

Chapter 14

Uncertain Frequent Pattern Mining

Carson Kai-Sang Leung

Abstract Frequent pattern mining aims to discover implicit, previously unknown and potentially useful knowledge—in the form of frequently occurring sets of items—that are embedded in data. Many of the models and algorithms developed in the early days mine frequent patterns from traditional transaction databases of precise data such as shopper market basket data, in which the contents of databases are known. However, we are living in an uncertain world, in which uncertain data can be found in various real-life applications. Hence, in recent years, researchers have paid more attention to frequent pattern mining from probabilistic datasets of uncertain data. This chapter covers key models, algorithms and topics about uncertain frequent pattern mining.

Keywords Data mining · Knowledge discovery from uncertain data · Association rule mining · Frequent patterns · Frequent itemsets · Probabilistic approach · Uncertain data

1 Introduction

As an important data mining task, *frequent pattern mining* [8, 12] aims to discover implicit, previously unknown and potentially useful knowledge—revealing patterns on collections of frequently co-occurring items, objects or events—that are embedded in data. Nowadays, frequent pattern mining is commonly used in various real-life business, government, and science applications (e.g., banking, bioinformatics, environmental modeling, epidemiology, finance, marketing, medical diagnosis, meteorological data analysis). Uncertain data are present in many of these applications. Uncertainty can be caused by (i) our limited perception or understanding of reality; (ii) limitations of the observation equipment; or (iii) limitations of available resources for the collection, storage, transformation, or analysis of data. It can also be inherent in nature (say, due to prejudice). Data collected by acoustic, chemical, electromagnetic, mechanical, optical radiation, thermal sensors [6] in environment surveillance, security, and manufacturing systems can be noisy. Dynamic

C. Kai-Sang Leung (✉)
University of Manitoba, Winnipeg, MB, Canada
e-mail: kleung@cs.umanitoba.ca

errors—such as (i) inherited measurement inaccuracies, (ii) sampling frequency of the sensors, (iii) deviation caused by a rapid change (e.g., drift, noise) of the measured property over time, (iv) wireless transmission errors, or (v) network latencies—also introduce uncertainty into the data reported by these sensors. Moreover, there is also uncertainty in survey data (e.g., number “0” vs. symbol “o” vs. letter “O” or “o”; similarly, number “1” vs. upper case letter “I” vs. lower case letter “l”) and uncertainty due to data granularity (e.g., city, province) in taxonomy. Disguised missing data, which are not explicitly represented as such but instead appear as potentially valid data values, also introduce uncertainty. Furthermore, in privacy-preserving applications [10], sensitive data may be intentionally blurred via aggregation or perturbation so as to preserve data anonymity. All these sources of uncertainty lead to huge amounts of *uncertain data* in real-life applications [48, 52].

Many key models and algorithms have been developed over the past few years for various uncertain data mining tasks [3, 11]. These include (i) clustering uncertain data [2, 7, 25], (ii) classifying uncertain data [47, 53] and (iii) detecting outliers from uncertain data [5, 9]. In this chapter, we examine another data mining task—namely, *uncertain frequent pattern mining*. To mine frequent patterns from uncertain data, different methodologies (e.g., fuzzy set theory, rough set theory) can be applicable. Among them, probability theory is more popular and widely used by many researchers.

In this chapter, we focus on uncertain frequent pattern mining in a probabilistic setting. The remainder of this chapter is organized as follows. The next section describes a key model for uncertain frequent pattern mining: the probabilistic model. Then, we present those key uncertain frequent pattern mining algorithms based on (i) the candidate generate-and-test paradigm, (ii) the frequent pattern growth paradigm with hyperlinked structures, and (iii) the frequent pattern growth paradigm with tree structures in Sects. 3, 4 and 5, respectively. Sections 6, 7 and 8 describe key algorithms for uncertain frequent pattern mining (i) with constraints, (ii) from Big data, and (iii) from data streams, respectively. Section 9 examines key algorithms for mining uncertain data that are in vertical representation. We briefly discuss and compare these algorithms in Sect. 10. While Sects. 2 to 10 focus on (expected support based) frequent patterns, Sect. 11 focuses on the probabilistic frequent patterns. Finally, Sect. 12 gives conclusions.

2 The Probabilistic Model for Mining Expected Support-Based Frequent Patterns from Uncertain Data

As a building block for association rule mining [12] (which helps reveal associative relationships embedded in data), frequent pattern mining aims to discover frequently occurring sets of items, objects or events (e.g., frequently purchased merchandise items in shopper market baskets, bundles of popular books, popular courses taken by students, events that are frequently collocated). In the early days, most frequent

Table 14.1 An example of a traditional database D_1 of precise data

Transaction ID	Set of items
t_1	$\{a, b, c\}$
t_2	$\{a, b, c, d\}$
t_3	$\{a, b, d, e\}$
t_4	$\{a, b, c, e\}$

Table 14.2 An example of a probabilistic dataset D_2 of uncertain data

Transaction ID	Set of items with existential probability
t_1	$\{a:0.2, b:0.9, c:0.4\}$
t_2	$\{a:0.6, b:0.6, c:0.6, d:0.9\}$
t_3	$\{a:0.6, b:0.5, d:0.5, e:0.7\}$
t_4	$\{a:0.9, b:0.2, c:0.8, e:0.3\}$

pattern mining algorithms searched traditional databases of precise data (e.g., Table 14.1) such as shopper market basket data, in which the contents of databases are known. However, we are living in an uncertain world. Uncertain data can be found in various real-life applications, in which users may not be certain about the presence or absence of an item x in a transaction t_i in a probabilistic dataset D of uncertain data (e.g., Table 14.2). Users may suspect, but cannot guarantee, that x is present in t_i . The uncertainty of such suspicion can be expressed in terms of *existential probability* $P(x, t_i)$, which indicates the likelihood of x being present in t_i in D . The existential probability $P(x, t_i)$ ranges from a positive value close to 0 (indicating that x has an insignificantly low chance to be present in D) to a value of 1 (indicating that x is definitely present). With this notion, each item in any transaction in traditional databases of precise data (e.g., shopper market basket data) can be viewed as an item with a 100 % likelihood of being present in such a transaction.

The **probabilistic model** [1, 23] is a key model that commonly used for uncertain frequent pattern mining. When using the “possible world” interpretation of uncertain data, there are two possible worlds for an item x in a transaction t_i :

(i) a possible world W_1 where x is present in t_i (i.e., $x \in t_i$)

and

(ii) another possible world W_2 where x is absent from t_i (i.e., $x \notin t_i$).

Although it is uncertain which of these two worlds is the true world, the probability of W_1 to be the true world is $P(x, t_i)$ and the probability of W_2 to be the true world is $1 - P(x, t_i)$. To a further extent, there are multiple items in each of many transactions in a probabilistic dataset D of uncertain data. In a domain of m distinct items, when there are a total of q independent items (which include multiple occurrences of some of the m domain items, where $m \ll q$) in all transactions of D , there are $O(2^q)$ possible worlds. The **expected support** of a pattern X in D —denoted as $expSup(X, D)$ —can then be computed by summing the support $sup(X, W_j)$ of X in possible world W_j (while taking into account the probability $Prob(W_j)$ of W_j to

Table 14.3 Possible worlds for the probabilistic dataset D_2 of uncertain data

Possible world W_j	$Prob(W_j)$	Transactions
W_1	6.349×10^{-5}	$\{t_1=\{a, b, c\},$ $t_2=\{a, b, c, d\},$ $t_3=\{a, b, d, e\},$ $t_4=\{a, b, c, e\}\}$
W_2	1.481×10^{-4}	$\{t_1=\{a, b, c\},$ $t_2=\{a, b, c, d\},$ $t_3=\{a, b, d, e\},$ $t_4=\{a, b, c, e\}\}$
W_3	1.587×10^{-5}	$\{t_1=\{a, b, c\},$ $t_2=\{a, b, c, d\},$ $t_3=\{a, b, d, e\},$ $t_4=\{a, b, e\}\}$
\vdots	\vdots	\vdots
W_{32767}	1.769×10^{-7}	$\{t_1=\{\}, t_2=\{\}, t_3=\{\}, t_4=\{e\}\}$
W_{32768}	4.129×10^{-7}	$\{t_1=\{\}, t_2=\{\}, t_3=\{\}, t_4=\{\}\}$
	$\sum_j Prob(W_j) = 1$	

be the true world) over all possible worlds, i.e.,

$$expSup(X, D) = \sum_j [sup(X, W_j) \times Prob(W_j)], \tag{14.1}$$

where $sup(X, W_j)$ counts the occurrences of X (i.e., the number of transactions containing *all* the items within X) and $Prob(W_j)$ can be computed by the following equation:

$$Prob(W_j) = \prod_{i=1}^{|D|} \left[\prod_{x \in t_i \text{ in } W_j} P(x, t_i) \times \prod_{y \notin t_i \text{ in } W_j} (1 - P(y, t_i)) \right]. \tag{14.2}$$

Table 14.3 shows all “possible worlds” of the probabilistic dataset D_2 in Table 14.2. When items within the pattern X are independent, Eq. (14.1) can be simplified [32] to become the following equation:

$$expSup(X, D) = \sum_{i=1}^{|D|} \left(\prod_{x \in X} P(x, t_i) \right). \tag{14.3}$$

In other words, the expected support of X in D can be computed as a sum (over all $|D|$ transactions) of the product of existential probabilities of all items within X . Then, we can define the research problem of uncertain frequent pattern mining in terms of $expSup(X, D)$ as follows.

Definition 14.1 Given (i) a probabilistic dataset D of uncertain data and (ii) a user-specified support threshold $minsup$, the research problem of **uncertain frequent pattern mining** from a probabilistic dataset D of uncertain data is to find every pattern X having $expSup(X, D) \geq minsup$. Such a pattern X is called an *expected support-based frequent pattern* or just *frequent pattern* for short.

3 Candidate Generate-and-Test Based Uncertain Frequent Pattern Mining

One way to mine frequent patterns from uncertain data is to apply the candidate generate-and-test paradigm. For example, Chui et al. [19] proposed the **U-Apriori** algorithm, which mines frequent patterns from uncertain data in a levelwise breadth-first bottom-up fashion. Specifically, U-Apriori first computes the expected support of all domain items. Those items with expected supports $\geq \text{minsup}$ become every frequent pattern consisting of 1 item (i.e., frequent 1-itemset). Afterwards, the U-Apriori algorithm repeatedly applies the candidate generate-and-test process to generate candidate $(k+1)$ -itemsets from frequent k -itemsets and test if they are frequent $(k+1)$ -itemsets. Like its counterpart for mining precise data (the Apriori algorithm [8]), U-Apriori also relies on the *Apriori property* (which is also known as the *anti-monotonic property* or the *downward closure property*) that all subsets of a frequent pattern must also be frequent. Equivalently, all supersets of any infrequent pattern are also infrequent.

