Mining Interesting Patterns from Uncertain Databases

Exam Roll: Curzon– 1902

Registration No: H–1638

Session: 2007–2008

Masters Session: 2011–2012

A Thesis submitted in partial fulfillment of the requirements for the degree of Master of Science in Computer Science and Engineering

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY OF DHAKA

May 2014

Declaration of Authorship

I declare that this thesis titled, 'Mining Interesting Patterns from Uncertain Databases' and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Countersigned: _____

Supervisor Name
Designation

Signed: _____

Name
Candidate

Abstract

Frequent itemsets are one of the most rudimentary and paramount knowledge in data mining research domain that help to extract further interesting information from large-scale databases. There exists many efficient approaches for mining frequent itemsets from precise or certain data collections but the growing urges for mining frequent itemsets from numerous real life databases that contain uncertain or probabilistic data demanded a novel technique because certain and uncertain data is different both semantically and computationally. These uncertain databases (e.g. satellite monitoring, biometric data etc.) are the resources to many emerging applications in medical science, disaster management, national security etc. Due to a growing demand of efficient algorithms for mining frequent itemsets from uncertain databases, several approaches have been proposed in the recent years but all of them use support based constraint to prune the combinatorial search space and only support based constraint is not enough because the frequent itemsets may have weak affinity. Even a very high minimum support is not effective for finding correlated patterns with increased weight or support affinity. There are a few approaches in certain databases which proposed new measures to mine correlated patterns but they are not applicable in uncertain databases because certain and uncertain databases have significant semantic and computational differences. In this thesis , we propose a new strategy, Weighted Uncertain Interesting Pattern Mining (WUIPM) in which a tree structure, WUIP-tree and several new measures such as $uConf$ and $wUConf$ are suggested to mine correlated patterns from uncertain databases. To our knowledge, ours is the first work specifically to consider weight or importance of an individual item alongside correlation between items of patterns in uncertain databases. Besides, we propose a new metric, prefix proxy value, $pProxy$ for our WUIP-tree that helps to improve the mining performance. A comprehensive performance study shows that not only our strategy generates fewer but valuable patterns but also faster than existing approaches even when affinity measures are not applied.

Acknowledgements

First of all, I would like to express my gratefulness to Almighty Allah who offers me His blessings to complete this thesis work. I would like to thank my thesis supervisor SUPERVISORNAME. I am also deeply grateful to my former supervisor, SUPERVISOR NAME for his proper guidance in selecting the research area and supporting by all means in the process of the research work. Their proper guidance and efforts made this work possible.

I am highly grateful to my parents, family members, friends and senior fellows who always helped me and gave me moral support and inspiration. Last, but not the least, I would like to thank my department, Computer Science and Engineering, University of Dhaka, for giving me the opportunity for this research work and facilitate me throughout the whole Master of Science program.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

One of the most elementary research steps in Knowledge Discovery in Databases (KDD) is data mining. The sole purpose of data mining is to obtain interesting and functional knowledge from massive databases and utilize this information in various applications and additional research. The mining of useful information refers to the acquisition of previously unknown knowledge (e.g. frequent itemsets) from large data collections.

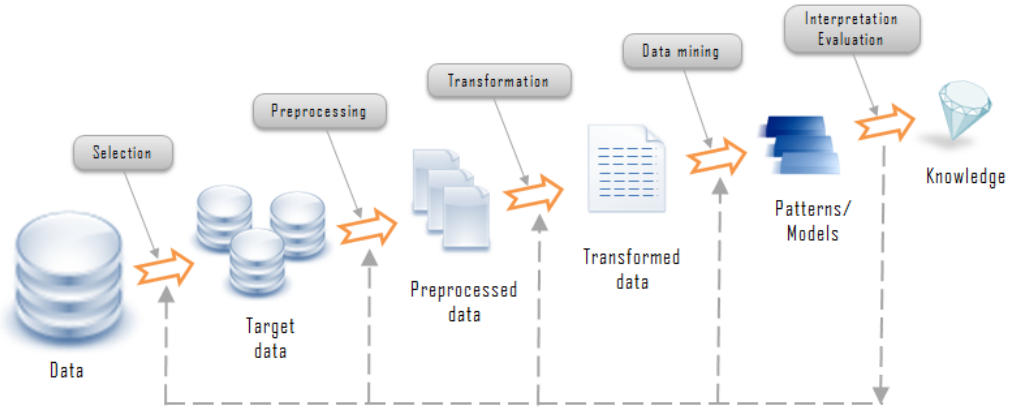


FIGURE 1.1: Knowledge Discovery Process

Data mining is generally utilized by companies (e.g., financial, retail, marketing and communication organizations) with a strong consumer focus. With the help of data mining they can find relationship among various internal factors such as product, price or staff skills and external factors such as indicators, economic, competition and customer demographics. Data mining assist them to determine the impact on sales, customer satisfaction and corporate profits. Therefore, it aids them the in depth summary as detail.

Frequent itemsets are such valuable information that are used in the process of discovering further interesting knowledge by association rule mining, classification and prediction, correlation mining etc. Hence, mining frequent itemsets is a principal step in data mining which is also known as frequent pattern mining.

Patterns (e.g. itemsets) that occur frequently in a database are called frequent patterns. Many approaches have been developed for mining frequent patterns in various fields as itemset (Apriori [1], FP-growth [2]), sequential itemset (GSP [3], Prefix-Span[4]), stream patterns (DSTree-Mining [5], RPS-tree [6]), high utility patterns (HUP-mining [7]), graph patterns (gSpan [8]), closed frequent patterns([9], [10], [11], [12], [13], [14]), association rules([15], [16], [17], [18], [19]), sequential patterns([20], [21], [22]), constraint based frequent patterns([23], [24], [25], [26], [27]) etc. A frequent itemset has to satisfy a threshold value, known as minimum support, to be considered as frequent. While mining frequent itemsets from certain and transactional database, this threshold value is merely the occurrence count of the itemset.

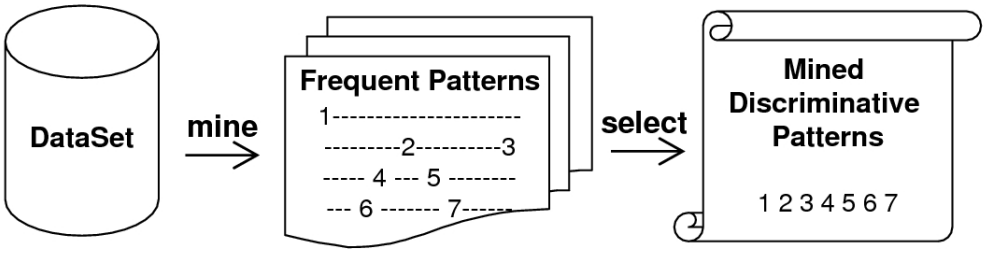


FIGURE 1.2: Frequent Pattern Mining

In a certain database, the presence or absence of an item is certainly known. For example in a market basket scenario an item, *milk* is either present or absent in a transaction of database, $DB = \{t_1, t_2, t_3\}$, where $t_1 = \{milk, bread, beer\}$, $t_2 = \{bread, beer\}$ and $t_3 = \{milk, bread\}$. Here, an item (e.g. *milk*) can never be partially present. In this example, if minimum support, $minSup = 60\%$ then an itemset, $I = \{milk, bread\}$ is frequent itemset having support, $sup = 66.67\%$.

However, real life databases are not always certain databases. Most real life databases contain data whose correctness is uncertain. In order to work with such data, there is a need to quantify the integrity of the data. Hence, uncertain or probabilistic databases were introduced that achieves the goal to store the uncertainty of data alongside concrete records. Here, a database could exist in multiple states called possible worlds with associated probability values.

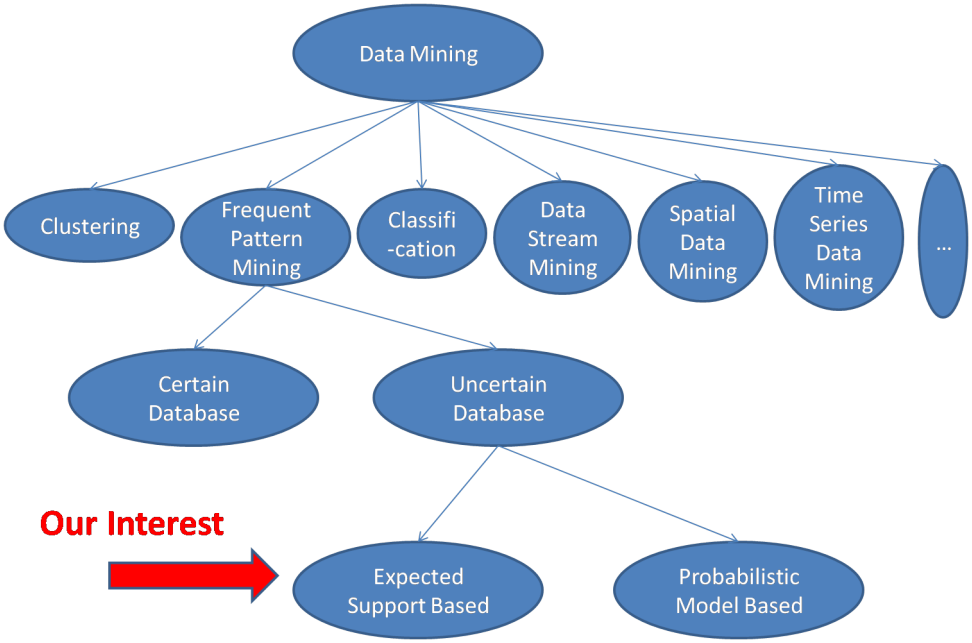


FIGURE 1.3: Data Mining Domains

For example, location detection using RFID sensors, patient diagnosis records, sensor readings, face recognition all have inherent uncertainty with the acquired data. A psychological symptom of a patient may look like { mood disorder: 78%, eating disorder 86 % } where the possible worlds are { mood disorder }, { eating disorder }, { mood disorder, eating disorder }. Every possible world has a probability of existence e.g., { mood disorder } has a existential probability of $0.78 * (1 - 0.86) = 0.1092$.

1.1 UDB(Uncertain Database)

	Mood Disorder	Anxiety Disorder	Eating Disorder	Obsessive-Compulsive Disorder	Depression	...	Self Destructive Disorder
Patient 1	97%	5%	84%	14%	76%	...	9%
Patient 2	90%	85%	100%	86%	65%	...	48%
...							

FIGURE 1.4: Psychological Symptoms Database Example

As discussed above, many real life scenarios exist wherein the presence or absence of an item, a_i cannot be certainly known. These situations are represented as uncertain databases where an item a_i is paired with a probability value, $p_i (0 < p_i < 1)$, which expresses its probability of existence. Hence, a transaction in uncertain database looks like $\{(a_1, p_1), (a_2, p_2), ..., (a_m, p_m)\}$.

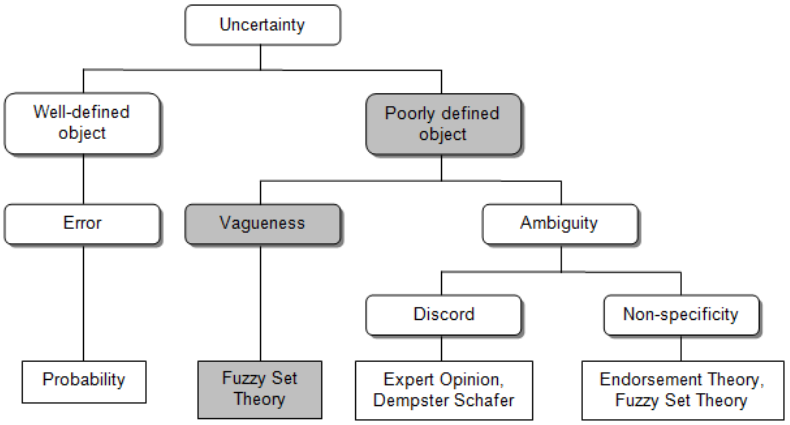


FIGURE 1.5: Types of Uncertainty

Appearently, it is hard to understand uncerainty in databases but they are everywhere. It is a state that can not be described entirely due to lack of knowledge. In most cases it can arise in the data when they are acquired, processed or visualized. Uncertainty can mean many things at a time e.g. poorly defined object, fuzzy sets, ambiguity, error, inherent probability etc.

