequence database. Genomics, Proteomics & Bioinformatics. 2024; 22(3):1-6.
[37] Petrey D, Zhao H, Trudeau SJ, Murray D, Honig B. PrePPI: a structure informed proteome-wide database of protein–protein interactions. Journal of Molecular Biology. 2023; 435(14):168052.
[38] Zhao Z, Yan D, Ng W. Mining probabilistically frequent sequential patterns in large uncertain databases. IEEE Transactions on Knowledge and Data Engineering. 2013; 26(5):1171-84.
[39] Fournier-viger P, Nkambou R, Tseng VS. RuleGrowth: mining sequential rules common to several sequences by pattern-growth. In proceedings of the symposium on applied computing 2011 (pp. 956-61). ACM.
[40] Sen P, Deshpande A, Getoor L. Representing tuple and attribute uncertainty in probabilistic databases. In seventh international conference on data mining workshops 2007 (pp. 507-12). IEEE.
[41] Yingtaweesittikul H, Wu J, Mongia A, Peres R, Ko K, Nagarajan N, et al. CREAMMIST: an integrative probabilistic database for cancer drug response prediction. Nucleic Acids Research. 2023; 51(D1): D1242-8.
[42] Civelli S, Forestieri E, Secondini M. Practical implementation of sequence selection for nonlinear probabilistic shaping. In optical fiber communications conference and exhibition 2023 (pp. 1-3). IEEE.
[43] Marchet C, Limasset A. Scalable sequence database search using partitioned aggregated bloom comb trees. Bioinformatics. 2023; 39(Supplement-1):252-9.
[44] Seddiki I, Nouioua F, Barkat A. Extracting sequential frequent itemsets from probabilistic sequences database. International Journal of Information Technology. 2023; 15(5):2509-15.
[45] Chui CK, Kao B, Hung E. Mining frequent itemsets from uncertain data. In advances in knowledge discovery and data mining: 11th Pacific-Asia conference, PAKDD 2007, Nanjing, China, 2007 (pp. 47-58). Springer Berlin Heidelberg.
[46] https://archive.ics.uci.edu/dataset/73/mushroom. Accessed 21 January 2025.

Imane Seddiki was born on August 16, 1993, in Algeria. She earned a Bachelor's degree in Computer Science from Mohamed Boudiaf University, M'sila, in 2013/2014, followed by a Master's degree in Computer Science from the same university in 2015/2016. In 2024, she completed her Ph.D. at Université Mohamed El Bachir El Ibrahimi in Bordj Bou Arréridj. Her research focuses on Data Mining, with a particular emphasis on Uncertain Data.
Email: seddiki.imane@univ-bba.dz

Farid Nouioua is currently a Full Professor of Computer Science at the University of Bordj Bou Arréridj. Previously, he served as an Assistant Professor at Aix-Marseille University from 2008 to 2017. He earned his engineering degree in 1997 and his Magister in 2002, both in computer science from the University of Sétif, Algeria. He then obtained an MSc in Artificial Intelligence and Combinatorial Optimization in 2003 and a Ph.D. in Computer Science in 2007 from Paris-Nord University, France. In 2016, he received his HDR (Habilitation à Diriger des Recherches) in Computer Science from Aix-Marseille University, France. He has published approximately 50 research papers in international conference proceedings and journals. His research interests include data mining, knowledge representation and reasoning, and diagnosis in discrete event systems.
Email: farid.nouioua@univ-bba.dz

Abdelbasset Barkat earned his Master's degree in 2012 from Biskra University, Algeria, with a focus on building ontologies from text. He later completed his Ph.D. in Computer Science at the University of Biskra in 2018. Currently, he is an Associate Professor in the Department of Computer Science at the University of M'sila. His research interests span various areas of Computer Science, with a particular emphasis on cloud computing, web services, multi-agent systems, and optimization. Dr. Barkat is affiliated with the Laboratory of Informatics and its Applications at the Faculty of Mathematics and Computer Science, University of M'sila, Algeria.
Email: abdelbasset.barkat@univ-msila.dz

Appendix I

S. No.	Abbreviation	Description
1	CMRules	Common Sequential Rules
2	CM-SPAM	Common Sequential Pattern Minin
3	DNA	Deoxyribonucleic Acid
4	EFO-Miner	Efficient Frequent Order-Preserving Pattern Miner
5	ERMiner	Equivalence Class Based Sequential Rule Miner
6	FP- Growth	Frequent Pattern Growth
7	HUSRM	High-Utility Sequential Rule Mining
8	HUSRs	High-Utility Sequential Rules
9	HTSR	High-Utility Totally-Ordered Sequential Rules
10	OPPs	Order-Preserving Patterns
11	OPR-Miner	Order-Preserving Rule Miner
12	PDB	Probabilistic Databases
13	PFPTree	Probabilistic Frequent Pattern Tree
14	ProFPGrowth	Probabilistic Frequent Pattern Growth
15	PSDB	Probabilistic Sequence Database
16	SD	Sequence Database
17	TaSPM	Targeted Sequential Pattern Mining
18	TDB	Transaction Database
19	TODIS	TOp-Down Inheritance of Support pmf
20	TP-rules	Temporal Probabilistic Rules
21	UIIP	Unpromising I-Extension Item Pruning
22	UPIP	Unpromising Prefix Item Pruning
24	USIP	Unpromising S-Extension Item Pruning
25	US-Rule	Utility Sequential Rule Algorithm
26	UTFP	Unpromising Transaction Filter Pruning
27	WFSPM	Weighted Frequent Sequential Pattern Mining
28	WUIPM	Weighted Uncertain Interesting Pattern Mining