U-Apriori improves its efficiency by incorporating the *LGS-trimming strategy* (which includes local trimming, global pruning, and single-pass patch up) [19]. This strategy trims away every item with an existential probability below the user-specified trimming threshold (which is local to each item) from the original probabilistic dataset D of uncertain data and then mines frequent patterns from the resulting trimmed dataset D_{Trim} . On the one hand, if a pattern X is frequent in D_{Trim} , then X must be frequent in D . On the other hand, a pattern Y is infrequent in D if

$$\text{expSup}(Y, D_{Trim}) + e(Y) < \text{minsup},$$

where $e(Y)$ is an upper bound of estimated error for $\text{expSup}(Y, D_{Trim})$. Such an infrequent pattern Y can be pruned. Moreover, a pattern Z is potentially frequent in D if

$$\text{expSup}(Z, D_{Trim}) \leq \text{minsup} \leq \text{expSup}(Z, D_{Trim}) + e(Z).$$

To patch up, U-Apriori recovers the missing frequent patterns by verifying expected supports of potentially frequent patterns with an additional single-pass scan of D . Although the LGS strategy improves the efficiency of U-Apriori, the algorithm still suffers from the following problems: (i) there is an overhead in creating D_{Trim} , (ii) only a subset of all the frequent patterns can be mined from D_{Trim} and there is overhead to patch up, (iii) the efficiency of the algorithm is sensitive to the percentage of items having low existential probabilities, and (iv) it is not easy to find an appropriate value for the user-specified trimming threshold.

Chui and Kao [18] applied the *decremental pruning technique* to further improve the efficiency of U-Apriori. The technique helps reduce the number of candidate patterns by progressively estimating the upper bounds of expected support of candidate patterns after each transaction is processed. If the estimated upper bound of a candidate pattern X falls below minsup , then X is immediately pruned.

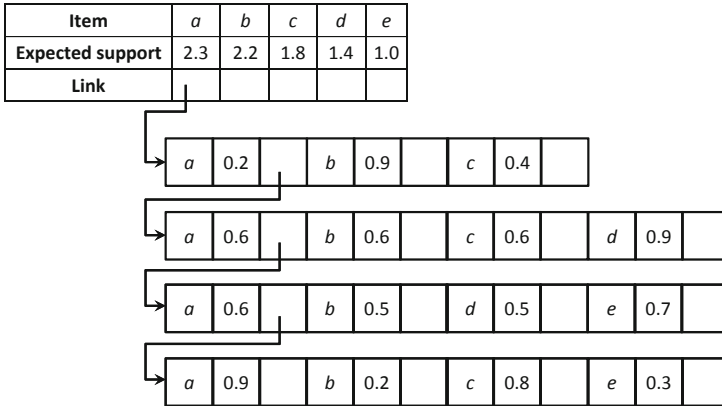


Fig. 14.1 The UH-struct for the probabilistic dataset D_2 of uncertain data

4 Hyperlinked Structure-Based Uncertain Frequent Pattern Mining

An alternative to candidate generate-and-test based mining is *pattern-growth mining*, which avoids generating a large number of candidates. Commonly used pattern-growth mining paradigms are mostly based on (i) hyperlinked structures or (ii) tree structures. In this section, let us focus on hyperlinked structure-based uncertain frequent pattern mining. As hyperlinked structure based mining employs a pattern-growth mining paradigm, the candidate generate-and-test mining paradigm of U-Apriori is avoided. In general, hyperlinked structure based algorithms capture the contents of datasets in a hyperlinked structure, from which frequent patterns are mined in a depth-first divide-and-conquer fashion.

Aggarwal et al. [13] proposed a hyperlinked structure based algorithm called **UH-mine** to mine frequent patterns from uncertain data. This algorithm captures the contents of a probabilistic dataset D of uncertain data in a hyperlinked structure called *UH-struct*. See Fig. 14.1. Like the H-struct for mining precise data, each row in the UH-struct represents a transaction t_i in D . Unlike the H-struct, the UH-struct captures the existential probability of items. In other words, for each item $x \in t_i$, the UH-struct maintains (i) x , (ii) its existential probability $P(x, t_i)$, and (iii) its hyperlink. Once the UH-struct is built, the corresponding UH-mine algorithm mines frequent patterns by recursively extending every frequent pattern X and adjusting its hyperlinks in the UH-struct.

As a preview, when compared with tree-based uncertain frequent pattern mining (i.e., another type of mining that relies on the pattern-growth mining paradigm), the UH-structure is not as compact as the tree structure used in tree-based mining. However, on the positive side, the UH-mine algorithm keeps only one UH-struct and adjusts the hyperlinks in it. In contrast, due to their recursive nature, tree-based mining algorithms usually keep multiple tree structures. Moreover, UH-mine

computes the expected support of frequent patterns on-the-fly so as to reduce the space requirement.

5 Tree-Based Uncertain Frequent Pattern Mining

Recall that (i) candidate generate-and-test based mining algorithms (e.g., the U-Apriori algorithm) use a levelwise bottom-up breadth-first mining paradigm and (ii) hyperlinked structure based algorithms (e.g., the UH-mine algorithm) recursively adjust the hyperlinks in the hyperlinked structure (e.g., UH-struct) to find frequent patterns from uncertain data in a depth-first fashion. As an alternative to Apriori-based and hyperlinked structure based mining, tree-based mining avoids generating many candidates and avoids recursively adjusting many hyperlinks. Tree-based algorithms use a depth-first divide-and-conquer approach to mine frequent patterns from a tree structure that captures the contents of the probabilistic dataset.

5.1 UF-growth

To mine frequent patterns from probabilistic datasets of uncertain data, Leung et al. [42] proposed a tree-based mining algorithm called **UF-growth**. Similar to its counterpart for mining precise data (the FP-growth algorithm [24]), UF-growth also constructs a tree structure to capture the contents of the datasets. However, it does not use the FP-tree (as in FP-growth) because each node in the FP-tree only maintains (i) an item and (ii) its occurrence count in the tree path. When mining precise data, the *actual support* of a pattern X depends solely on the occurrence counts of items within X . However, when mining uncertain data, the *expected support* of X is the sum of the product of the occurrence count and existential probability of every item within X . Hence, each node in the **UF-tree** (the tree structure for UF-growth) consists of three components: (i) an item, (ii) its existential probability, and (iii) its occurrence count in the path. See Fig. 14.2. Such a UF-tree is constructed in a similar fashion as the construction of the FP-tree, except that a new transaction is merged with a child node only if the same item *and the same existential probability* exist in both the transaction and the child node. As such, it may lead to a lower compression ratio than the original FP-tree. Fortunately, the number of nodes in a UF-tree is bounded above by the sum of the number of items in all transactions in the probabilistic dataset of uncertain data.

To reduce the memory consumption, UF-growth incorporates two improvement techniques [42]. The first technique is to discretize the existential probability of each node (e.g., round the existential probability to k decimal places such as $k=2$ decimal places), which reduces the potentially infinite number of possible existential probability values to a maximum of 10^k possible values. The second improvement technique is to limit the construction of UF-trees to only the first two levels (i.e., only construct the global UF-tree for the original probabilistic dataset D and a UF-tree for

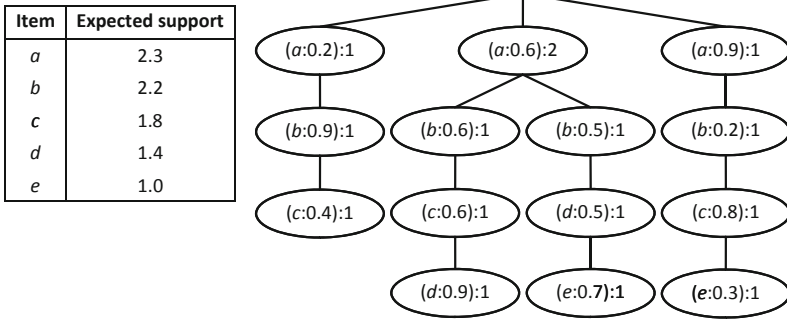


Fig. 14.2 The UF-tree for the probabilistic dataset D_2 of uncertain data

each frequent item—i.e., each singleton pattern) and to enumerate frequent patterns for higher levels (by traversing the tree paths and decrementing the occurrence counts) during the mining process.

On the one hand, as paths in a UF-tree are shared only if they have the same item and the same existential probability, the UF-tree accurately captures the contents (especially, the existential probabilities) of the probabilistic datasets of uncertain data so that frequent patterns can be mined without producing false positives or false negatives. On the other hand, the UF-tree may be large and may not be as compact as its counterpart for precise data (i.e., FP-tree).

5.2 UFP-growth

To make the tree more compact by reducing the tree size (via a reduction in the number of tree nodes), Aggarwal et al. [13] proposed the **UFP-growth algorithm**. Like UF-growth, the UFP-growth algorithm also scans the probabilistic dataset of uncertain data twice and builds a **UFP-tree**. As nodes for item x having similar existential probability values are clustered into a mega-node, the resulting mega-node in the UFP-tree captures (i) an item x , (ii) the *maximum* existential probability value (among all nodes within the cluster), and (iii) its occurrence count (i.e., the number of nodes within the cluster). Tree paths are shared if the nodes on these paths share the same item but *similar* existential probability values. In other words, the path sharing condition is less restrictive than that of the UF-tree. See Fig. 14.3. By extracting appropriate tree paths and constructing UFP-trees for subsequent projected databases, UFP-growth finds all truly frequent patterns at the end of the second scan of the probabilistic dataset of uncertain data. At the same time, due to the approximate nature (e.g., caused by the use of the maximum existential probability value among all the nodes clustered into a mega-node) of UFP-growth, UFP-growth also finds some infrequent patterns (i.e., some *false positives*) in addition to those truly frequent patterns (i.e., true positives). Hence, a third scan of the probabilistic dataset of uncertain data is then required to remove these false positives.