1.1.1 Examples of Uncertain Data

Most of the real life databases have uncertainty in data. For example,

Information Extraction from Web pages: Extracting information or structure from web pages have inherent uncertainty because one can not be confident on the result.

As an example, a user behavior analysis system automatically extracts user information from various websites and hence find that some person, Mr. C works at Google. This result is not entirely reliable so a probabilty value is assoacited like there is 80% chance that Mr. A works at Google.

Human Readings : In many cases, information acquired from human experiences are not utterly reliable. Different people can describe a similar event differently.

As an example, in a bird viewing event Mr. C saw a crow flying by. But, Mr. D thinks it was a raven. So, uncertainty values can be attached with these observations like the bird has 75% probabilty to be crow and 20% probability to be a raven.

Sensors : Sensors are naturally error prone.

As an example, assume that sensore S reported that temperature of a certain place at a certain time is 40 degree with possible error between 5 degree.

Inherent Uncertainty : Features like weather forecast is inherently uncertain.

As an example, todays weather forecast may be rainy but it has only 65% confidence.

Data Integration or Entity Resolution : Mapping schemas of various tables in a database or de-duplicating entries creates uncertainty.

As an example, a name, Edgar Allan Poe in one table may be the same person in another table inseerted as Edgar. A. Poe but it can not be assured.

1.1.2 Uncertainty Models of UDB

There are three main models of uncertain data in databases. i.e.,

Attribute Uncertainty : In this model, every attribute of a tuple has its own probability distribution.

For example, if sensor readings are collected for temperature and wind speed then each would be presented with its own probability distribution.

Correlated Uncertainty : In this model, several attributes may be described with a joint probabilty distribution.

For example, if x and y coordinates of an object are taken then the probabilty of various values may depend on distance from the previous coordinates. Here, disance should be described by a joint probability distribution for these coordinates because distance depends on both.

Tuple Uncertainty : In this model, all attributes of a tuple have one joint probability distribution and the tuple itself has a probability value of existence.

For example, the cause of mood disodrer may be a tuple {(eating disorder: 0.5), (panic disorder: 0.4)} where the probailty of existence is also 90% for this tuple.

In uncertain databases, the expected support count of an itemset is used instead of the actual support count of the certain databases. The expected support count of an itemset I, in an uncertain database DB is calculated as,

$$ExpSup(I) = \sum_{i=1}^{|DB|} (\prod_{x \in I} p(x, t_i))$$

where $|DB|$ is the number of transactions in DB and $p(x, t_i)$ is the existential probability of an item x in a transaction t_i. An itemset is considered frequent if its expected support

satisfies the predefined threshold. From the above formula, it is clear that there is a significant computational difference between frequent itemsets of certain and uncertain databases.

Several algorithms(U-Apriori [28], UF-growth [29], UFP-growth [30], CUF-growth [31], CUF-growth* [31], PUF-growth [32] etc.) have been proposed in recent years to mine frequent itemsets from uncertain databases. CUF-growth* [31] has outperformed UFP-growth [30] by introducing limiting values and thus has a compact tree structure but originates many false positives.

Then, PUF-growth [32] has successfully achieved a better running time by introducing limiting values for prefix branches of the tree and reducing false positives but this can be further improved which is shown in our proposed algorithm. All these have focused only in mining frequent patterns from uncertain databases.

Earlier, WIP [33] and Hyperclique Miner [34] have shown interest in mining correlated and weighted correlated patterns but the basic difference between certain and uncertain databases makes it obsolete for uncertain databases. As a result, there is no suitable approach for mining affinity patterns from uncertain databases which might have been more useful to users.

1.2 Motivating example

Let us consider a medical diagnosis database where a sample record may look like {fever:70%, flu:60%, AIDS:50%, leukemia:40%} where the patient is reckoned to have fever with 70% probability and so on. Frequent pattern mining algorithms will extract the most frequent itemsets given a minimum support threshold as interesting patterns but diseases like AIDS and leukemia are expected to occur far less than fever and flu in a common diagnosis database.

Therefore, patterns with AIDS and leukemia will be regarded as non interesting unless minimum support is very low. But, these kind of diseases are more critical than others and hence, experts in medical science want to know more about the behaviors of them. This problem can be solved using a very low minimum support but having a low minimum support will introduce other problems like mining too many redundant or non valuable patterns which violates the primary goal of data mining. As a result, we can conclude that AIDS and leukemia must be given more importance/weight than fever and flu.

Now, let us assume that we want to find the correlated diseases e.g., fever and AIDS are correlated illness. In frequent pattern mining, it is very usual to extract {fever, AIDS, leukemia} as a frequent pattern which indicates that items are equally correlated but this is not the scenario because fever, AIDS or fever, leukemia are correlated but AIDS and leukaemia is not. Therefore, a new way to mine correlation between items is mandatory.

In this paper, we propose a new tree structure, WUIP-tree to capture uncertain data with a faster mining algorithm, WUIPM where weight or importance value for each item is considered and the given weight represents the importance of the item. In the above example AIDS and leukemia will be given more weight than fever and flu. A new measure weighted expected support confidence for uncertain databases, $wUConf$ is introduced to mine interesting patterns with importance. Another correlation measure expected support confidence for uncertain databases, $uConf$ is also proposed to mine correlated patterns from uncertain databases. Moreover, WUIPM is faster than others while mining frequent patterns and mines more useful patterns with correlation and according to importance.

1.3 Objectives

It has been discovered that existing approaches lack some key features regarding extraction of valuable information from uncertain databases. Briefly, existing works have the following short comings:

- While adjusting with the computational differences of uncertain data, existing algorithms had to compromise with either desired running time efficiency or compactness of data storage compared to the approaches for certain databases.
- All the approaches failed to realize the importance of affinity among items of frequent patterns and hence, did not consider affinity patterns. As a result, they mine too many patterns with weak affinity.
- They did not consider the weight of individual item too. Therefore, many interesting knowledge was undiscovered alongside frequent patterns.

To overcome the stated problems we have the following objectives:

- Devise a complete approach that is both fast in running time and have a compact data structure to store the uncertain database.

- Consider affinity among items and implement relevant measures and algorithms to mine correlated or strong affinity patterns.
- Attach weights to items according to importance of individual item and mine correlated frequent patterns that have similar importance.
- Let the user decide for mining all types of interesting patterns together using threshold inputs and only one solution.

1.4 Our Contributions

The key contributions of the paper are as follows:

- A new tree structure WUIP-tree for capturing uncertain database with a faster mining algorithm WUIPM to mine frequent patterns from WUIP-tree.
- A new metric prefix proxy value, $pProxy$ is introduced in WUIP-tree to prune infrequent itemsets efficiently.
- Contemplation and addition of weight and expected support correlation in uncertain databases.
- Introduction of weighted expected support confidence, $wUConf$ measure for uncertain databases.
- Introduction of expected support confidence, $uConf$ as a correlation measure for uncertain databases.
- Exploration of pruning techniques by the anti-monotone property of the $wUConf$ and $uConf$ measures.
- Incorporation of correlation measures into WUIP-tree with minimum computation cost.
- Improvement of WUIPM algorithm by using $uConf$ and $wUConf$ measures.
- Extensive performance analysis and experimental results study that show the superiority of WUIPM algorithm.

1.5 Thesis Organization

We have carried out the thesis work to consummate the objectives. An outline of rest of the chapters is provided as follows:

Chapter 2 describes the necessary background study and existing approaches.

Chapter 3 provides the details of our proposed approach with required analysis.

Chapter 4 presents the results of experimental results in detail with comparison and analysis.

Chapter 5 shows several real life applications for our proposed approach.

Chapter 6 ends the thesis with a brief conclusion and future scopes.

Chapter 2

Background Study and Related Works

In this chapter we have discussed some basic terminologies and background knowledge in uncertain data mining. This chapter also focuses on very recent and related work in frequent pattern mining of uncertain databases and other important terminologies related with association and correlation analysis in data mining and contains the most recent frequent pattern mining algorithms for uncertain databases CUF*-growth [31] and PUF-growth [32] alongside some earlier works on similar problem i.e., U-Apriori [28], UF-growth [29], UFP-growth [30], CUF-growth [31],. We have highlighted their working procedure, analyzed their complexity and provided example workout to better understanding the algorithms. We have also included the terminologies needed to define the approaches.

2.1 Data Mining

In Chapter 1 we have discussed what is data mining and frequent pattern mining. There are two common approaches in determining frequent itemset. First one is Apriori [1] algorithm, based on a prior knowledge, that is, if any pattern is not frequent then none of its super-pattern will be frequent. It works in a level wise approach. In each level it generates frequent sub-patterns from the given database and merges them to generate the candidates for next level. The second one is pattern growth e.g., FP-growth [2] method. Here no candidate generation is needed. A tree is constructed based on the patterns in the dataset and then frequent patterns are mined from the tree. The FP-growth [2] algorithm needs at most two scans of the database, while the number of database scans for the candidate generating algorithms (Apriori) increases with the dimension of

the candidate itemsets. Also, the performance of the FP-growth [2] algorithm is not influenced by the support factor, while the performance of the Apriori [2] algorithm decreases with the support factor. Thus, the candidate generating algorithms (derived from Apriori) behave well only for small databases with a large support factor (at least 30%). [35].

2.2 Uncertain Data Mining

In a large range of application domains, the analysis of meteorological trends, of medical behavior of living organisms, or of recorded physical activity is built on temporally dependent observations or multi observation data. The presence of a temporal ordering of the observations of a multi-observation object incorporates the key property of temporal variability and leads to the data model of time series. In particular in environmental, biological or medical applications, we are faced with time series data that features the occurrence of temporal patterns composed of regularly repeating sequences of events, where cyclic activities play a key role.

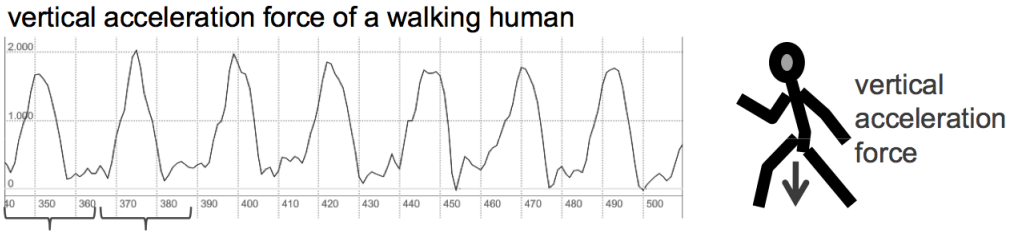


FIGURE 2.1: Evolution of periodic patterns in medical and biological applications

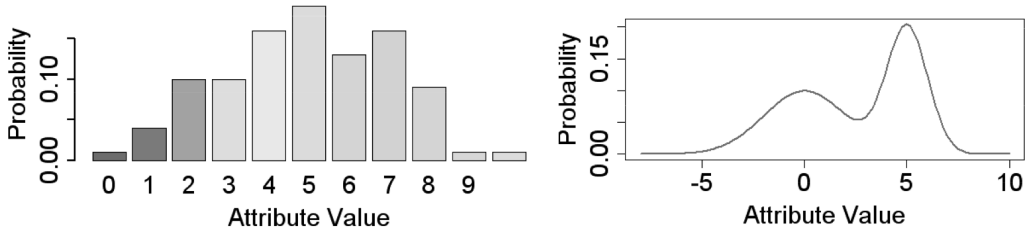
An example of a periodic time series is depicted in Figure 2.1, showing the motion activity of a human, in particular the vertical acceleration force that repetitively occurs during a human motion like walking or running. Though consecutive motion patterns show similar characteristics, they are not equal. It is possible to observe changes in the shape of consecutive periodic patterns that are of significant importance.