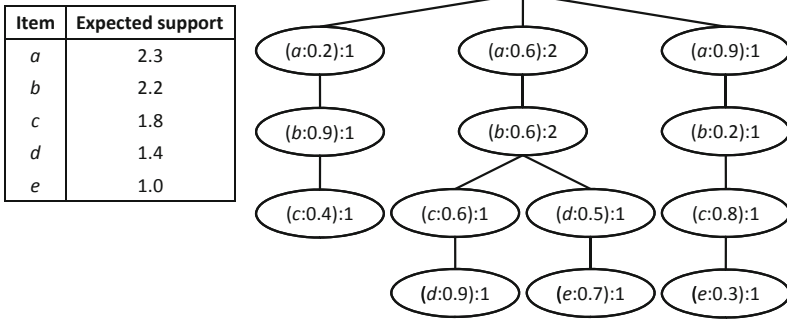


Fig. 14.3 The UFP-tree for the probabilistic dataset D_2 of uncertain data

Table 14.4 Transaction caps for the probabilistic dataset D_2 of uncertain data

Transaction ID	Set of items with existential probability	Transaction cap
t_1	{ <i>a</i> :0.2, <i>b</i> :0.9, <i>c</i> :0.4}	0.36
t_2	{ <i>a</i> :0.6, <i>b</i> :0.6, <i>c</i> :0.6, <i>d</i> :0.9}	0.54
t_3	{ <i>a</i> :0.6, <i>b</i> :0.5, <i>d</i> :0.5, <i>e</i> :0.7}	0.42
t_4	{ <i>a</i> :0.9, <i>b</i> :0.2, <i>c</i> :0.8, <i>e</i> :0.3}	0.72

5.3 CUF-growth

To further reduce the tree size (by reducing the number of tree nodes), Leung and Tanbeer [39] proposed an uncertain frequent pattern mining algorithm called **CUF-growth**, which builds a new tree structure called **CUF-tree**. Specifically, for each transaction t_i , CUF-growth computes a *transaction cap* which is defined as follows.

Definition 14.2 The **transaction cap**, denoted by $cap(t_i)$, of a transaction t_i is defined as the product of the two highest existential probability values of items within transaction t_i . Let

- (i) $h = |t_i|$ represent the length of t_i ,
- (ii) $M_1 = \max_{q \in [1, h]} P(x_q, t_i)$, and
- (iii) $M_2 = \max_{r \in [1, h], r \neq q} P(x_r, t_i)$.

Then,

$$cap(t_i) = \begin{cases} M_1 \times M_2 & \text{if } |t_i| > 1 \\ P(x_1, t_i) & \text{if } |t_i| = 1 \text{ (i.e., } t_i = \{x_1\}) \end{cases} \quad (14.4)$$

Table 14.4 shows the transaction cap for each transaction in a probabilistic dataset D_2 of uncertain data. The CUF-growth algorithm captures the transaction cap in the CUF-tree. Unlike the UF-tree (which captures an item, its existential probability and its occurrence count in each tree node), CUF-tree only capture (i) an item and (ii) its transaction cap. Paths in a CUF-tree are shared if the nodes on these paths share the same item. By doing so, the CUF-tree (for capturing uncertain data) can be as

Fig. 14.4 The CUF-tree for the probabilistic dataset D_2 of uncertain data

Item	Expected support
<i>a</i>	2.3
<i>b</i>	2.2
<i>c</i>	1.8
<i>d</i>	1.4
<i>e</i>	1.0

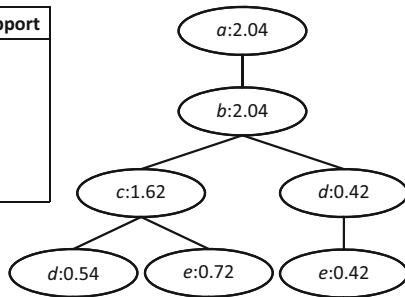
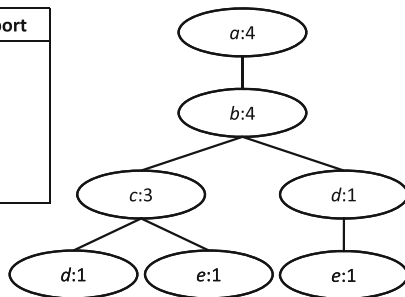


Fig. 14.5 The FP-tree for the traditional database D_1 of precise data

Item	Actual support
<i>a</i>	4
<i>b</i>	4
<i>c</i>	3
<i>d</i>	2
<i>e</i>	2



compact as the FP-tree (for capturing precise data). See Fig. 14.4, which shows the CUF-tree for probabilistic dataset D_2 of uncertain data. Note that this CUF-tree is as compact as an FP-tree (ref. Fig. 14.5) for the traditional database D_1 of precise data.

Like UFP-growth, the CUF-growth algorithm also takes three scans of the probabilistic dataset of uncertain data to mine frequent patterns. CUF-growth first scans the dataset to compute the transaction caps, and it then scans the dataset the second time to build the CUF-tree. The header table associated with the CUF-tree gives the expected support of frequent 1-itemsets (i.e., frequent singletons or frequent items). The CUF-tree stores transaction caps, which provide upper bounds to the expected support of frequent k -itemsets (for $k \geq 2$). For any k -itemset X , if the upper bound to its expected support is less than *minsup*, then X can be safely pruned.

By extracting appropriate tree paths and constructing CUF-trees for subsequent projected databases, CUF-growth finds all potentially frequent patterns at the end of the second scan of the probabilistic dataset of uncertain data. As these potentially frequent patterns include all truly frequent patterns and some infrequent patterns (i.e., some *false positives*), CUF-growth then quickly scans the dataset the third time to check each of them to verify whether or not they are truly frequent (i.e., prune false positives).

Table 14.5 Prefixed item caps for the probabilistic dataset D_2 of uncertain data

Transaction ID	Set of items w/existential probability (& prefixed item cap)
t_1	{ $a:0.2$ (0.2), $b:0.9$ (0.18), $c:0.4$ (0.36)}
t_2	{ $a:0.6$ (0.6), $b:0.6$ (0.36), $c:0.6$ (0.36), $d:0.9$ (0.54)}
t_3	{ $a:0.6$ (0.6), $b:0.5$ (0.30), $d:0.5$ (0.30), $e:0.7$ (0.42)}
t_4	{ $a:0.9$ (0.9), $b:0.2$ (0.18), $c:0.8$ (0.72), $e:0.3$ (0.27)}

5.4 PUF-growth

Along this direction, Leung and Tanbeer [40] observed that (i) the transaction cap provides CUF-growth with an upper bound to expected support of patterns and (ii) such an upper bound can be tightened in a tree-based environment. They introduced the concept of a *prefixed item cap*, which can be defined as follows.

Definition 14.3 The **prefixed item cap**—denoted by $I^{Cap}(x_r, t_i)$ —of an item x_r in a transaction $t_i = \{x_1, \dots, x_r, \dots, x_h\}$, where $1 \leq r \leq h$ (i.e., $h=|t_i|$ represent the length of t_i), is defined as the product of $P(x_r, t_i)$ and the highest existential probability value M of items from x_1 to x_{r-1} in t_i (i.e., in the *proper prefix* of x_r in t_i). More formally,

$$PIcap(x_r, t_i) = \begin{cases} P(x_r, t_i) \times M & \text{if } |t_i| > 1 \\ P(x_1, t_i) & \text{if } |t_i| = 1 \text{ (i.e., } t_i = \{x_1\}) \end{cases} \quad (14.5)$$

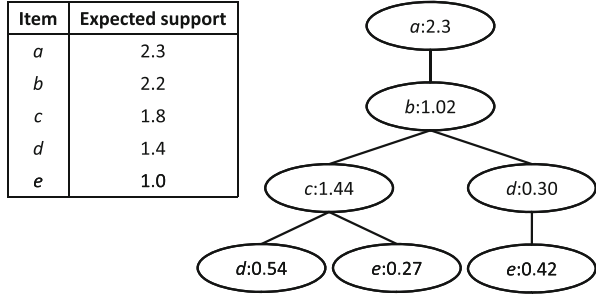
where $M = \max_{q \in [1, r-1]} P(x_q, t_i)$.

Assume that items are arranged in the order $\langle a, b, c, d, e \rangle$ from the root to leaves. Then, Table 14.5 shows the prefixed item cap for every item in a transaction in a probabilistic dataset D_2 of uncertain data. See Fig. 14.6 for how these prefixed item caps are captured in a new tree structure called **PUF-tree**, from which the corresponding algorithm called **PUF-growth** mines uncertain frequent patterns.

Like UFP-growth and CUF-growth, the PUF-growth algorithm also takes three scans of the probabilistic dataset of uncertain data to mine frequent patterns. With the first scan, PUF-growth computes the prefixed item caps. With the second scan, PUF-growth builds a PUF-tree to capture (i) an item and (ii) its corresponding prefixed item cap. Like those in CUF-tree, paths in the PUF-tree are shared if the nodes on these paths share the same item. Hence, the resulting PUF-tree is of the same size as the CUF-tree (also for capturing uncertain data), which can be as compact as the FP-tree (for capturing precise data). The header table associated with the PUF-tree gives the expected support of frequent 1-itemsets (i.e., singleton patterns or frequent items). The prefixed item caps in the PUF-tree provide upper bounds to the expected support of k -itemsets (for $k \geq 2$). For any k -itemset X , if the upper bound to its expected support is less than *minsup*, then X can be safely pruned.

By extracting appropriate tree paths and constructing PUF-trees for subsequent projected databases, PUF-growth finds all potentially frequent patterns at the end of the second scan of the probabilistic dataset of uncertain data. As these potentially frequent patterns include all truly frequent patterns and some infrequent patterns

Fig. 14.6 The PUF-tree for the probabilistic dataset D_2 of uncertain data



(i.e., some *false positives*), PUF-growth then quickly scans the dataset a third time to check each of them to verify whether or not they are truly frequent (i.e., prune false positives). As illustrated by Table 14.6, the prefixed item caps tighten the upper bound to the expected support of non-singleton patterns (when compared with the transaction caps in the CUF-tree). Consequently, the number of false positives that need to be examined by PUF-growth during the third scans of the probabilistic dataset of uncertain data is usually smaller than that by CUF-growth. Hence, PUF-growth runs faster than CUF-growth.

6 Constrained Uncertain Frequent Pattern Mining

Recall from Sect. 5 that the UF-growth, UFP-growth, CUF-growth and PUF-growth algorithms are useful in finding *all* the frequent patterns from probabilistic datasets of uncertain data in many situations. However, there are other situations in which users are interested in only *some* of the frequent patterns. In these situations, users express their interest in terms of constraints. This leads to *constrained mining* [20, 27, 30, 34, 35]. In response, Leung et al. [33, 43] extended the UF-growth algorithm to mine probabilistic datasets of uncertain data for frequent patterns that satisfy user-specified constraints. The two resulting algorithms, called **U-FPS** [33] and **U-FIC** [43], push the constraints in the mining process and exploit properties of different kinds of constraints (instead of a naive approach of first mining all frequent patterns and then pruning all uninteresting or invalid ones).