In the medical domain, physical activity becomes more and more important in the modern society. Nowadays, cardiovascular diseases cover a significant part of annually occurring affections, which is due to the reduced amount of activity in the daily life.

This leads to the research direction of uncertain or probabilistic data. Here, the basic question arises which of these observations is most likely to represent this object. Materializing this likelihood, the observations are associated with probability values; this creates existential dependencies among the observations, as the existence of an observation affects the existence of the other observations of the same object.

Coping with data uncertainty initiates a need for developing suitable data models and techniques for searching and mining. The models commonly applied for uncertain databases is presented below.

2.2.1 Modeling Uncertain Data



(a) Discrete attribute uncertainty. (b) Continuous attribute uncertainty.

FIGURE 2.2: Variants of attribute uncertainty

2.2.1.1 Categorization

Uncertainty in databases can generally be incorporated as tuple uncertainty and attribute uncertainty. Assuming tuple uncertainty, tuples are associated with a probability to appear in the database. This characteristic is also called existential uncertainty. The property of attribute uncertainty implies that a tuple has at least one uncertain attribute where the possible values are contained within a defined range. In the literature, probabilistic data models are classified in two types w.r.t. the attribute uncertainty: the discrete uncertainty model (Figure 2.2(a)) and the continuous uncertainty model (Figure 2.2(b)).

In many real-world applications, uncertain objects are already given by discrete observations, in particular if the objects are derived from sensor signals. This type of representation is motivated by the fact that, in many cases, only discrete but ambiguous object information as usually returned by common sensor devices is available, e.g., discrete

snapshots of continuously moving objects. Respectively, the ULDB model or x-relation model [36], introduced in the Trio system [37], will be presented in the following.

2.2.1.2 The X-Relational Model

The x-relation model extends the relational database model by incorporating uncertainty and lineage [36] and it supports existential uncertainty and attribute uncertainty. Relations in the *x-relation* model are called *x-relations* and contain uncertain tuples with alternative instances, which are called *x-tuples*.

2.2.1.3 The Possible World Semantics

(a) Tuples and x-tuples.

Tuples		
Tuple	Location	Prob.
t_1	Renzy's Den	50%
t_2	Waterhole	20%
t_3	Hunting Grounds	30%
t_4	Waterhole	10%
t_5	Hunting Grounds	10%
t_6	The Forest	20%

Tiger X-Relation	
Name	X-Tuple
Renzy	$\{t_1, t_2, t_3\}$
Unknown Tiger ?	$\{t_4, t_5, t_6\}$

(b) Possible worlds.

Possible Worlds		
World	Tuples	Prob.
W_1	$\{t_1\}$,{}	30%
W_2	$\{t_1\}$, $\{t_4\}$	5%
W_3	$\{t_1\}$, $\{t_5\}$	5%
W_4	$\{t_1\}$, $\{t_6\}$	10%
W_5	$\{t_2\}$,{}	12%
W_6	$\{t_2\}$, $\{t_4\}$	2%
W_7	$\{t_2\}$, $\{t_5\}$	2%
W_8	$\{t_2\}$, $\{t_6\}$	4%
W_9	$\{t_3\}$,{}	18%
W_{10}	$\{t_3\}$, $\{t_4\}$	3%
W_{11}	$\{t_3\}$, $\{t_5\}$	3%
W_{12}	$\{t_3\}$, $\{t_6\}$	6%

Figure 2.3: Tuples describing locations of tigers, an x-relation containing x-tuples with their possible locations and corresponding possible worlds with their probabilities.

In relational databases, a popular semantics to cope with the uncertainty of data has been introduced by adopting Saul Kripkes Possible Worlds Semantics [38], e.g., as performed in [39]. Incorporating this semantics into the *x-relation* model, an uncertain database is instantiated into a possible world.

Example: Figure 2.3 shows an *x-relation* that contains information about the possible positions of tigers in a wildlife sanctuary. Here, the first *x-tuple* describes the tiger named Renzy, who may be found at three possible (alternative) locations t1, t2 and t3. He may be in his cave with a probability of 50% or located at the water hole and at the hunting grounds with a probability of 20% and 30%, respectively. This *x-tuple* logically yields three mutually exclusive, possible tuple instances, one for each alternative location. Now, we know that an unknown tiger may have entered the wildlife sanctuary with a probability of 40%. In this case, it is not certain that the unknown tiger exists at all, which is an existential uncertainty of the *x-tuple*, denoted by a ? symbol. To incorporate this existential uncertainty, an additional, empty tuple is inserted added to the *x-tuple* of the unknown tiger, such that the probabilities of the possible worlds can be computed. Taking into account the four alternatives (including the alternative of no unknown tiger) for the position of the unknown tiger, there are twelve possible instances (worlds) of the tiger *x-relation*. In general, the possible worlds of an *x-relation* R correspond to all combinations of alternatives for the *x-tuples* in R. In this model, the probability of the unknown tiger being at the water hole is not affected by the current position of Renzy, due to the independence assumption among *x-tuples*. Considering the general case with possible tuple dependencies, this example could be extended by the natural restriction that male tigers are territorial and the position of a tiger may be affected by the presence of other tigers in its close vicinity. Thus, for example, world W6 might not occur by rule.

2.3 Frequent Pattern Mining In Uncertain Databases

There are various data mining applications that have to cope with the presence of uncertainty. Then, techniques designed for similarity query processing can apply in order to obtain effective and efficient solutions. For example, association rule analysis is one of the most important fields in data mining. It is commonly applied to market-basket databases for the analysis of consumer purchasing behavior. Such databases consist of a set of transactions, each containing the items a customer purchased. The most important and computationally intensive step in the mining process is the extraction of frequent itemsets, sets of items that occur in a minimum number of transactions. It is generally assumed that the items occurring in a transaction are known for certain.

However, also in transaction databases, this is not always the case due to several reasons:

- In many applications, the data is inherently noisy, such as data collected by sensors or in satellite images.

- In privacy protection applications, artificial noise can be added deliberately [40]. Finding patterns despite this noise is a challenging problem.
- By aggregating transactions by customer, it is possible to mine patterns across customers instead of transactions. This produces estimated purchase probabilities per item per customer rather than certain items per transaction.

In such applications, the information captured in transactions is uncertain, since the existence of an item is associated with a likelihood measure or existential probability. Given an uncertain transaction database, it is not obvious how to identify whether an item or itemset is frequent because it cannot be generally said for certain whether an itemset ap- pears in a transaction. In a traditional (certain) transaction database, the solution is to simply perform a database scan and count the transactions that include the itemset. This does not work in an uncertain transaction database.

Customer	Item	Prob.
A	Game	1.0
A	Music	0.2
B	Video	0.4
B	Music	0.7

ID	Transaction
t_A	(Game, 1.0); (Music, 0.2)
t_B	(Video, 0.4); (Music, 0.7)

FIGURE 2.4: Example application of an uncertain transaction database.

World	TransactionDB	Prob.
W_1	{Game}; {}	0.144
W_2	{Game, Music}; {}	0.036
W_3	{Game}; {Video}	0.096
W_4	{Game, Music}; {Video}	0.024
W_5	{Game}; {Music}	0.336
W_6	{Game, Music}; {Music}	0.084
W_7	{Game}; {Video, Music}	0.224
W_8	{Game, Music}; {Video, Music}	0.056

FIGURE 2.5: Corresponding possible worlds.

Dealing with such databases is a difficult, but interesting problem. While a na ve approach might transform uncertain items into certain ones by thresholding the probabilities, this loses useful information and leads to inaccuracies. Existing approaches in the literature are based on expected support, first introduced in [41]. Approaches described in [41] and [42] take the uncertainty of items into account by computing

the expected support of itemsets. There, itemsets are considered to be frequent if the *expected support* exceeds *minSup*. Our approach described in Chapter 3 is based on *expected support*

Example: Consider a department store. To maximize sales, customers can be analyzed to find sets of items that are all purchased by a large group of customers. This information can be used for advertising directed to this group. For example, by providing special offers that include all of these items along with new products, the store can encourage new purchases. Figure 2.4 shows such information. Here, Customer A purchases games every time he visits the store and music (CDs) 20% of the time. Customer B buys music in 70% of her visits and videos (DVDs) in 40% of them. The store uses a database that represents each customer as a single uncertain transaction, also shown in Figure 2.4.

2.3.1 Expected Support

Previous work addressing the problem of frequent itemset mining in uncertain databases was based on the expected support [41] , [42], [43], [31] and [32] which is defined as follows :

$$ExpSup(I) = \sum_{W_i \in W} \left(p(W_i) \times support_{W_i}(I) \right)$$

where W is the set of all possible worlds, $p(W_i)$ is the probability of a particular world, W_i and $support_{W_i}(I)$ is the support of I for the world W_i.

In general it can be assumed in uncertain databases that tuple and item are independent of each other from this assumption the *expected support* can be redefined to scan database once instead of enumerating all possible worlds i.e.,

$$ExpSup(I) = \sum_{i=1}^{|UDB|} \left(\prod_{x \in I} p(x, t_i) \right)$$

where UDB is the uncertain database, $p(x, t_i)$ is the probability of a particular item x in t_i

2.4 Existing Approaches

Many algorithms (U-Apriori [44], UF-growth [29], UFP-growth[30], UH-Mine [45], CUF-growth[31], CUF-growth*[31], PUF-growth [32] etc.) have been developed to mine frequent itemsets from uncertain databases. Among them CUF-growth*[31] was proposed in 2012 that introduced a compact tree and fast mining algorithm which outperformed others. Then, in 2013, PUF-growth [32] was proposed which used a different kind of tree structure that depends on prefix branches to mine frequent patterns. All these approaches only applicable for mining frequent patterns and did not address a valuable problem data mining i.e., correlation among the items of frequent itemsets. In this section, we will discuss the CUF-growth*[31], PUF-growth [32]. Before describing them a brief discussion of the existing approaches is given chronologically.

2.4.1 U-Apriori

To handle uncertain data, the U-Apriori algorithm was proposed, which is a modication of the Apriori algorithm. Specically, instead of incrementing the support counts of candidate patterns by their actual support, U-Apriori increments the support counts of candidate patterns by their *expected support*. U-Apriori suffers from the following problems:

- Inherited from the Apriori algorithm, U-Apriori does not scale well when handling large amounts of data because it also follows a level wise generate-and-test framework.
- If the existential probabilities of most items within a pattern I are small, increments for each transaction can be insignificantly small. Consequently, many candidates would not be recognized as infrequent until most (if not all) transactions were processed.

2.4.2 UF-growth

Observing that FP-growth [2] (a tree-based algorithm) usually outperforms Apriori [1] when mining precise data, UF-growth [29] was proposed for mining uncertain data. Note that key success of FP-growth [2] over Apriori [1] is mainly due to its FP-tree [2], which is a compact tree structure capturing frequent items within transactions in the databases of precise data. By extracting appropriate tree paths to construct subsequent FP-trees

[2], frequent itemsets can be mined. Each tree path represents a transaction. Each node in a tree path captures (i) an item x and (ii) its actual support (i.e., occurrence count of x in that tree path). Tree paths (from the root) are merged if they share the same items (i.e., the captured transactions share the same prefix items). Due to this path sharing, the FP-tree [2]is usually compact. However, when dealing with uncertain data, the situation is different (because each item is associated with an existential probability value). The expected support of any itemset X is the sum of products of existential probability of items within X. Hence, UF-growth [29] uses a UF-tree [29] to capture frequent items within transactions of uncertain data. Each node in an UF-tree captures (i) an item x, (ii) its existential probability value, and (iii) the occurrence count of x in that tree path. By doing so, UF-growth [29] finds all and only those frequent itemsets by computing the expected support of an itemset X (as the sum of products of the captured existential probability values). Tree paths are merged if they share the same items and existential probability values. Consequently, UF-trees [29] may not be as compact as FP-trees [2].

2.4.3 UFP-growth

To reduce the tree size, UFP-growth [30] groups similar nodes (i.e., nodes with the same item x but similar existential probability values) into a cluster. Each cluster of the item x captures (i) the maximum existential probability value of all nodes within the cluster and (ii) the number of existential probability values in each cluster. Depending on the clustering parameter, the resulting tree namely, UFP-tree [30] may be as large as the UF-tree [29] (i.e., no reduction in tree size). On the other hand, if the UFP-tree [30] is smaller than the UF-tree [29], then UFP-growth [30] may return approximate results (e.g., with false positives or infrequent itemsets).

2.4.4 UH-Mine

The UH-Mine algorithm [45] stores all frequent items in each DB transaction in a hyper-structure called UH-struct [45]. As UH-Mine [45] does not take advantage of prefix-sharing, the size of the resulting UH-struct [45] is always as large as that of the DB for the frequent items. However, UH-Mine [45] was reported [45],[46] to be faster than UFP-growth [30].

2.4.5 CUF-growth*

In this section, CUF-growth* [31] is described using the example database Table 3.1, with minimum support of 18% or 1.26. CUF-growth* [31] constructs a tree named CUF-tree* to store the uncertain database and generates frequent itemsets using CUF-growth*.

A CUF-tree* introduces two new terms namely *transaction cap* of a transaction $t_j = \{(a_1, p_1), ..., (a_n, p_n)\}$, denoted as $P^{Cap}(t_j)$ and an extra limiting value, *bronze*.
$P^{Cap}(t_j)$ is defined as,

$$P^{Cap}(t_j) = \begin{cases} m_1 \times m_2 & \text{if } l > 1, \\ p_1 & \text{if } l = 1 \end{cases}$$

where $l = |t_j|$ represent the length of transaction t_j, $m_1 = max_{i \in [1,l]} p_i$ and $m_2 = max_{k \in [1,l], k \neq i} p_k$. A *bronze* value is the third highest p_i of t_j.

A CUF-tree* node stores these two values to calculate expected support cap of a k-itemset I, $ExpSup^{Cap}(I)$ which is defined as,

$$ExpSup^{Cap}(I) = \begin{cases} \sum_{i=1, I \subseteq t_i}^{|DB|} P^{Cap}(t_i) & \text{if } l < 3, \\ \sum_{i=1, I \subseteq t_i}^{|DB|} P^{Cap}(t_i) \times \prod_{j=3}^{k} bronze & \text{if } l \geq 3 \end{cases}$$

In Figure 2.4.5 a CUF-tree* is constructed for sample database, Table 3.1. CUF-tree* construction process is shown in Figure 2.4.5. At first an *Item-List*={d, a, c, b, e} of frequent items is generated and ordered in decreasing expected support. Then, transactions t_1, ..., t_7 are inserted in CUF-tree* one by one.

Initially transaction t_1={$d : 0.6$, $a : 0.4$, $c : 0.9$, $b : 0.5$, $e : 0.3$} is ordered according to the order of *Item-List* and infrequent items are removed. $P^{Cap}(t_1)$=0.54 and *bronze* = 0.5 are calculated. Items of t_1 are inserted one by one with values (0.54,0.5) in an empty CUF-tree*. The t_1 marked segment of Figure 2.4.5 shows the insertion of transaction t_1.

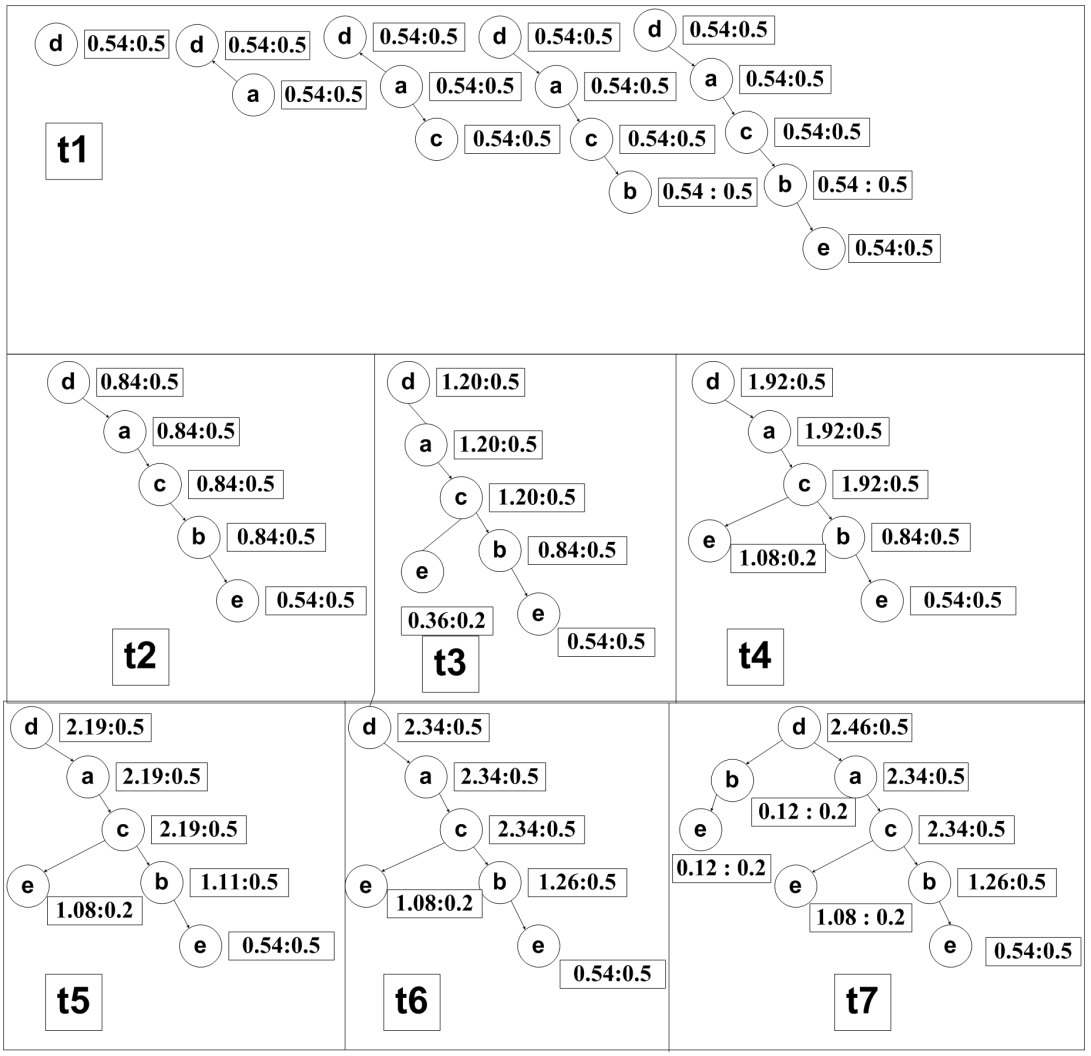


FIGURE 2.6: CUF-tree construction flow for Table 3.1 with minimum support 18%

Then, t_2={d : 0.5, a : 0.6, c : 0.4, b : 0.3} is arranged like t_1. Its $P^{Cap}(t_1)$=0.30 and *bronze*=0.4 are calculated. All the items of t_2 are shared with the existing tree branch. Hence, $P^{Cap}(t_1)$ value is added with the existing ones and *bronze* values are updated with the maximum of current and previous value. This is shown at segment labeled t_2 in Figure 2.4.5.

Transaction t_3 has three shared items i.e., {d, a, c}. Therefore, these nodes are updated like before but a new node is created for {e} as depicted in Figure 2.4.5.

Accordingly, in Figure 2.4.5, other transactions are inserted and the CUF-tree* is constructed. Any item that has expected support cap, $ExpSup^{Cap}$ less than minimum support is removed from the tree. In Figure 2.4.5 there is no such node.

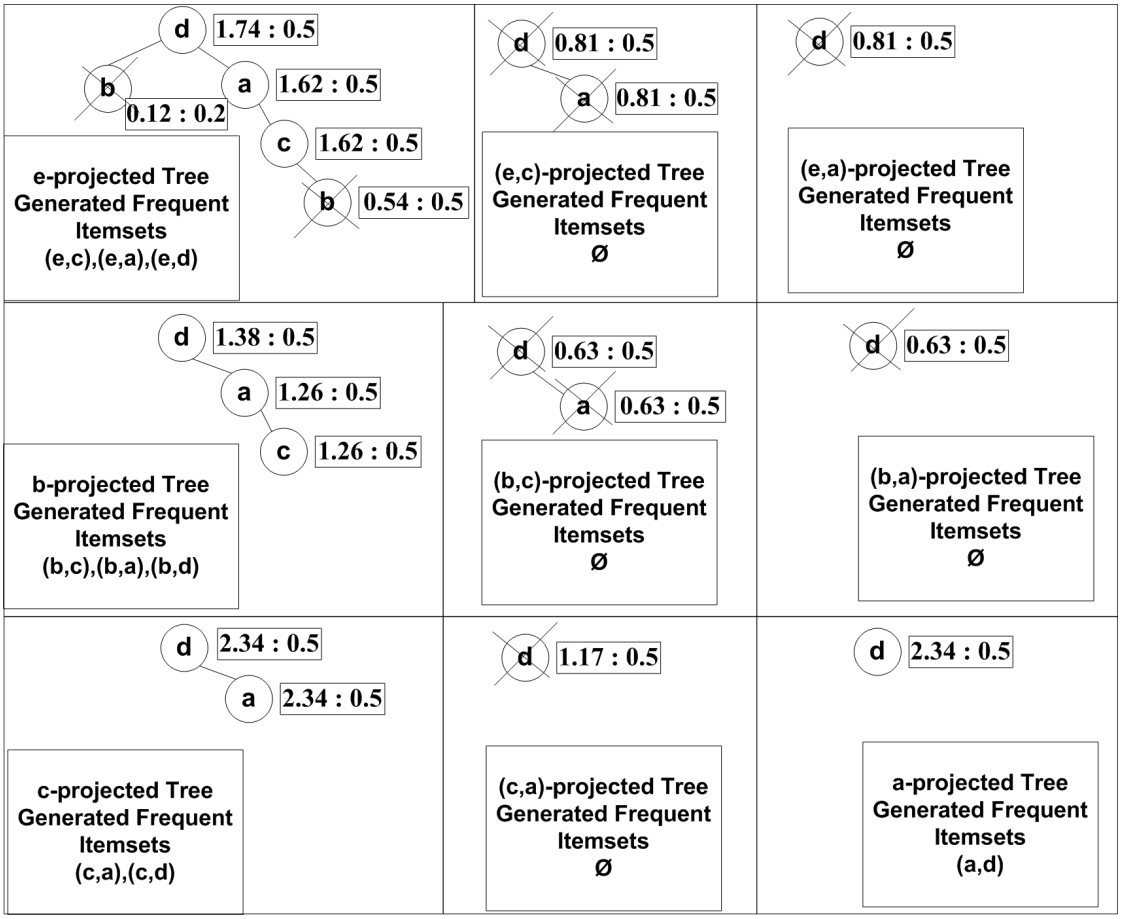


FIGURE 2.7: CUF-growth mining algorithm flow for Table 3.1 with minimum support 18%

Figure 2.4.5 shows the steps of CUF-growth* that mines frequent itemsets from CUF-tree* of Figure 2.4.5. CUF-growth* generates frequent itemsets by recursive tree projection, pruning unnecessary nodes and mining candidate frequent itemsets. After generating all the frequent itemsets, it scans database again and removes any infrequent itemsets called false positives.