U-FPS exploits properties of (two types of) succinct constraints [31]. More specifically, by exploiting that “all patterns satisfying any *succinct and anti-monotone (SAM) constraint* C_{SAM} must comprise only items that individually satisfy C_{SAM} ”, U-FPS stores only these items in the UF-tree when handling C_{SAM} . Similarly, by exploiting that “all patterns satisfying any *succinct but not anti-monotone (SUC) constraint* C_{SUC} consist of at least one item that individually satisfies C_{SUC} and may contain other items”, U-FPS partitions the domain items into two groups (one group contains items individually satisfying C_{SUC} and another group contains those not) and stores items belonging to each group separately in the UF-tree. See Fig. 14.7.

Table 14.6 Sample patterns mined from the probabilistic dataset D_2

Pattern X	Upper bound to $expSup(X, D_2)$ based on transaction cap	Upper bound to $expSup(X, D_2)$ based on prefixed item cap	$expSup(X, D_2)$
$\{a, c\}$	1.62	1.44	1.16
$\{a, d\}$	0.96	0.84	0.84
$\{b, d\}$	0.96	0.96	0.79
$\{c, d\}$	0.54	0.54	0.54
$\{a, c, e\}$	0.72	0.27	0.22

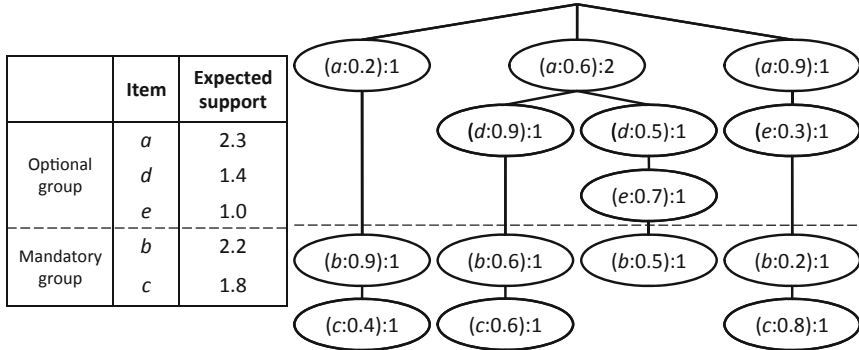


Fig. 14.7 The UF-tree for mining constrained frequent patterns from D_2

As arranging domain items in decreasing order of their support in the original FP-tree is just a heuristic, U-FIC exploits properties of (two types of) convertible constraints [29] and arranges the domain items in the UF-tree according to some monotonic order of attribute values relevant to the constraints. By doing so, U-FIC does not need to perform constraint checking against any extensions of patterns satisfying any *convertible monotone (COM) constraint* C_{COM} because all these extensions are guaranteed to satisfy C_{COM} . Similarly, U-FIC prunes all the patterns that violate any *convertible anti-monotone (CAM) constraint* C_{CAM} because these patterns and their extensions are guaranteed to violate C_{CAM} . By exploiting the user-specified constraints, computation of both U-FPS and U-FIC is proportional to the selectivity of the constraints.

7 Uncertain Frequent Pattern Mining from Big Data

As technology advances further, high volumes of valuable data—such as banking, financial, and marketing data—are generated in various real-life business applications in modern organizations and society. This leads us into the new era of Big Data [45]. Intuitively, *Big Data* are interesting high-velocity, high-value, and/or high-variety data with volumes beyond the ability of commonly-used software to capture, manage, and process within a tolerable elapsed time. Hence, new forms of processing data

are needed to enable enhanced decision making, insight, knowledge discovery, and process optimization. To handle Big Data, researchers proposed the use of a high-level programming model—called *MapReduce*—to process high volumes of data by using parallel and distributed computing [54] on large clusters or grids of nodes (i.e., commodity machines), which consist of a master node and multiple worker nodes. As implied by its name, MapReduce involves two key functions: “map” and “reduce”. An advantage of using the Map-Reduce model is that users only need to focus on (and specify) these “map” and “reduce” functions—without worrying about implementation details for (i) partitioning the input data, (ii) scheduling and executing the program across multiple machines, (iii) handling machine failures, or (iv) managing inter-machine communication.

To mine frequent patterns from Big probabilistic datasets of uncertain data, Leung and Hayduk [37] proposed the **MR-growth** algorithm. The algorithm uses Map-Reduce—by applying two sets of the “map” and “reduce” functions—in a pattern-growth environment. Specifically, the master node reads and divides a probabilistic dataset D of uncertain data into partitions, and then assigns them to different worker nodes. The worker node corresponding to each partition P_j (where $D = \bigcup_j P_j$) then outputs a pair consisting of an item x and its existential probability $P(x, t_i)$ —i.e., $\langle x, P(x, t_i) \rangle$ —for every item x in transaction t_i assigned to P_j as intermediate results:

$$\begin{aligned} \text{map:} & \langle \text{ID of } t_i \text{ in } P_j, \text{ contents of } t_i \rangle \\ \mapsto & \text{list of } \langle x \in t_i, P(x, t_i) \rangle. \end{aligned} \quad (14.6)$$

Afterwards, these $\langle x, P(x, t_i) \rangle$ pairs in the list (i.e., intermediate results) are shuffled and sorted (e.g., grouped by x). Each worker node then executes the “reduce” function, which (i) “reduces”—by summing—all the $P(x, t_i)$ values for each item x so as to compute its expected support $\text{expSup}(\{x\}, D)$ and (ii) outputs $\langle \{x\}, \text{expSup}(\{x\}, D) \rangle$ (representing a frequent 1-itemset $\{x\}$ and its expected support) if $\text{expSup}(\{x\}, D) \geq \text{minsup}$:

$$\begin{aligned} \text{reduce:} & \langle x, \text{list of } P(x, t_i) \rangle \\ \mapsto & \text{list of } \langle \text{frequent 1-itemset } \{x\}, \text{expSup}(\{x\}, D) \rangle, \end{aligned} \quad (14.7)$$

where $\text{expSup}(\{x\}, D) = \text{sum of } P(x, t_i) \text{ in the list for an item } x$.

Afterwards, MR-growth rereads the datasets to form a $\{x\}$ -projected database (i.e., a collection of transactions containing x) for each item x in the list produced by the first reduce function (i.e., for each frequent 1-itemset $\{x\}$). The worker node corresponding to each projected database then (i) builds appropriate local UF-trees (based on the projected database assigned to the node) to mine frequent k -itemsets (for $k \geq 2$) and (ii) outputs $\langle X, \text{expSup}(X, D) \rangle$ (which represents a frequent k -itemset X and its expected support) if $\text{expSup}(X, D) \geq \text{minsup}$. In other words, MR-growth executes the second set of “map” and “reduce” functions as follows:

$$\begin{aligned} \text{map:} & \langle \text{ID of } t_i \text{ in } P_j, \text{ contents of } t_i \rangle \\ \mapsto & \text{list of } \langle \{x\}, \{x\}\text{-proj. DB} \rangle; \end{aligned} \quad (14.8)$$

Table 14.7 Streaming uncertain data D_3

Batch ID	Set of items with existential probability
B_1	$t_1 = \{f:0.7\},$ $t_2 = \{f:0.7, g:0.9, h:0.7, i:0.5\},$ $t_3 = \{f:0.7, g:0.9, h:0.8\}$
B_2	$t_4 = \{f:0.7, g:0.9\},$ $t_5 = \{g:0.6\},$ $t_6 = \{g:0.6\}$
B_3	$t_7 = \{f:0.7, g:0.9, h:0.7, i:0.5\},$ $t_8 = \{f:0.7, g:0.9, h:0.8\},$ $t_9 = \{f:0.7, g:0.9, i:0.5\}$

and

$$\begin{aligned} \text{reduce: } & \langle \{x\}, \{x\}\text{-proj. DB} \rangle \\ \mapsto & \text{list of } \langle \text{frequent } k\text{-itemset } X, \text{expSup}(X, D) \rangle. \end{aligned} \quad (14.9)$$

To recap, by using the above two sets of “map” and “reduce” functions, the MR-growth (i) first finds all frequent 1-itemsets with their expected support and (ii) then builds appropriate local UF-trees (for projected databases) to find all frequent k -itemsets (for $k \geq 2$) with their expected support.

8 Streaming Uncertain Frequent Pattern Mining

In addition to *static* probabilistic datasets of uncertain data, *dynamic streams* of uncertain data can also be generated (e.g., by wireless sensors) in many real-life applications (e.g., environment surveillance). This leads to stream mining [22].

8.1 SUF-growth

To mine frequent patterns from streams of uncertain data, Leung and Hao [36] extended the UF-growth algorithm (ref. Sect. 5.1) to produce an exact mining algorithm called **SUF-growth**. During the mining process, the (i) sliding window model, (ii) time-fading model, or (iii) landmark model is commonly used in processing batches of transactions in the data streams. When using a sliding window model, SUF-growth captures only the contents of streaming data in batches of transactions belonging to the current window (that captures the recent w batches) in a tree structure called **SUF-tree**. When the window slides, SUF-growth removes from the SUF-tree those data belonging to older batches and adds to the SUF-tree those data belonging to newer batches. Hence, each tree node in the SUF-tree consists of three components: (i) an item, (ii) its existential probability, and (iii) a list of its w occurrence counts in the path. By doing so, when the window slides, the oldest occurrence counts (representing the oldest streaming data) are replaced by the newest occurrence

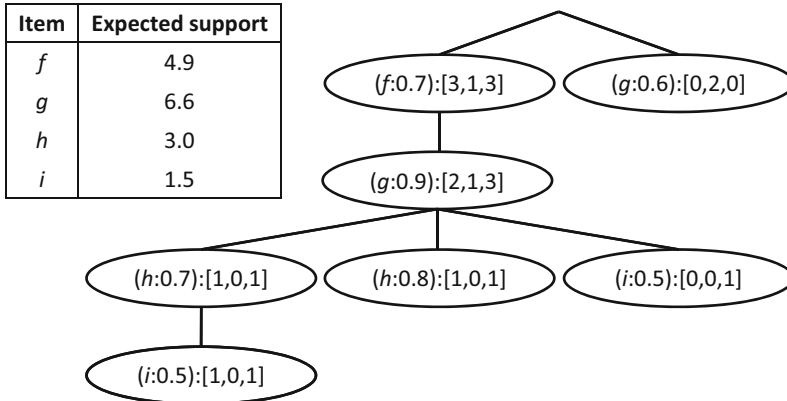


Fig. 14.8 The SUF-tree for streaming uncertain data D_3

counts (representing the newest streaming data). The SUF-tree is constructed in a similar fashion as the construction of the UF-tree, except that the occurrence count is inserted as the newest entry in the list of occurrence counts. Figure 14.8 shows an example of a SUF-tree constructed from the streaming uncertain data in Table 14.7.

Once the SUF-tree is constructed, it is always kept up-to-date when the window slides. As the window continues to slide in the dynamic streams, SUF-growth delays the mining of frequent patterns from uncertain streaming data until the user requests for the patterns. At that time, SUF-growth mines the up-to-dated SUF-tree in a fashion similar to UF-growth with the user-specified *minsup* to find all frequent patterns.

8.2 UF-streaming for the Sliding Window Model

Besides the SUF-growth algorithm, Leung and Hao [36] also extended the UF-growth algorithm to produce an approximate algorithm called **UF-streaming**. Unlike SUF-growth (which is an exact algorithm but uses a “delayed” mining mode), UF-streaming is an approximate algorithm that uses an “immediate” mining mode. Specifically, for every incoming batch of streaming uncertain data, UF-streaming applies UF-growth (ref. Sect. 5.1) to that batch with a *preMinsup* threshold (where *preMinsup* < *minsup*) to find “frequent” patterns (or more precisely, sub-frequent or potentially frequent patterns due to the use of the *preMinsup* threshold). These “frequent” patterns having expected support \geq *preMinsup* are then stored in a tree structure called **UF-stream**. Each tree path represents a “frequent” pattern. Each tree node stores a list of w expected support values (one for each batch), where the i -th value indicates the expected support of that “frequent” pattern in the i -th batch. When a new batch flows in, the window slides, and the algorithm shifts the w expected support values of each node in the UF-stream structure so as to ensure that it always captures the “frequent” patterns mined from the w most recent batches of streaming uncertain data. Figure 14.9 shows an example of a UF-stream structure

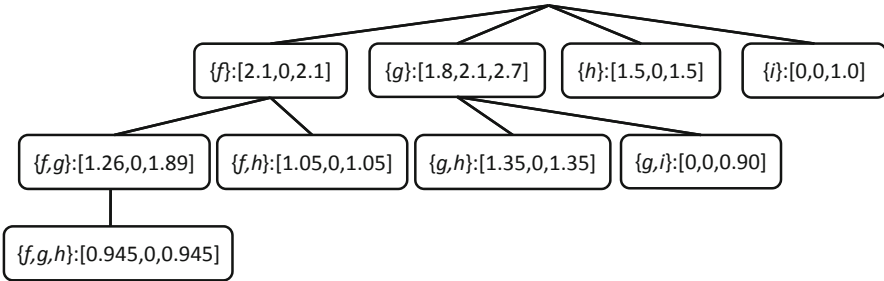


Fig. 14.9 The UF-stream structure for streaming uncertain data D_3

constructed for the streaming uncertain data in Table 14.7 with $preMinsup = 0.9 < 1.0 = minsup$.

As UF-streaming uses $preMinsup$, the UF-stream structure may contain some false positives (i.e., every pattern Y having $preMinsup \leq expSup(Y, D_w) < minsup$, where D_w represents the streaming uncertain data in the current sliding window) in addition to all truly frequent patterns (i.e., every pattern X having $expSup(X, D_w) \geq minsup$). At the time when the user requests for frequent patterns from uncertain streaming data, UF-streaming traverses the UF-stream structure and returns only those truly frequent patterns.

8.3 TUF-streaming for the Time-Fading Model

Besides the sliding window model (used by both SUF-growth and UF-streaming), there are also other stream processing models such as time-fading and landmark models. Leung and Jiang [38] proposed the **TUF-streaming** algorithm to mine frequent patterns from probabilistic datasets of uncertain data in a fashion similar to UF-streaming, except that TUF-streaming uses the *time-fading model* instead of the sliding window model. When using the time-fading model, TUF-streaming puts heavier weights on recent batches of streaming uncertain data than older batches. Specifically, like UF-streaming, the TUF-streaming algorithm also uses the “immediate” mining mode to apply UF-growth (ref. Sect. 5.1) to every incoming batch of streaming uncertain data with a $preMinsup$ threshold to find “frequent” patterns from that batch. When using the time-fading model, the corresponding **TUF-stream** structure (i) captures all “frequent” patterns mined from all batches but weights recent batches heavier than older batches (i.e., monotonically decreasing weights from recent to older data). See Fig. 14.10, in which α (where $0 < \alpha \leq 1$) is a time-fading factor.

While TUF-streaming handles batches of streaming uncertain data that come in order, there are situations where batches may get delayed and thus arrived out of the desired order. To deal with these situations, Jiang and Leung [26] extended the TUF-streaming algorithm to mine frequent patterns from these delayed batches when using the time-fading model.

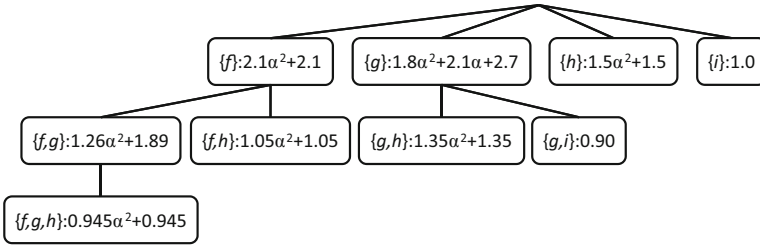
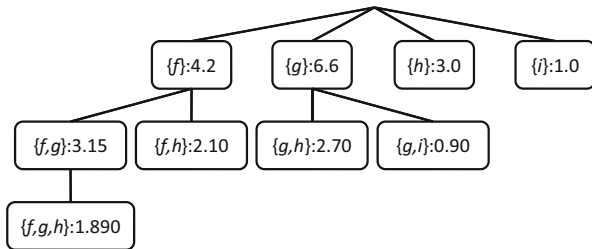


Fig. 14.10 The TUF-stream structure for streaming uncertain data D_3

Fig. 14.11 The LUF-stream structure for streaming uncertain data D_3



8.4 LUF-streaming for the Landmark Model

Moreover, Leung et al. [41] extended the TUF-streaming algorithm to become the **LUF-streaming** algorithm, which mines frequent patterns from streaming uncertain data in a fashion similar to the UF-streaming and TUF-streaming algorithms, except that LUF-streaming uses the *landmark model*. When using the landmark model, the corresponding **LUF-stream** structure (i) captures all “frequent” patterns mined from all batches of streaming uncertain data generated from a landmark to the present time and (ii) treats all batches with the same importance. See Fig. 14.11.

8.5 Hyperlinked Structure-Based Streaming Uncertain Frequent Pattern Mining

So far, we have described notable exact and approximate *tree-based* streaming uncertain frequent pattern mining algorithms. Besides them, there are also hyperlinked structure based algorithms. For instance, Nadungodage et al. [46] proposed two false positive-oriented algorithms called **UHS-Stream** and **TFUHS-Stream**. The UHS-Stream algorithm applies uncertain hyperlinked structure stream mining for finding all the frequent patterns seen up to the current moment (i.e., with the landmark model). Similarly, the TFUHS-Stream algorithm applies uncertain hyperlinked structure stream mining, but it uses the time-fading model. Hence, the TFUHS-Stream algorithm puts heavier weights on the recent transactions than historical data in the stream.

9 Vertical Uncertain Frequent Pattern Mining

The aforementioned candidate generate-and-test based, hyperlinked structure based, as well as tree-based mining algorithms use *horizontal mining*, for which a dataset can be viewed as a collection of transactions. Each transaction is a set of items. Alternatively, *vertical mining* can be applied, for which each dataset can be viewed as a collection of items and their associated lists (or sets or vectors) of transaction IDs (i.e., tIDs). Each list of tID of an item x represents all the transactions containing x . With this vertical representation of datasets, the support of a pattern X can be computed by intersecting the lists of tIDs of items within X .

9.1 U-Eclat: An Approximate Algorithm

To mine frequent patterns using the vertical representation of probabilistic datasets of uncertain data, Calders et al. [17] instantiated “possible worlds” of the datasets to get instantiated samples (in which data become precise) and then applied the Eclat algorithm [55] to each of these samples of instantiated databases. The resulting algorithm is called **U-Eclat**. Given a probabilistic dataset D of uncertain data, U-Eclat generates an independent random number r for each item x in a transaction t_i . If the existential probability $P(x, t_i)$ of item x in transaction t_i is no less than such a random number r (i.e., $P(x, t_i) \geq r$), then x is instantiated and included in a “precise” sampled database, which is then mined using the original Eclat algorithm. This sampling and instantiation process is repeated multiple times, and thus generates multiple sampled “precise” databases. The estimated support of any pattern X is the average support of X over the multiple sampled databases. Figure 14.12 shows three sampled databases for probabilistic dataset D_2 . As a sampling-based algorithm, U-Eclat gains efficiency but loses accuracy. More instantiations (i.e., more samples) helps improve accuracy, but it comes at the cost of an increase in execution time.

9.2 UV-Eclat: An Exact Algorithm

Alternatively, to avoid instantiations and to *directly* mine frequent patterns using the vertical representation of probabilistic datasets containing uncertain data, Budhia et al. [16] proposed the **UV-Eclat** algorithm. When mining uncertain data, in addition to recording which transactions contain x in the set of tIDs (i.e., *tIDsets*), it is also important to capture additional information: If x is likely to be present in a transaction t_i , then its associated existential probability $P(x, t_i)$ —which expresses the likelihood of x appearing in transaction t_i of the dataset—needs to be captured.