For example, in Figure 2.4.5, CUF-tree* is projected for itemset $\{e\}$. Here, item b is an infrequent item because expected support cap for item b is 0.54+0.12=0.66 $< minSup$, 1.26. Hence, item b is pruned and itemsets $\{e, c\}$, $\{e, a\}$, $\{e, d\}$ are mined.

Then, the e-projected tree is further projected for $\{e, c\}$ as shown in Figure 2.4.5. Now node d has value (0.81,0.5) because the value of node d in previous e-projected tree had (1.62,0.5), which gives $1.62 \times 0.5 = 0.81$ according to definition of expected support cap. Similarly, node a has value (0.81,0.5). Nodes d, a are pruned and no itemsets can

be mined further. Hence, mining stops for $\{e\}$ and goes back to previous condition.

Similarly, frequent itemsets are mined for items d, a, c, b as shown in Figure 2.4.5 and thus, a list of candidate frequent itemsets, $\{d, a, c, b, e, (e,c), (e,a), (e,d), (b,c), (b,a), (b,d), (c,a), (c,d), (a,d)\}$ are generated. Among these itemsets, $\{d, a, c, b, e\}$ are actually frequent itemsets and other nine itemsets are false positives which are removed by determining the actual expected support from database.

2.4.6 PUF-growth

To reduce the size of the UF-tree [29] and UFP-tree [30], the prefix-capped uncertain frequent pattern tree (PUF-tree [32]) structure was proposed, in which important information about uncertain data is captured so that frequent patterns can be mined from the tree. The PUF-tree [32] is constructed by considering an upper bound of existential probability value for each item when generating a *k-itemset* (where $k > 1$). This upper bound of an item x_r in a transaction t_j is called the (prefixed) item cap of x_r in t_j. Thus PUF-tree [32] was compact.

2.5 Summary

For the computational difference of uncertain data, every approach has to choose between tree compactness and mining efficiency. Though, CUF-growth* [31] and PUF-growth [32] were able to balance between memory and processing time but they had introduced another problem i.e., false positive generation. Besides, they did not mine correlated patterns or did not address the scenario of weighted uncertain data stated in secion 1.2. In Chapter 3, we will describe a novel approach that mines frequent itemsets alongside itemsets with similar or dissimilar support and weight values for both certain and uncertain databases with minimal memory and processing time.

Chapter 3

Our Proposed Approaches

In this section, we have proposed the WUIP (Weighted Uncertain Interesting Pattern)-tree data structure and algorithm WUIPM to mine frequent itemsets using WUIP-tree. We have proposed a new metric in WUIP-tree named prefix proxy value, *pProxy* to prune infrequent itemsets effectively. We have also studied correlation between items in itemsets and importance of weight measure for finding interesting patterns.

As we have seen earlier that *support* has a new definition namely, *expectedSupport*, defined as

$$ExpSup(I) = \sum_{i=1}^{|UDB|} (\prod_{x \in I} p(x, t_i))$$

where I is itemset, $p(x, t_i)$ is existential probability value for any item x in transaction t_i and UDB is an uncertain database.

Transactions	**Items**				
t_1	$d : 0.6$	$a : 0.5$	$c : 0.4$	$b : 0.9$	$e : 0.3$
t_2	$d : 0.5$	$a : 0.6$	$c : 0.4$	$b : 0.3$	-
t_3	$d : 0.2$	$a : 0.1$	$c : 0.9$	-	$e : 0.4$
t_4	$d : 0.2$	$a : 0.9$	$c : 0.1$	-	$e : 0.8$
t_5	$d : 0.9$	$a : 0.1$	$c : 0.2$	$b : 0.3$	-
t_6	$d : 0.5$	$a : 0.1$	$c : 0.2$	$b : 0.3$	-
t_7	$d : 0.4$	-	-	$b : 0.2$	$e : 0.3$
Item	d	a	c	b	e
Weight	0.1	0.2	0.3	0.4	0.5

TABLE 3.1: Example Database

for uncertain databases because of computational and interpretation differences between certain and uncertain data. Besides, working with correlated patterns (both support and weighted support), we have come to some valuable findings regarding redefinition of correlation measures for uncertain databases. These findings can be described with the following observations.

3.1 Support Affinity or Confidence for Uncertain Databases

From Section 1.2, it is visible that correlated or affinity patterns has great value but to our knowledge we are the first to define support confidence for uncertain database. Earlier in Hyperclique Miner [47], support affinity patterns from ceratin databases were mined using $h\text{-}conf$ defined as,

$$h\text{-}conf(I) = \frac{support(\{i_1,i_2,...,i_m\})}{max_{1<k<m}(support(i_k))},$$
$$\text{where } I = \{i_1, i_2, ..., i_m\} \text{ is an itemset.}$$

Although, $h\text{-}conf$ is capable of finding strong affinity patterns from uncertain databases but it is not feasible for uncertain databases because of difference in *support* measure between ceratin and uncertain databases . Inspired from the versatility of $h\text{-}conf$ we have proposed $uConf$ [Definition 9] that is applicable to both certain and uncertain databases for finding affinity patterns. Alongside, $uConf$ follows anti-monotone property [Property 1] which helps to incorporate it easily into the pattern growth tree.

3.2 Weighted Support Affinity for Uncertain Databases

As we have seen in Section 1.2 that weight affinity patterns have particular value to users regarding patterns of similar weight or importance, but to define weight affinity measure for uncertain databases is a challenging task. Formally, weight of a pattern I is defined as,

$$Weight(I) = \frac{\sum i{=}1^{length(I)} Weight(x_i)}{length(I)}$$
$$\text{where } I \text{ is an itemset and } x_i \text{ is an item within } I.$$

Thus, weighted support for any itemset, I was defined as,

$$WeightedSupport(I) = Weight(I) \times Support(I)$$

To apply this concept in uncertain databases has some demerits. $WeightedSupport(I)$ violates the desired anti-monotone[15] property and for this reason it can not be used with a pattern growth tree. Earlier, WIP [33] proposed some measures to mine weight affinity patterns from certain databases that also follows anti-monotone property but it only works for certain databases. If $ExpectedSupprt(I)$ is used instead of $Support(I)$ in $WeightedSupport(I)$ then it still has the same problem of not following anti-monotone property. To mine weight affinity patterns from uncertain databases weight affinity required a new definition that successfully follows anti-monotone [15] property and mines weight affinity patterns. That is why we have proposed $wUConf$ [Definition 11] which achieves the desired goal.

To fulfil the described targets, we have proposed two measures on correlation i.e., $uConf$ for finding correlated patterns and $wUConf$ for correlated patterns with similar or dissimilar importance. The anti-monotone property of these measures are also discussed that helps to push them into our growth mining algorithm that saves a lot of extra computational effort and improves performance.

However, before describing our approach some terms and the problem (to be solved) need to be defined that are relevant to understand the proposed tree structure and algorithm. The proposed approach uses the concept of using an estimated boundary value for expected support instead of the actual expected support. This technique helps to maintain the compactness of the tree structure.

3.3 Preliminaries

Definition 1. "**Mining Frequent Patterns in Uncertain Databases**": Given $DB = \{t_1, ..., t_N\}$, an uncertain database with N transactions where minimum expected support threshold is δ. The problem is to mine frequent itemsets $FI \subseteq DB$, where $ExpSup(FI_i) \geq \delta$ and $FI_i \in FI$.

Definition 2. "**Mining Strong Affinity Patterns in Uncertain Databases**": Given $DB = \{t_1, ..., t_N\}$, an uncertain database with N transactions where minimum correlation or expected support confidence threshold is β. The problem is to mine correlated itemsets $CI \subseteq DB$, where $uConf(CI_i) \geq \beta$ and $CI_i \in CI$ 9.

Definition 3. "**Mining Weighted Interesting Patterns in Uncertain Databases**": Given $DB = \{t_1, ..., t_N\}$, an uncertain database with N transactions where individual

weight or importance is assigned for each item is attached and minimum weighted expected support confidence threshold is γ. The problem is to mine weighted correlated itemsets $WCI \subseteq DB$, where $wUConf(WCI_i) \geq \gamma$ and $WCI_i \in WCI$ 11.

Definition 4. "**Prefix**": The prefix of an item a_j for a transaction $t_i(a_j \in t_i)$, denoted as $P(t_i, a_j)$, is defined as the longest prefix subset of a_j in t_i. Let, t_i is the set $\{a_1 : p_1, a_2 : p_2, ..., a_{j-1} : p_{j-1}, a_j : p_j, ..., a_n : p_n\}$. Then,

$$P(t_i, a_j) = \{a_k : p_k \mid a_k : p_k \in t_i \text{ and } 1 \leq k < j\}$$

As an example, in Table 3.1, transaction $t_1 = \{d : 0.6, a : 0.5, c : 0.4, b : 0.9, e : 0.3\}$. Here, $P(t_1, b) = \{d : 0.6, a : 0.5, c : 0.4\}$

Definition 5. "**Transaction Prefix Cap**": The transaction prefix cap of a transaction t_i for $a_j(a_j \in t_i)$, an item in the transaction t_i, denoted as $T^{pCap}(t_i, a_j)$, is defined as the product of maximum existential probability value of items within $P(t_i, a_j)$ and p_j, after sorting t_i in a predefined order. Let, $t_i = \{a_1 : p_1, a_2 : p_2, ..., a_{j-1} : p_{j-1}, a_j : p_j, ..., a_n : p_n\}$, l is the length of $P(t_i, a_j)$ and maximum existential probability value within $P(t_i, a_j)$ is $p_m = max_{u \in [1,l])}(p_u)$ Then,

$$T^{pCap}(t_i, a_j) = \begin{cases} p_m \times p_j, & \text{if } l > 0 \\ \infty & otherwise \end{cases}$$

As an example, in Table 3.1, transaction $t_1 = \{d : 0.6, a : 0.5, c : 0.4, b : 0.9, e : 0.3\}$ sorted in the order of decreasing expected support. Here, $P(t_1, b) = \{d : 0.6, a : 0.5, c : 0.4\}$, hence, $T^{pCap}(t_1, b) = 0.6 \times 0.9 = 0.54$

Theorem 1. The existential probability of any *k-itemset*$(k > 1), I = \{a_1 : p_1, ..., a_k : p_k\}$ of a transaction t_i will always be less than or equal to the transaction prefix cap, $T^{pCap}(t_i, a_k)$, i.e., $p(I, t_i) \leq T^{pCap}(t_i, a_k)$.

Proof: Let, $I = \{a_1 : p_1, ..., a_k : p_k\}$ be an ordered itemset in t_i. Hence, prefix $P(t_i, a_k)$=$\{a_1 : p_1, ..., a_{k-1} : p_{k-1}\}$ and let, $l = |P(t_i, a_k)|$. Now, if $l > 0$ then existential probability, $p(I, a_k) = p_1 \times ...p_m... \times p_k$ where p_m is maximum among p_j $(1 \leq j < k$ and $0 < p_j < 1)$. Hence, $p(I, a_k) = p_m \times p_k \times \prod_{j=1, j \neq m}^{k-1} p_j \leq p_m \times p_k \implies p(I, a_k) \leq T^{pCap}(t_i, a_k)$ [Definition 5].