As it would be impractical to build a tIDset for each distinct (domain item, existential probability value) pair due to the huge number of such pairs, UV-Eclat builds a tIDset for each domain item x instead. In each set, UV-Eclat augments every tID (representing t_i) with its corresponding existential probability value $P(x, t_i)$. In

Probabilistic database D_2 of uncertain data:

$$D_2 = \begin{cases} t_1 = \{a:0.2, b:0.9, c:0.4\}, \\ t_2 = \{a:0.6, b:0.6, c:0.6, d:0.9\}, \\ t_3 = \{a:0.6, b:0.5, d:0.5, e:0.7\}, \\ t_4 = \{a:0.9, b:0.2, c:0.8, e:0.3\} \end{cases}$$

Samples of instantiated possible worlds for D_2 containing uncertain data:

	Sample 1					Sample 2					Sample 3				
Item	a	b	c	d	e	a	b	c	d	e	a	b	c	d	e
tID list	{t ₁ , t ₂ , t ₃ , t ₄ }	{t ₁ , t ₂ }	{t ₂ , t ₄ }	{}	{t ₃ }	{t ₁ , t ₂ , t ₄ }	{t ₁ , t ₂ }	{t ₁ , t ₂ , t ₄ }	{t ₂ , t ₃ }	{t ₃ }	{t ₂ , t ₄ }	{t ₁ , t ₂ , t ₄ }	{t ₂ , t ₄ }	{t ₂ }	{t ₄ }

Observations:

- $expSup(\{e\}, D_2) = 1.0$ for D_2 vs. $avg(sup(\{e\}, Sample S)) = 1.0$ over the 3 samples
- $expSup(\{a, c\}, D_2) = 1.16$ for D_2 vs. $avg(sup(\{a, c\}, Sample S)) \approx 2.33$ over the 3 samples
- $expSup(\{b, d\}, D_2) = 0.79$ for D_2 vs. $avg(sup(\{b, d\}, Sample S)) \approx 0.67$ over the 3 samples where $S = 1, 2, 3$

Fig. 14.12 Samples of instantiated “possible worlds” for D_2

Table 14.8 Augmented tIDsets for domain items in the probabilistic dataset D_2

Item	Augmented tIDset
a	{t ₁ :0.2, t ₂ :0.6, t ₃ :0.6, t ₄ :0.9}
b	{t ₁ :0.9, t ₂ :0.6, t ₃ :0.5, t ₄ :0.2}
c	{t ₁ :0.4, t ₂ :0.6, t ₄ :0.8}
d	{t ₂ :0.9, t ₃ :0.5}
e	{t ₃ :0.7, t ₄ :0.3}

other words, the resulting augmented tIDset for any item x is of the form $\{t_i: P(x, t_i)\}$, which is equivalent to $\{t_i: expSup(\{x\}, t_i)\}$. See Table 14.8.

With the use of augmented tIDsets to vertically represent the probabilistic dataset D of uncertain data, the expected support of any 1-itemset $\{x\}$ in D can be computed by summing all $P(x, t_i)$ values in the augmented tIDset for $\{x\}$. The tIDset of any $(k+1)$ -itemset $X \equiv Y \cup \{z\}$ (where Y is a k -pattern and z is an item) for $k \geq 1$ can be formed by intersecting the tIDsets of Y and $\{z\}$. Each t_i in the intersection result is associated with an expected support value $expSup(X, t_i)$, which is the product of $expSup(Y, t_i)$ and $P(z, t_i)$.

9.3 U-VIPER: An Exact Algorithm

Vertical representations for a probabilistic dataset D of uncertain data are not confined to set-based representations (e.g., augmented tIDsets used in UV-Eclat). There are

Table 14.9 Vectors for domain items in the probabilistic dataset D_2

Vector for every domain item				
$\vec{a} = \begin{pmatrix} 0.2 \\ 0.6 \\ 0.6 \\ 0.9 \end{pmatrix}$,	$\vec{b} = \begin{pmatrix} 0.9 \\ 0.6 \\ 0.5 \\ 0.2 \end{pmatrix}$,	$\vec{c} = \begin{pmatrix} 0.4 \\ 0.6 \\ 0 \\ 0.8 \end{pmatrix}$,	$\vec{d} = \begin{pmatrix} 0 \\ 0.9 \\ 0.5 \\ 0 \end{pmatrix}$,	and $\vec{e} = \begin{pmatrix} 0 \\ 0 \\ 0.7 \\ 0.3 \end{pmatrix}$.

other representations. For instance, Leung et al. [44] proposed the **U-VIPER** algorithm, in which D is vertically represented by a collection of fixed-size vectors—one for each domain item x . The length of each vector is fixed and is equal to the number of transactions (i.e., $|D| = n$) in the probabilistic dataset of uncertain data. When mining uncertain data, in addition to using a Boolean value (say, 0 or 1) to denote whether or not transaction t_i contains x in the vector, it is also important to capture additional information: If x is likely to be present in a transaction t_i , then its associated existential probability $P(x, t_i)$ —which expresses the likelihood of x appearing in transaction t_i of the dataset—needs to be captured.

As it would be a waste of space to augment $P(x, t_i)$ to the Boolean value “1” for t_i (i.e., the i -th element of the vector for x), U-VIPER replaces the Boolean value “1” by $P(x, t_i)$ as the i -th element of each vector \vec{x} representing domain item x . Specifically, the i -th element of \vec{x} (denoted as $\vec{x}[i]$) stores (i) “0” if x is absent from t_i and (ii) $P(x, t_i)$ if x is likely to be present in t_i :

$$\begin{aligned}
 \text{\textit{i}-th element of } \vec{x} \text{ (i.e., } \vec{x}[i]) &= \textit{expSup}(\{x, t_i\}) \\
 &= \begin{cases} 0 & \text{if } x \notin t_i \\ P(x, t_i) & \text{if } x \in t_i \end{cases} \quad (14.10)
 \end{aligned}$$

See Table 14.9. With this vector-based representation, the expected support of any 1-itemset $\{x\}$ can be computed by summing all non-zero $P(x, t_i)$ values in \vec{x} (i.e., taking the L_1 -norm of \vec{x}):

$$\begin{aligned}
 \textit{expSup}(\{x\}, D) &= \sum_{i=1}^n \textit{expSup}(\{x, t_i\}) \\
 &= \sum_{i=1}^n P(x, t_i) \\
 &= \sum_{i=1}^n \vec{x}[i] \\
 &= \|\vec{x}\|_1 \quad (14.11)
 \end{aligned}$$

The i -th element of the vector of any $(k+1)$ -itemset $X \equiv Y \cup \{z\}$ (where Y is a k -itemset and z is an item) for $k \geq 1$ can be formed by taking the product of $\textit{expSup}(Y, t_i)$ and $P(z, t_i)$. The expected support of X is then the *dot product* of \vec{Y} and $\vec{\{z\}}$.

10 Discussion on Uncertain Frequent Pattern Mining

So far, we have described various uncertain frequent pattern mining algorithms. Tong et al. [50] compared some of these algorithms.

In terms of functionality, the U-Apriori, UH-mine, UF-growth, UFP-growth, CUF-growth, PUF-growth, MR-growth, U-Eclat, UV-Eclat and U-VIPER algorithms all mine static datasets of uncertain data. Among them, the first seven mine the datasets horizontally, whereas the remaining three algorithms mine the datasets vertically. In contrast, the SUF-growth, UF-streaming, TUF-streaming, LUF-streaming, UHS-Stream and TFUHS-Stream algorithms mine dynamic streaming uncertain data. Unlike these 16 algorithms that find all frequent patterns, both U-FPS and U-FIC algorithms find only those frequent patterns satisfying the user-specified constraints.

In terms of accuracy, most algorithms return all the patterns with expected support (over all “possible worlds”) meeting or exceeding the user-specified threshold *minsup*. In contrast, U-Eclat returns patterns with estimated support (over only the sampled “possible worlds”) meeting or exceeding *minsup*. Hence, U-Eclat may introduce false positives (when the support is overestimated) or false negatives (when the support is underestimated). More instantiations (i.e., more samples) helps improve accuracy.

In terms of memory consumption, U-Apriori keeps a list of candidate patterns, whereas the tree-based and hyperlinked structure based algorithms construct in-memory structures (e.g., UF-tree and its variants, extended H-struct). On the one hand, a UF-tree is more compact (i.e., requires less space) than the extended H-struct. On the other hand, UH-mine keeps only one extended H-struct, whereas tree-based algorithms usually construct more than one tree. Sizes of the trees may also vary. For instance, when U-FPS handles a succinct and anti-monotone constraint C_{SAM} , the tree size depends on the selectivity of C_{SAM} because only those items that individually satisfy C_{SAM} are stored in the UF-tree. Both CUF-tree and PUF-tree (for uncertain data) can be as compact as a FP-tree (for precise data). When SUF-growth handles streams, the tree size depends on the size of sliding window (e.g., a window of w batches) because a list of w occurrence counts is captured in each node of SUF-trees (cf. only one occurrence count is captured in each node of UF-trees). Moreover, when items in probabilistic datasets take on a few distinct existential probability values, the trees contain fewer nodes (cf. the number of distinct existential probability values does not affect the size of candidate lists or the extended H-struct). Furthermore, *minsup* and density also affect memory consumption. For instance, for a sparse dataset called kosarak, different winners (requiring the least space) have been shown for different *minsup*: U-Apriori when $minsup < 0.15\%$, UH-mine when $0.15\% \leq minsup < 0.5\%$, and, UF-growth when $0.5\% \leq minsup$; for a dense dataset called connect4, UH-mine was the winner for $0.2\% \leq minsup < 0.8\%$.

In terms of performance, most algorithms perform well when items in probabilistic datasets take on low existential probability values because these datasets do not lead to long frequent patterns. When items in probabilistic datasets take on high existential probability values, more candidates are generated-and-tested by U-Apriori, more and

bigger UF-trees are constructed by UF-growth, more hyperlinks are adjusted by UH-mine, and more estimated supports are computed by U-Eclat. Hence, longer runtimes are required. Similarly, when *minsup* decreases, more frequent patterns are returned and longer runtimes are also required. Moreover, the density of datasets also affects runtimes. For instance, when datasets are dense (e.g., connect4 augmented with existential probability), UF-trees obtain higher compression ratios and thus require less time to traverse than sparse datasets (e.g., kosarak augmented with existential probability). Some experimental results showed the following: (i) datasets with a low number of distinct existential probabilities led to smaller UF-trees and shorter runtimes for UF-growth (than U-Apriori); (ii) U-Apriori requires shorter runtimes than UH-mine when *minsup* was low (e.g., *minsup* < 0.3 % for kosarak, *minsup* < 0.6 % for connect4) but vice versa when *minsup* was high; (iii) depending on the number of samples, U-Eclat could take longer or shorter to run than U-Apriori.

11 Extension: Probabilistic Frequent Pattern Mining

The aforementioned algorithms all find (expected support-based) frequent patterns from uncertain data. These are patterns with expected support meeting or exceeding the user-specified threshold *minsup*. Note that expected support of a pattern X provides users with frequency information of X summarized over all “possible worlds”, but it does not reveal the confidence on the likelihood of X being frequent (i.e., percentage of “possible worlds” in which X is frequent). However, knowing the confidence can be helpful in some applications. Hence, in recent years, there is also algorithmic development on extending the notion of frequent patterns based on expected support to useful patterns—such as probabilistic heavy hitters and probabilistic frequent patterns as described below.

11.1 Mining Probabilistic Heavy Hitters

Although the expected support of an item x (i.e., a singleton pattern $\{x\}$) provides users with an estimate of the frequency of x in many real-life applications, it is also helpful to know the confidence on the likelihood of x being frequent in the uncertain data in some other applications. Hence, Zhang et al. [56] formalized the notion of probabilistic heavy hitters (i.e., *probabilistic frequent items*, which are also known as probabilistic frequent singleton patterns) following the “possible world” semantics [21] for probabilistic datasets of uncertain data.

Definition 14.4 Given (i) a probabilistic dataset D of uncertain data, (ii) a user-specified support threshold ϕ , and (iii) a user-specified frequentness probability threshold τ , the problem of **mining probabilistic heavy hitters from uncertain data** is to find all (ϕ, τ) -probabilistic heavy hitters (PHHs). An item x is a (ϕ, τ) -probabilistic heavy hitter (PHH) if $P(\text{sup}(x, W_j) > \phi | W_j) > \tau$ (where $\text{sup}(x, W_j)$ is the support of x in a random possible world W_j and $|W_j|$ is the number of items

Table 14.10 An example of a probabilistic dataset D_4 with mutually exclusive items

Transaction ID	Set of items with existential probability
t_1	$\{j_1:0.5, j_2:0.5\}$
t_2	$\{k_1:0.1, k_2:0.2, k_3:0.3, k_4:0.4\}$
t_3	$\{j_1:0.3, j_2:0.3, j_3:0.1\}$

Observation 1: The sum of existential probability of all items in each transaction is bounded above by 1.

Observation 2: There are 60 “possible worlds” for D_4 (cf. 512 “possible worlds” if items in each transaction were independent (i.e., not mutually exclusive)).

in W_j), which represents the probability of x being frequent exceeding the user expectation.

Equivalently, given (i) a probabilistic dataset D of uncertain data, (ii) a user-specified support threshold $minsup$, (iii) a user-specified frequentness probability threshold $minProb$, the research problem of **mining probabilistic heavy hitters (PHHs) from uncertain data** is to find every item x that is highly likely to be frequent—i.e., the probability that x occurs in at least $minsup$ transactions in D is no less than $minProb$. In other words, x is a probabilistic heavy hitter if $P(sup(x, D) \geq minsup) > minProb$.

To find these PHHs from a probabilistic dataset of uncertain data where items in each transaction are mutually inclusive (e.g., D_4 in Table 14.10), Zhang et al. [56] proposed an exact algorithm and an approximate algorithm. The exact algorithm uses dynamic programming to mine offline uncertain data for PHHs. Such an algorithm runs in polynomial time when there is sufficient memory. When the memory is limited, the approximate algorithm uses sampling techniques to mine streaming uncertain data for approximate PHHs.

11.2 Mining Probabilistic Frequent Patterns

The expected support of a pattern X (that consists of one or more items) provides users with an estimate of the frequency of X , but it does not take into account the variance or the probability distribution of the support of X . In some applications, knowing the confidence on which pattern is highly likely to be frequent helps interpreting patterns mined from uncertain data. Hence, Bernecker et al. [15] extended the notion of frequent patterns and introduced the research problem of mining probabilistic frequent patterns (p-FPs). Figure 14.11 illustrates the differences between expected support and probabilistic support.

Definition 14.5 Given (i) a probabilistic dataset D of uncertain data, (ii) a user-specified support threshold $minsup$, (iii) a user-specified frequentness probability threshold $minProb$, the research problem of **mining probabilistic frequent patterns (p-FPs) from uncertain data** is to find (i) all patterns that are highly likely to be frequent and (ii) their support. Here, the support $sup(X, D)$ of any pattern X is defined by a discrete probability distribution function (pdf) or probability mass

Table 14.11 Frequent pattern vs. probabilistic frequent patterns

Let user-specified $minsup = 1.0$ and $minProb = 0.90$

$sup(\{d\})$	Set of items	$Prob(W_j)$	COUNT(W_j)
2	$(d:0.9) \in t_2 \wedge (d:0.5) \in t_3$	0.45	8192
	$(d:0.9) \in t_2 \wedge (d:0.5) \notin t_3$	0.45	8192
1	$(d:0.9) \notin t_2 \wedge (d:0.5) \in t_3$	+0.05	+8192
		= 0.50	= 16384
0	$(d:0.9) \notin t_2 \wedge (d:0.5) \notin t_3$	0.05	8192
		$\sum_j Prob(W_j) = 1$	$\sum_j COUNT(W_j) = 32768$

As $expSup(\{d\}, D_2) = (2 \times 0.45) + (1 \times 0.50) + (0 \times 0.05) = 0.9 + 0.5 = 1.4 \geq minsup$, $\{d\}$ is an expected support-based **frequent pattern**.

As $Prob(sup(\{d\}, D_2) \geq minsup) = 0.45 + 0.50 = 0.95 \geq minProb$, $\{d\}$ is also a **probabilistic frequent pattern**.

$sup(\{e\})$	Set of items	$Prob(W_j)$	COUNT(W_j)
2	$(e:0.7) \in t_3 \wedge (e:0.3) \in t_4$	0.21	8192
	$(e:0.7) \in t_3 \wedge (e:0.3) \notin t_4$	0.49	8192
1	$(e:0.7) \notin t_3 \wedge (e:0.3) \in t_4$	+0.09	+8192
		= 0.58	= 16384
0	$(e:0.7) \notin t_3 \wedge (e:0.3) \notin t_4$	0.21	8192
		$\sum_j Prob(W_j) = 1$	$\sum_j COUNT(W_j) = 32768$

As $expSup(\{e\}, D_2) = (2 \times 0.21) + (1 \times 0.58) + (0 \times 0.21) = 0.7 + 0.3 = 1.0 \geq minsup$, $\{e\}$ is an expected support-based **frequent pattern**.

However, as $Prob(sup(\{e\}, D_2) \geq minsup) = 0.21 + 0.58 = 0.79 < minProb$, $\{e\}$ is *not* a probabilistic frequent pattern.

function (pmf). A pattern X is highly likely to be frequent (i.e., X is a probabilistic frequent pattern) if and only if its frequentness probability is no less than $minProb$, i.e., $P(sup(X, D) \geq minsup) \geq minProb$. The frequentness probability of X is the probability that X occurs in at least $minsup$ transactions of D . (Table 14.11)

Note that frequentness probability is anti-monotonic: All subsets of a p-FP are also p-FPs. Equivalently, if X is not a p-FP, then none of its supersets is a p-FP, and thus all of them can be pruned. Moreover, when $minsup$ increases, frequentness probabilities of p-FPs decrease.

Bernecker et al. [15] used a dynamic computation technique in computing the probability function $f_X(k) = P(sup(X, D) = k)$, which returns the probability that the support of a pattern X equals to k . Summing the values of such a probability function $f_X(k)$ over all $k \geq minsup$ gives the frequentness probability of X because

$$\sum_{k \geq minsup}^{|D|} f_X(k) = \sum_{k \geq minsup}^{|D|} P(sup(X, D) \geq minsup).$$

Any pattern X having the sum no less than $minProb$ becomes a p-FP.

Sun et al. [49] proposed the top-down inheritance of support probability function (TODIS) algorithm, which runs in conjunction with a divide-and-conquer (DC) approach, to mine probabilistic frequent patterns from uncertain data by extracting

patterns that are supersets of p-FPs and deriving p-FPs in a top-down manner (i.e., descending cardinality of p-FPs).

To accelerate probabilistic frequent pattern mining, Wang et al. [51] applied a model-based approach that supports both tuple uncertainty (as in the TODIS algorithm [49]) and attribute uncertainty (as in the aforementioned dynamic computation technique [15]). Specifically, they represented the support pmf of a p-FP as some existing probability models (e.g., Poisson binomial distribution model, normal distribution). By doing so, Wang et al. quickly found two types of p-FPs: (i) threshold-based p-FP (i.e., $P(\text{sup}(X) \geq \text{minsup}) \geq \text{minProb}$) and (ii) rank-based p-FP (e.g., top- k p-FP).

In addition to finding p-FPs from static datasets of uncertain data, there are also algorithms that find p-FPs from dynamic streaming uncertain data. For instance, Akbarinia and Massegli [14] proposed an exact algorithm called FMU for fast mining of streaming uncertain data with the sliding window model.

12 Conclusions

Frequent pattern mining is an important data mining task. It helps discover implicit, previously unknown and potentially useful knowledge; it also helps reveal sets of frequently co-occurring items in numerous real-life applications (e.g., bundles of books that are frequently bought together, collections of courses that are taken in the same academic terms, events that are often co-located, groups of individuals that have common interests). Here, it has drawn the attention of many researchers over the past two decades. The research problem of frequent pattern mining was originally proposed to analyze shoppers' market basket transaction databases containing precise data, in which the contents of transactions in the databases are known. Such a research problem also plays an important role in other data mining tasks, such as the mining of interesting or unexpected patterns, sequential mining, associative classification, as well as outlier detection, in various real-life applications. As we are living in an uncertain world, data in many real-life applications are uncertain. Recently, researchers have paid more attention to the mining of frequent patterns from probabilistic datasets of uncertain data.

In this chapter, we presented some recent works on mining frequent patterns from probabilistic datasets of uncertain data. These include candidate generate-and-test based, hyperlinked structure based, tree-based, as well as vertical uncertain frequent pattern mining algorithms. Among them, the U-Apriori algorithm generates candidate patterns and tests if their expected support meets or exceeds a user-specified threshold. To avoid such a candidate generate-and-test approach, both UH-mine and UF-growth algorithms use a pattern-growth mining approach. The UH-mine algorithm keeps a UH-struct, from which frequent patterns are mined; the UF-growth algorithm constructs a UF-tree, from which frequent patterns are mined. The UFP-growth algorithm applies clustering to help reduce the number of nodes in a UFP-tree. The PUF-growth and CUF-growth algorithms respectively construct a PUF-tree and

a CUF-tree, which are more compact than the corresponding UF-tree. Instead of applying horizontal mining, U-Eclat, UV-Eclat and U-VIPER use vertical mining.

Moreover, the UF-growth algorithm has also been extended for constrained mining, Big Data mining, and stream mining. The resulting U-FPS and U-FIC algorithms exploit properties of the user-specified succinct constraints and convertible constraints, respectively, to find all and only those frequent patterns satisfying the constraints from uncertain data. MR-growth uses the MapReduce model for Big Data analytics. Both SUF-growth and UF-streaming use the sliding window model for mining. SUF-growth mines all frequent patterns from a SUF-tree, which captures the contents of the current few batches of streaming uncertain data. The UF-streaming algorithm applies UF-growth to each batch and stores the mining results in the UF-stream structure, from which frequent patterns can be retrieved. The UF-streaming algorithm was extended to become the TUF-streaming and LUF-streaming algorithms, which use the time-fading and landmark models respectively for (tree-based) mining. Similarly, the TFUHS-Stream and UHS-Stream algorithms also use the time-fading and landmark models respectively, but for hyperlinked structure-based mining.

In addition to expected support-based frequent patterns, there are algorithms that mine probabilistic heavy hitters as well as probabilistic frequent patterns.