Definition 6. "**Transaction Prefix Proxy**": The transaction prefix proxy value of a transaction t_i for $a_j(a_j \in t_i)$, an item in the transaction t_i, denoted as $T^{pProxy}(t_i, a_j)$, is defined as the second maximum existential probability value of items within $P(t_i, a_j)$. Let, $t_i = \{a_1 : p_1, a_2 : p_2, ..., a_{j-1} : p_{j-1}, a_j : p_j, ..., a_n : p_n\}$ and $l = |P(t_i, a_j)|$ is the length of $P(t_i, a_j)$. Then,

$$T^{pProxy}(t_i, a_j) = \begin{cases} secondmax_{u\in[1,l])}(p_u), & \text{if } l > 1 \\ 0 & \text{otherwise} \end{cases}$$

As an example, in Table 3.1, transaction $t_1 = \{d : 0.6, a : 0.5, c : 0.4, b : 0.9, e : 0.3\}$, sorted in the order of decreasing expected support. Here, $P(t_1, b) = \{d : 0.6, a : 0.5, c : 0.4\}$, therefore, $T^{pProxy}(t_1, b) = 0.5$.

Theorem 2. The existential probability of any *k-itemset* $(k > 2)$, $I = \{a_1 : p_1, ..., a_k : p_k\}$ of a transaction t_i, will always be less than or equal to the estimated upper boundary i.e. the product of "*Transaction Prefix Cap*" and the $(k-2)^{th}$ order of "*Transaction Prefix Proxy*" for I in t_i, i.e., $p(I, t_i) \leq T^{pCap}(t_i, a_k) \times T^{pProxy}(t_i, a_k)^{k-2}$.

Proof: Let, $I = \{a_1 : p_1, ..., a_k : p_k\}$ be an ordered itemset in t_i. Hence, prefix $P(t_i, a_k) = \{a_1 : p_1, ..., a_{k-1} : p_{k-1}\}$ and let, $l = |P(t_i, a_k)|$. Now, if $l > 1$ then existential probability, $p(I, a_k) = p_1 \times ...p_m...p_{sm}... \times p_k$ where p_m is maximum and p_{sm} is second maximum among p_j $(1 \leq j < k$ and $0 < p_j < 1)$. Hence, $p(I, a_k) = p_m \times p_k \times \prod_{j=1, j\neq m}^{k-1} p_j \leq p_m \times p_k \times \prod_{j=1}^{k-2} p_{sm} \implies p(I, a_k) \leq p_m \times p_k \times p_{sm}^{(k-2)}$, because p_{sm} is maximum among p_j other than p_m. Hence, $p(I, a_k) \leq T^{pCap}(t_i, a_k) \times T^{pProxy}(t_i, a_k)^{k-2}$ [Definition 5, 6].

Definition 7. "**Prefix Proxy**": The prefix proxy value of an item a_j, denoted as $pProxy_k(a_j)$ is defined as the maximum of $T^{pProxy}(t_i, a_j)$ values among all t_i in $|DB|$ where $P(t_i, a_j)$ are same. Let, TP_k is a set of all transactions having same $P(t_i, a_j)$. Thus,

$$pProxy_k(a_j) = max_{t_i\in TP_k}(T^{pProxy}(t_i, a_j)).$$

As an example, in Table 3.1, $TP_1 = \{t_1, t_2, t_5, t_6\}$ and $TP_2 = \{t_7\}$. $T^{pProxy}(t_1, b) = 0.5, T^{pProxy}(t_2, b) = 0.5, T^{pProxy}(t_5, b) = 0.1, T^{pProxy}(t_6, b) = 0.1, T^{pProxy}(t_7, b) = 0$. Thus, $pProxy_1(b) = max_{t_i\in TP_1}(T^{pProxy}(t_i, b)) = 0.5$ and $pProxy_2(b)= max_{t_i\in TP_2}(T^{pProxy}(t_i, b)) = 0$

Definition 8. "**Expected Support Prefix Cap**": The prefix cap of expected support of an itemset, $I = \{a_1, a_2, ..., a_j\}$, where $a_p \prec a_q$ (a_p is a proper ancestor of a_q) for $1 \leq p < q \leq j$, denoted as $ExpSup^{pCap}(I)$ is defined as

$$ExpSup^{pCap}(I) = \begin{cases} \sum_{i=1}^{|DB|} T^{pCap}(t_i, a_j), & \text{if } |I| \leq 2 \\ \sum_{i=1}^{|DB|} (T^{pCap}(t_i, a_j) \times pProxy(a_j)^{|I|-2}), & \text{otherwise} \end{cases}$$

As an example, in Table 3.1, $|DB| = 7$ and so $ExpSup^{pCap}(\{c, b\}) = \sum_{i=1}^{7} T^{pCap}(t_i, b) = 0.54 + 0.15 + 0.27 + 0.15 + 0.08 = 1.19$.

Theorem 3. The actual expected support of any *itemset*, $I = \{a_1, ..., a_k\}$ is always less than or equal to the expected support prefix cap for I, $ExpSup^{pCap}(I)$. i.e., $ExpSup(I) \leq ExpSup^{pCap}(I)$.

Proof: Let, DB is an uncertain database with n transactions, $\{t_1, ..., t_n\}$ and $I = \{a_1, a_2, ..., a_k, ..., a_j\}$ be an itemset where $a_p \prec a_q$ (a_p is a proper ancestor of a_q) for $1 \leq p < q \leq j$. Now, if $1 < |I| < 3$, Expected support of I, $ExpSup(I) = \sum_{i=1}^{n} p(I, t_i)$ $\leq \sum_{i=1}^{n} T^{pCap}(t_i, a_j)$ [Theorem 1] $\implies$ $ExpSup(I) \leq ExpSup^{pCap}(I)$ [Definition 8]. If $|I| > 2$, Expected support of I, $ExpSup(I) = \sum_{i=1}^{n} p(I, t_i) \leq \sum_{i=1}^{n} T^{pCap}(t_i, a_j) \times T^{pProxy}(t_i, a_j)^{k-2}$ [Theorem 2]. From Definition 7, we know $pProxy_m(a_j)$ is maximum among all $T^{pProxy}(t_i, a_j)$. Hence, $ExpSup(I) \leq \sum_{i=1}^{n} T^{pCap}(t_i, a_j) \times pProxy_m(a_j)^{k-2}$. Hence, $ExpSup(I) \leq ExpSup^{pCap}(I)$ [Definition 8].

Definition 9. "**Uncertain Support Confidence**": The support confidence for patterns in uncertain databases, $uConf$ is defined as,

$$uConf(Z) = \frac{ExpSup(Z)}{\max_{z \in Z}(ExpSup(z))}$$

As an example, in Table 3.1, $|DB| = 7$ and so $uConf(\{c, b\}) = \frac{ExpSup(\{c,b\})}{\max_{z \in \{c,b\}}(ExpSup(z))} = \frac{0.36+0.12+0.06+0.06}{2.2} = 0.27$.

Property 1. (Anti monotone property of $uConf$: If a pattern Z has $uConf$ no less than minimum threshold then every subset of Z also has $uConf$ no less than that threshold. In other words, if the $uConf$ of pattern Z is less than the threshold then any super set of Z has $uConf$ less than the threshold. This property helps to prune weak affinity patterns rapidly.

Lemma 1. If Z_1 and Z_2 are two patterns such that $Z_1 \subseteq Z_2$, then $uConf(Z_1) \geq uConf(Z_2)$.

Proof: If $Z_1 \subseteq Z_2$, $ExpSup(Z_1) \geq ExpSup(Z_2)$ and $\max_{z \in Z_1}(ExpSup(z)) \leq \max_{z \in Z_2}(ExpSup(z))$. Then, $uConf(Z_1) \geq uConf(Z_2)$.

Definition 10. "**Pefix Cap of Uncertain Support Confidence**": The support confidence for patterns in uncertain databases, $uConf^{pCap}$ is defined as,

$$uConf^{pCap}(Z) = \frac{ExpSup^{pCap}(Z)}{\max_{z \in Z}(ExpSup(z))}$$

As an example, in Table 3.1, $|DB| = 7$ and so $uConf^{pCap}(\{c, b\}) = \frac{ExpSup^{pCap}(\{c,b\})}{\max_{z \in \{c,b\}}(ExpSup(z))}$ $= \frac{1.19}{2.2} = 0.54$.

Property 2. (Anti monotone property of $uConf^{pCap}$): If a pattern Z has $uConf^{pCap}$ no less than minimum threshold then every subset of Z also has $uConf^{pCap}$ no less than that threshold. In other words, if the $uConf^{pCap}$ of pattern Z is less than the threshold then any super set of Z has $uConf^{pCap}$ less than the threshold. This property helps to prune weak affinity patterns rapidly.

Lemma 2. If Z_1 and Z_2 are two patterns such that $Z_1 \subseteq Z_2$, then $uConf^{pCap}(Z_1) \geq uConf^{pCap}(Z_2)$.

Proof: If $Z_1 \subseteq Z_2$, $ExpSup^{pCap}(Z_1) \geq ExpSup^{pCap}(Z_2)$ and $\max_{z \in Z_1}(ExpSup(z)) \leq \max_{z \in Z_2}(ExpSup(z))$. Then, $uConf^{pCap}(Z_1) \geq uConf^{pCap}(Z_2)$.

Theorem 4. The actual support confidence of any $itemset, I = \{a_1, ..., a_k\}$ is always less than or equal to the support confidence prefix cap for I, $uConf^{pCap}(I)$. i.e., $uConf(I) \leq uConf^{pCap}(I)$.

Proof: Let, DB is an uncertain database with n transactions, $\{t_1, ..., t_n\}$ and $I = \{a_1, a_2, ..., a_k, ..., a_j\}$ be an itemset where $a_p \prec a_q$ (a_p is a proper ancestor of a_q) for $1 \leq p < q \leq j$. Now, $ExpSup(I) \leq ExpSup^{pCap}(I)$ [Theorem 3]. Hence, $uConf(I) \leq uConf^{pCap}(I)$.

Definition 11. "**Weighted Uncertain Support Confidence**": The weighted support confidence for patterns in uncertain databases, $wUConf$ is defined as,

$$wUConf(Z) = \frac{\min_{x \in Z}(weight(x)) \times ExpSup(Z)}{\max_{y \in Z}(weight(y) \times ExpSup(y))}$$

As an example, in Table 3.1, $|DB| = 7$ and so $wUConf(\{c, b\}) = \frac{0.4 \times 0.6}{0.8} = 0.3$.

Property 3. (Anti monotone property of $wUConf$): If a pattern Z has $wUConf$ no less than minimum threshold then every subset of pattern Z also has $wUConf$ no less than the threshold. In other words, if the $wUConf$ of pattern Z is less than threshold then any super set of Z has $wUConf$ less than threshold. This property helps to prune weak weighted affinity patterns rapidly.

Lemma 3. If Z_1 and Z_2 are two patterns such that $Z_1 \subseteq Z_2$, $wUConf(Z_1) \geq wUConf(Z_2)$.

Proof: If $Z_1 \subseteq Z_2$, $\min_{x \in Z_1}(weight(x)) \leq \min_{x \in Z_2}(weight(x))$, $ExpSup(Z_1) \geq ExpSup(Z_2)$ and $\max_{y \in Z_1}(weight(y) \times ExpSup(y)) \leq \max_{y \in Z}(weight(y) \times ExpSup(y))$, then $wUConf(Z_1) \geq wUConf(Z_2)$.

Definition 12. "**Pefix Cap of Weighted Uncertain Support Confidence**": The weighted support confidence for patterns in uncertain databases, $wUConf^{pCap}$ is defined as,

$$wUConf^{pCap}(Z) = \frac{\min_{x \in Z}(weight(x)) \times ExpSup^{pCap}(Z)}{\max_{y \in Z}(weight(y) \times ExpSup(y))}$$

As an example, in Table 3.1, $|DB| = 7$ and so $wUConf(\{c, b\}) = \frac{0.4 \times 1.19}{0.8} = 0.595$.

Property 4. (Anti monotone property of $wUConf^{pCap}$): If a pattern Z has $wUConf^{pCap}$ no less than minimum threshold then every subset of pattern Z also has $wUConf^{pCap}$ no less than the threshold. In other words, if the $wUConf^{pCap}$ of pattern Z is less than threshold then any super set of Z has $wUConf^{pCap}$ less than threshold. This property helps to prune weak weighted affinity patterns rapidly.

Lemma 4. If Z_1 and Z_2 are two patterns such that $Z_1 \subseteq Z_2$, $wUConf^{pCap}(Z_1) \geq wUConf^{pCap}(Z_2)$.

Proof: If $Z_1 \subseteq Z_2$, $\min_{x \in Z_1}(weight(x)) \leq \min_{x \in Z_2}(weight(x))$, $ExpSup^{pCap}(Z_1) \geq ExpSup^{pCap}(Z_2)$ and $\max_{y \in Z_1}(weight(y) \times ExpSup(y)) \leq \max_{y \in Z}(weight(y) \times ExpSup(y))$, then $wUConf^{pCap}(Z_1) \geq wUConf^{pCap}(Z_2)$.

Property 5. (Anti monotone property of $wUConf$ and $wUConf^{pCap}$): If a pattern Z has $wUConf$ or $wUConf^{pCap}$ no less than minimum threshold then every subset of pattern Z also has $wUConf$ or $wUConf^{pCap}$ no less than the threshold. In other words, if the $wUConf$ or $wUConf^{pCap}$ of pattern Z is less than threshold then any super set of Z has $wUConf$ or $wUConf^{pCap}$ less than threshold. This property helps to prune weak affinity patterns rapidly.

Theorem 5. The actual weight affinity value of any *itemset*, $I = \{a_1, ..., a_k\}$ is always less than or equal to the weight affinity prefix cap for I, $wUConf^{pCap}(I)$. i.e., $wUConf(I) \leq wUConf^{pCap}(I)$.

Proof: Let, DB is an uncertain database with n transactions, $\{t_1, ..., t_n\}$ and $I = \{a_1, a_2, ..., a_k, ..., a_j\}$ be an itemset where $a_p \prec a_q$ (a_p is a proper ancestor of a_q) for $1 \leq p < q \leq j$. Now, $ExpSup(I) \leq ExpSup^{pCap}(I)$ [Theorem 3]. Hence, $wUConf(I) \leq wUConf^{pCap}(I)$.

3.4 Mining Frequent Patterns from Uncertain Databases

The proposed approach for frequent pattern mining is divided into two parts, (1) WUIP-tree construction and (2) mining frequent itemsets using WUIPM from WUIP-tree. The steps are shown in Figure 3.3 and Figure 3.4. In the sections below, we have used Table 3.1 as a running example database DB to illustrate the parts (1) and (2) of our proposed approach.

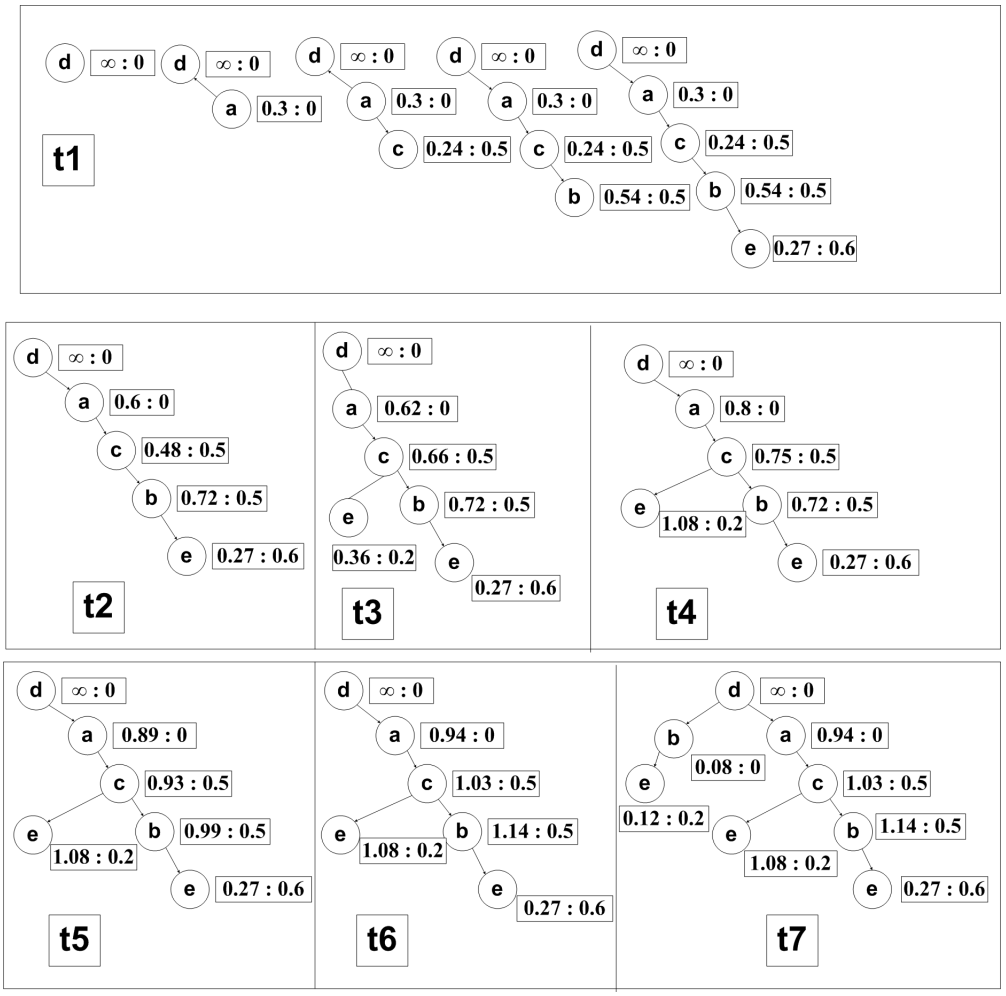


FIGURE 3.1: WUIP-tree construction flow for Table 3.1 with minimum expected support 18%

3.4.1 WUIP-tree construction

WUIP-tree construction is done in two database scans. Frequent items are selected from first scan and WUIP-tree is built during second scan using those frequent items. In general, a transaction is taken at a time and frequent items of that transaction are inserted as a tree branch. While insertion, prefix cap and prefix proxy values are also calculated gradually and stored in the corresponding nodes. Figure 3.3 depicts the steps to construct WUIP-tree.

Consider Table 3.1 as a sample database DB with 7 transactions $\{t_1, t_2, ..., t_7\}$ and minimum support 18%. Expected supports $(a : 2.3, b : 2.0, c : 2.2, d : 3.3, e : 1.8)$ of all the items are calculated in first database scan, then the infrequent items are removed and list is sorted in decreasing expected support order. The result is a list called $Item\_list = \{d : 3.3, a : 2.3, c : 2.2, b : 2.0, e : 1.8\}$.

In the second database scan, transactions are inserted into the tree as a branch and required information for each item is calculated one by one. For example, the first transaction $t_1 = \{a : 0.5, b : 0.9, c : 0.4, d : 0.6, e : 0.3\}$ is read and all the infrequent items within t_1 are removed. Then, they are sorted in the order of positions in $Item\_list$. Thus, t_1 becomes $\{d : 0.6, a : 0.5, c : 0.4, b : 0.9, e : 0.3\}$. Now, items are inserted into the tree in this order and prefix caps with prefix proxy values are also calculated.

Accordingly, $d : 0.6$ is inserted into an empty tree whose prefix is $P(t_1, d) = \{\phi\}$. So, $T^{pCap}(t_1, d) = \infty$ and $pProxy_1(d) = 0$. Next, $a : 0.5$ is inserted as a child of $d(\infty : 0)$ whose prefix, $P(t_1, a) = \{d : 0.6\}$ and so $T^{pCap}(t_1, a) = 0.6 \times 0.5 = 0.3$ and $pProxy_1(a) = 0$. Then $c : 0.4$ is inserted as a child of $a(0.30 : 0)$ whose prefix, $P(t_1, a) = \{d : 0.6, a : 0.5\}$. Maximum value from $P(t_1, a), p_m$ is 0.6 and so $T^{pCap}(t_1, c) = 0.6 \times 0.4 = 0.24$ and $pProxy_1(c) = 0.5$. Thus $b : 0.9, e : 0.3$ are inserted one by one and t_1 is captured in the tree. All these steps are depicted in the t_1 labeled area of Figure 3.1.

Like the first transaction all other transactions are captured by the WUIP-tree in a similar way except that the common prefixes are shared and transaction prefix caps are added with the existing node values. The $pProxy_k$ value is replaced with the maximum between existing and the current value of T^{pProxy}.

For example, transaction $t_2 = \{d : 0.5, a : 0.6, c : 0.4, b : 0.3\}$ (after sorting in the $Item\_list$ order and removing infrequent items) shares the branch $\{d : \infty, a : 0.30, c : 0.24, b : 0.54\}$ from root. Therefore, no new node is created and transaction prefix caps $(d : \infty, a : 0.30, c : 0.24, b : 0.18)$ are added with the existing node values $(\{d : \infty, a : 0.30, c : 0.24, b : 0.54\})$ for transaction prefix caps. Calculated T^{pProxy} values $(d : 0, a : 0, c : 0.5, b : 0.5)$ are updated with the existing $pProxy_k$ values $(\{d : 0, a : 0, c : 0.5, b : 0.5\})$. Thus, after inserting t_2 , WUIP-tree has one branch that is $\{d : \infty : 0, a : 0.3 + 0.3 : 0, c : 0.24 + 0.24 : 0.5, b : 0.54 + 0.18 : 0.5, e : 0.27 : 0.6\}$ or $\{d(\infty : 0), a(0.6 : 0), c(0.48 : 0.5), b(0.72 : 0.5), e(0.27 : 0.6)\}$. Figure 3.1 shows the above stated steps graphically. Likewise, each transaction of DB is processed and captured in the WUIP-tree.

3.4.2 WUIPM Algorithm

At the beginning $Item\_list$ is copied to the list of approximated set of frequent itemsets, $result$. Hence, $result = \{d : 3.3, a : 2.3, c : 2.2, b : 2.0, e : 1.8\}$ is the initial list of frequent itemsets.

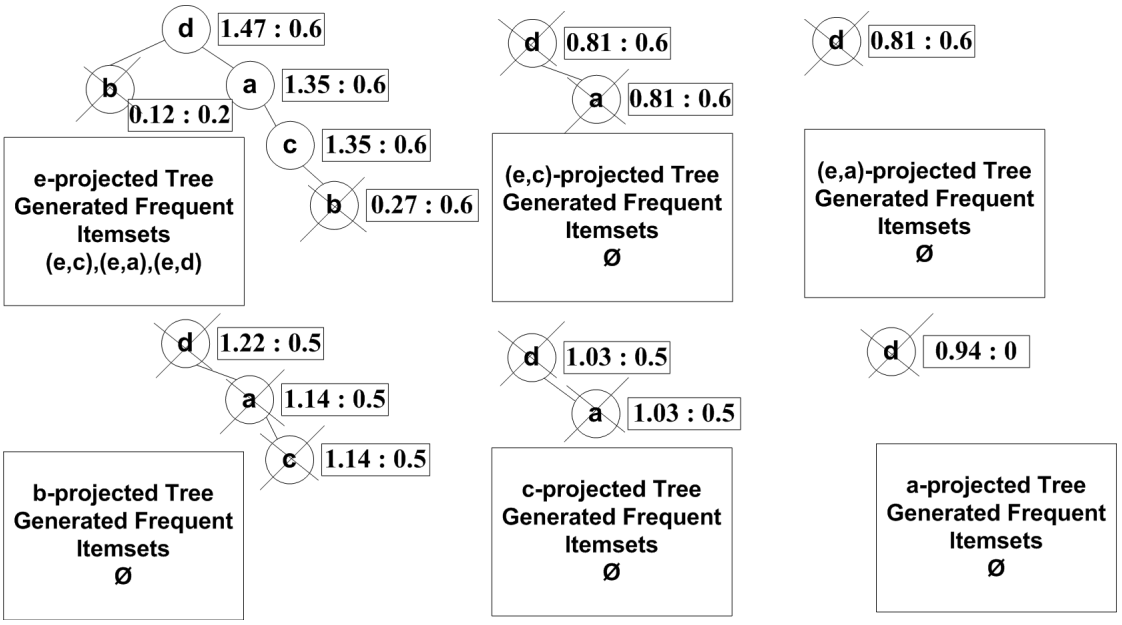


FIGURE 3.2: WUIPM mining algorithm flow for Table 3.1 with minimum support 18%

WUIPM has three main parts: (1) tree projection for an itemset, (2) filtering of projected tree by pruning and (3) mining. WUIPM does these three parts recursively for each item from $Item\_list = \{d : 3.3, a : 2.3, c : 2.2, b : 2.0, e : 1.8\}$, to generate all the frequent itemsets with some false positives. These three parts are more elaborately presented in pseudo code format in Figure 3.4. For simplicity, the algorithm of Figure 3.4 is discussed below using our sample database of Table 3.1.