Future research directions include (i) mining frequent patterns from uncertain data in applications areas such as social network analysis [4, 28], (ii) mining frequent sequences and frequent graphs from uncertain data, as well as (iii) visual analytics of uncertain frequent patterns.

References

1. Abiteboul, S., Kanellakis, P., & Grahne, G. 1987. On the representation and querying of sets of possible worlds. In *Proceedings of the ACM SIGMOD 1987*, pages 34–48.
2. Aggarwal, C.C. 2009. On clustering algorithms for uncertain data. In C.C. Aggarwal (ed.), *Managing and Mining Uncertain Data*, pages 389–406. Springer.
3. Aggarwal, C.C. (ed.) 2009. *Managing and Mining Uncertain Data*. Springer.
4. Aggarwal, C.C. (ed.) 2011. *Social Network Data Analytics*. Springer.
5. Aggarwal, C.C. 2013. *Outlier Analysis*. Springer.
6. Aggarwal, C.C. (ed.) 2013. *Managing and Mining Sensor Data*. Springer.
7. Aggarwal, C.C. & Reddy, C.K. (eds.), *Data Clustering: Algorithms and Applications*. CRC Press.
8. Agrawal, R., & Srikant, R. 1994. Fast algorithms for mining association rules in large databases. In *Proceedings of the VLDB 1994*, pages 487–499. Morgan Kaufmann.
9. Aggarwal, C.C., & Yu, P.S. 2008. Outlier detection with uncertain data. In *Proceedings of the SIAM SDM 2008*, pages 483–493.
10. Aggarwal, C.C., & Yu, P.S. (eds.) 2008. *Privacy-Preserving Data Mining: Models and Algorithms*. Springer.
11. Aggarwal, C.C., & Yu, P.S. 2009. A survey of uncertain data algorithms and applications. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 21(5), pages 609–623.
12. Agrawal, R., Imieliński, T., & Swami, A. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD 1993*, pages 207–216.

13. Aggarwal, C.C., Li, Y., Wang, J., & Wang, J. 2009. Frequent pattern mining with uncertain data. In *Proceedings of the ACM KDD 2009*, pages 29–38.
14. Akbarinia, R., & Massegli, F. 2012. FMU: fast mining of probabilistic frequent itemsets in uncertain data streams. In *Proceedings of the BDA 2012*.
15. Bernecker, T., Kriegel, H.-P., Renz, M., Verhein, F., & Zuefle, A. 2009. Probabilistic frequent itemset mining in uncertain databases. In *Proceedings of the ACM KDD 2009*, pages 119–127.
16. Budhia, B.P., Cuzzocrea, A., & Leung, C.K.-S. 2012. Vertical frequent pattern mining from uncertain data. In *Proceedings of the KES 2012*, pages 1273–1282. IOS Press.
17. Calders, T., Garboni, C., & Goethals, B. 2010. Efficient pattern mining of uncertain data with sampling. In *Proceedings of the PAKDD 2010, Part I*, pages 480–487. Springer.
18. Chui, C.-K., & Kao, B. 2008. A decremental approach for mining frequent itemsets from uncertain data. In *Proceedings of the PAKDD 2008*, pages 64–75. Springer.
19. Chui, C.-K., Kao, B., & Hung, E. 2007. Mining frequent itemsets from uncertain data. In *Proceedings of the PAKDD 2007*, pages 47–58. Springer.
20. Cuzzocrea, A., Leung, C.K.-S., & MacKinnon, R.K. 2014. Mining constrained frequent itemsets from distributed uncertain data. *Future Generation Computer Systems*. Elsevier.
21. Dalvi, N., & Suciu, D. 2004. Efficient query evaluation on probabilistic databases. In *Proceedings of the VLDB 2004*, pages 864–875. Morgan Kaufmann.
22. Gaber, M.M., Zaslavsky, A.B., & Krishnaswamy, S. Mining data streams: a review. *ACM SIGMOD Record*, 34(2), pages 18–26.
23. Green, T., & Tannen, V. 2006. Models for incomplete and probabilistic information. *Bulletin of the Technical Committee on Data Engineering*, 29(1), pages 17–24. IEEE Computer Society.
24. Han, J., Pei, J., & Yin, Y. 2000. Mining frequent patterns without candidate generation. In *Proceedings of the ACM SIGMOD 2000*, pages 1–12.
25. Jiang, B., Pei, J., Tao, Y., & Lin, X. 2013. Clustering uncertain data based on probability distribution similarity. *IEEE Transactions on Knowledge and Data Engineering (TKDE)*, 25(4), pages 751–763.
26. Jiang, F., & Leung, C.K.-S. 2013. Stream mining of frequent patterns from delayed batches of uncertain data. In *Proceedings of the DaWaK 2013*, pages 209–221. Springer.
27. Lakshmanan, L.V.S., Leung, C.K.-S., & Ng, R.T. 2003. Efficient dynamic mining of constrained frequent sets. *ACM Transactions on Database Systems (TODS)*, 28(4), pages 337–389.
28. Lee, W., Leung, C.K.-S., Song, J.J., & Eom, C.S.-H. 2012. A network-flow based influence propagation model for social networks. In *Proceedings of the CGC/SCA 2012*, pages 601–608. IEEE Computer Society (The best paper of SCA 2012).
29. Leung, C.K.-S. 2009. Convertible constraints. In *Encyclopedia of Database Systems*, pages 494–495. Springer.
30. Leung, C.K.-S. 2009. Frequent itemset mining with constraints. In *Encyclopedia of Database Systems*, pages 1179–1183. Springer.
31. Leung, C.K.-S. 2009. Succinct constraints. In *Encyclopedia of Database Systems*, page 2876. Springer.
32. Leung, C.K.-S. 2011. Mining uncertain data. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery (WIDM)*, 1(4), pages 316–329.
33. Leung, C.K.-S., & Brajczuk, D.A. 2009. Efficient algorithms for the mining of constrained frequent patterns from uncertain data. *ACM SIGKDD Explorations*, 11(2), pages 123–130.
34. Leung, C.K.-S., & Brajczuk, D.A. 2009. Mining uncertain data for constrained frequent sets. In *Proceedings of the IDEAS 2009*, pages 109–120. ACM.
35. Leung, C.K.-S., & Brajczuk, D.A. 2010. uCFS₂: an enhanced system that mines uncertain data for constrained frequent sets. In *Proceedings of the IDEAS 2010*, pages 32–37. ACM.
36. Leung, C.K.-S., & Hao, B. 2009. Mining of frequent itemsets from streams of uncertain data. In *Proceedings of the IEEE ICDE 2009*, pages 1663–1670.
37. Leung, C.K.-S., & Hayduk, Y. 2013. Mining frequent patterns from uncertain data with MapReduce for Big Data analytics. In *Proceedings of the DASFAA 2013, Part I*, pages 440–455. Springer.

38. Leung, C.K.-S., & Jiang, F. 2011. Frequent pattern mining from time-fading streams of uncertain data. In *Proceedings of the DaWaK 2011*, pages 252–264. Springer.
39. Leung, C.K.-S., & Tanbeer, S.K. 2012. Fast tree-based mining of frequent itemsets from uncertain data. In *Proceedings of the DASFAA 2012, Part I*, pages 272–287. Springer.
40. Leung, C.K.-S., & Tanbeer, S.K. 2013. PUF-tree: a compact tree structure for frequent pattern mining of uncertain data. In *Proceedings of the PAKDD 2013, Part I*, pages 13–25. Springer.
41. Leung, C.K.-S., Cuzzocrea, A., & Jiang, F. 2013. Discovering frequent patterns from uncertain data streams with time-fading and landmark models. *LNCS Transactions on Large-Scale Data- and Knowledge-Centered Systems (TLDKS) VIII*, pages 174–196. Springer.
42. Leung, C.K.-S., Mateo, M.A.F., & Brajczuk, D.A. 2008. A tree-based approach for frequent pattern mining from uncertain data. In *Proceedings of the PAKDD 2008*, 653–661. Springer.
43. Leung, C.K.-S., Hao, B., & Brajczuk, D.A. 2010. Mining uncertain data for frequent itemsets that satisfy aggregate constraints. In *Proceedings of the ACM SAC 2010*, pages 1034–1038.
44. Leung, C.K.-S., Tanbeer, S.K., Budhia, B.P., & Zacharias, L.C. 2012. Mining probabilistic datasets vertically. In *Proceedings of the IDEAS 2012*, pages 199–204. ACM.
45. Madden, S. 2012. From databases to big data. *IEEE Internet Computing*, 16(3), pages 4–6.
46. Nadungodage, C.H., Xia, Y., Lee, J.J., & Tu, Y. 2013. Hyper-structure mining of frequent patterns in uncertain data streams. In *Knowledge and Information Systems (KAIS)*, 37(1), pages 219–244. Springer.
47. Ren, J., Lee, S.D., Chen, X., Kao, B., Cheng, R., & Cheung, D. 2009. Naive Bayes classification of uncertain data. In *Proceedings of the IEEE ICDM 2009*, pages 944–949.
48. Suciu, D. 2009. Probabilistic databases. In *Encyclopedia of Database Systems*, pages 2150–2155. Springer.
49. Sun, L., Cheng, R., Cheung, D.W., & Cheng, J. 2010. Mining uncertain data with probabilistic guarantees. In *Proceedings of the ACM KDD 2010*, pages 273–282.
50. Tong, Y., Chen, L., Cheng, Y., & Yu, P.S. 2012. Mining frequent itemsets over uncertain databases. In *Proceedings of the VLDB Endowment (PVLDB)*, 5(11), pages 1650–1661.
51. Wang, L., Cheng, R., Lee, S.D., & Cheung, D.W. 2010. Accelerating probabilistic frequent itemset mining: a model-based approach. In *Proceedings of the ACM CIKM 2010*, pages 429–438.
52. Wasserkrug, S. 2009. Uncertainty in events. In *Encyclopedia of Database Systems*, pages 3221–3225. Springer.
53. Xu, L., & Hung, E. 2012. Improving classification accuracy on uncertain data by considering multiple subclasses. In *Proceedings of the Australasian AI 2012*, pages 743–754. Springer.
54. Zaki, M.J. 1999. Parallel and distributed association mining: a survey. *IEEE Concurrency*, 7(4), pages 14–25.
55. Zaki, M.J., Parthasarathy, S., Ogihara, M., & Li, W. 1997. New algorithms for fast discovery of association rules. In *Proceedings of the ACM KDD 1997*, pages 283–286.
56. Zhang, Q., Li, F., & Yi, K. 2008. Finding frequent items in probabilistic data. In *Proceedings of the ACM SIGMOD 2008*, pages 819–832.