For example, consider the projected trees in Figure 3.2, projected from the final WUIP-tree of Figure 3.1. As shown in Figure 3.2, item e is taken from $Item\_list$ and (1) the projected tree for the last item of currently selected itemset is created. Note that currently selected itemset is actually $\{e\}$ with last item e. This tree will be called *e-projected* tree. Tree branches from root to the parent of e node are collected and each node is assigned the node value of the e node appearing in the corresponding branch.

Hence, *e-projected* tree has three separate projected paths $\{d(0.27 : 0.6), a(0.27 : 0.6), c(0.27 : 06), b(0.27 : 0.6)\}$, $\{d(1.08 : 0.2), a(1.08 : 0.2), c(1.08 : 0.2)\}$, and $\{d(0.12 : 0.2), b(0.12 : 0.2)\}$. As shown in Figure 3.2, these are merged by sharing common prefixes and summing up the node values for T^{pCap}s and selecting maximum $pProxy_k$ value for shared nodes. Finally, the tree has two branches $\{d(1.35 : 0.6), a(1.35 : 0.6), c(1.35 : 0.6), b(0.27 : 0.6)\}$ and $\{d(0.12 : 0.2), b(0.12 : 0.2)\}$.

Then, (2) filtering of *e-projected* tree is done by pruning the infrequent items from the tree. $ExpSup^{pCap}(\{a_j\})$ is calculated for every item a_j in $Item\_list_{a_j}$ of *e-projected* tree. Here, the list is $\{d : 1.47, a : 1.35, c : 1.35, b : 0.39\}$. Nodes representing b are

Input: uncertain database DB, minimum support δ
Output: A WUIP-tree capturing items from DB with minimum support δ

```
1  begin
2      Calculate expected support for each item in DB;
3      Remove all infrequent items, arrange frequent items in a pre-defined order, let
       Item_list be the resultant item list;
4      Create the root node R;
5      foreach transaction T_i in DB do
6          Delete all infrequent items from T_i;
7          if T_i ≠ NULL then
8              Sort T_i according to the order in Item_list;
9              Insert transaction T_i ;
10         end
11     end
12 end
```

FIGURE 3.3: Capturing of uncertain data by WUIP-tree

removed because it has $ExpSup^{pCap}(\{b\}) = 0.39 < minSup, 1.26$ and *e-projected* tree becomes $\{d(1.47 : 0.6), a(1.35 : 0.6), c(1.35 : 0.6)\}$.

After pruning, (3) mining is done by adding each item in the *e-projected* tree with current itemset $\{e\}$. Thus, all the two length frequent itemset containing $\{e\}$ is generated. This set is called $temp\_result = \{\{e, d\}, \{e, a\}, \{e, c\}\}$ and $temp\_result$ is added with $result$ ($result = result \cup temp\_result$) to get the new frequent itemsets. Finally, $result = \{d, a, c, b, e, \{e, d\}, \{e, a\}, \{e, c\}\}$.

Then, each itemset from current $temp\_result$ is taken to do the above stated three steps recursively. For example $\{e, c\}$ is taken and *ec-projected* tree is constructed from *e-projected* tree. c is the last item of $\{e, c\}$ and thus *ec-projected* tree becomes $\{d(1.35 : 0.6), a(1.35 : 0.6)\}$ after (1) tree projection and (2) pruning. Thus current $temp\_result = \emptyset$ gives updated $result = \{d, a, c, b, e, \{e, d\}, \{e, a\}, \{e, c\}\}$.

The above process is done recursively to generate all the possible frequent itemsets. For the sample Table 3.1 and minimum support of 18% the final frequent itemsets are $result = \{d, a, c, b, e, \{e, d\}, \{e, a\}, \{e, c\}\}$. Among itemsets in $result$, $\{e, d\}, \{e, a\}, \{e, c\}$ are false positives. Figure 3.2 shows these mining steps and mined itemsets for individual candidates.

Input: WUIP-tree and minimum support δ, minimum support confidence $minUConf$ and minimum weighted support confidence $minwUConf$
Output: Interesting itemsets from DB

```
1  begin
2      foreach item α in Item_list do
3          P_α = ProjectTree(WUIP-tree,α);
4          Mine(P_α,α,Item_list_α,{α});
5          Remove all the N_α nodes and merge divided subtrees;
6      end
7  end
8  Procedure ProjectTree(WUIP-tree,α)
9  begin
10     foreach node N_α in WUIP-Tree do
11         Project path path_α from the parent of N_α up to the root;
12         foreach node PN_α in path_α do
13             Calculate PN_α.T^pCap and PN_α.pProxy in projected tree P_α;
14         end
15     end
16     return P_α;
17 end
18 Procedure Mine(P_α,α,Item_list_α,candidate)
19 begin
20     Put candidate as a candidate frequent itemsets;
21     foreach β in Item_list_α do
22         if ExpSup^pCap(β) < δ or uConf^pCap(β) < minUConf or
           wUConf^pCap(β) < minwUConf then
23             Remove all the N_β nodes and merge divided subtrees;
24             Delete β from Item_list_α;
25         end
26     end
27     if P_α ≠ NULL then
28         foreach item β in Item_list_α do
29             nextCandidate = candidate ∪ β;
30             P_β=ProjectTree(P_α,β);
31             Mine(P_β,β,Item_list_β,nextCandidate);
32         end
33     end
34 end
```

FIGURE 3.4: Mining of interesting itemsets by WUIPM

3.5 Mining Support Affinity Patterns from Uncertain Databases

To mine correlation among items of an itemset I, $uConf^{pCap}(I)$ serves the desired purpose because from observation it can be seen that any association rule from I has a rule $confidence$ greater than or equal to $uConf^{pCap}(I)$. $uConf$ is inspired from the famous measure called $all-confidence$ [48] for certain databases.

3.5.1 Pruning of Weak Affinity Patterns

Given, a threshold $minUConf$, WUIPM can prune weak affinity patterns at any moment while mining process is running. For any itemset I, $uConf(I)$ depends on expected support $ExpSup(I)$ and $ExpSup(x_i)$ where x_i is an item of I. But, using WUIP-tree we can find $ExpSup^{pCap}(I)$ instead of exact $ExpSup(I)$ of any itemset I alongside exact expected support of all the items (expected support of items are calculated at the beginning of WUIPM and stored for future use) i.e., $uConf^{pCap}$ can be calculated instead of $uConf$. From, Theorem 4 we know that $uConf^{pCap}(I) \geq uConf(I)$ i.e., $uConf^{pCap}$ serves as a boundary value like $ExpSup^{pCap}(I)$.

Moreover, from Lemma 2 if itemset I has $uConf^{pCap}(I)$ less than minimum support confidence $minUConf$, no further mining is required for its supersets. Hence, if some pattern P, fails to satisfy minimum support confidence $minUConf$ then WUIPM stops growing for that pattern P. Thus, a large number of non interesting itemsets will be pruned at once.

Alongside, using WUIP-tree it is very easy and cost effective to calculate $uConf^{pCap}$ of any itemset, I at any moment. At line 22 of Figure 3.4 a single comparison of $uConf^{pCap}$ with minimum support confidence $minUConf$ is added to prune non interesting or correlated items. Thus, WUIP-tree can be used to extract correlated patterns beside frequent patterns without any remarkable computation effort.

3.5.2 Effect of $uConf$ on Affinity Detection

The definition of $uConf$ is inspired from $all-confidence$ [48] measure i.e., any itemset I with $uConf(I)$ has at least a rule confidence value of $uConf(I)$ for any of its subsets. This means $uConf$ indicates the correlation among a collection of items instead of two items which is very useful. For example, in Table 3.1 pattern {b, c, e} has $uConf$ value 0.108 i.e., any rule from {b $\rightarrow$ c, c $\rightarrow$ b, c $\rightarrow$ e, e $\rightarrow$ c, b $\rightarrow$ e, e $\rightarrow$ b, b $\rightarrow$ ce, ce $\rightarrow$ b, bc $\rightarrow$ e, e $\rightarrow$ bc, be $\rightarrow$ c, c $\rightarrow$ be, ...} has atleast a $confidence$ value of $uConf(\{b, c, e\})$.

So, any itemset that satisfies $minUConf$ has a strong correlation or affinity among its items.

We have shown only to mine strong affinity patterns from WUIP-tree but from a users perspective sometimes it may be important to mine weak affinity patterns from uncertain databases. This can be done easily with a slight modification of our WUIPM algorithm.

3.6 Mining Weight Affinity Patterns from Uncertain Databases

Mining weighted correlation among items of an itemset I requires an extra list of $weights$ for items where $weight(i)$ is the weight or importance value of item i, $wUConf^{pCap}(I)$ groups the items that are correlated and have similar importance to users. It fulfils the goal because any itemset with $wUConf^{pCap}(I)$, satisfying minimum weighted support confidence, $minwUConf$ must have a rule confidence of $minwUConf$ for any pair of its subsets and it also groups the items of simlar importance.

3.6.1 Pruning of Weak Weight Affinity Patterns

Given, a threshold $minUConf$, WUIPM can prune weak weighted affinity patterns at any moment while mining process is running. For any itemset I, $wUConf(I)$ depends on expected support $ExpSup(I)$ and $ExpSup(x_i)$ where x_i is an item of I. But, using WUIP-tree we can find $ExpSup^{pCap}(I)$ instead of exact $ExpSup(I)$ of any itemset I alongside exact expected support of all the items (expected support of items are calculated at the beginning of WUIPM and stored for future use) i.e., $wUConf^{pCap}$ can be calculated instead of $wUConf$. From, Theorem 5 we know that $wUConf^{pCap}(I) \geq wUConf(I)$ i.e., $wUConf^{pCap}$ serves as a boundary value like $ExpSup^{pCap}(I)$.

Moreover, from Lemma 3 if itemset I has $wUConf^{pCap}(I)$ less than minimum weighted support confidence $minwUConf$, no further mining is required for its supersets. Hence, pruning technique is vastly improved as before.

3.6.2 Effect of $wUConf$ on Weight Affinity Detection

The definition of $wUConf$ is inspired from $all-confidence$ [48] measure i.e., any itemset I with $wUConf(I)$ has at least a rule confidence value of $wUConf(I)$ for any of its subsets. This means $wUConf$ indicates the correlation among a collection of items instead of two items which is very useful. For example, in Table 3.1 pattern {b, c, e}

has $wUConf$ value 0.0324 i.e., any rule from {b → c, c → b, c → e, e → c, b → e, e → b, b → ce, ce → b, bc → e, e → bc, be → c, c → be, ...} has atleast a weight $confidence$ value of $wUConf(\{b, c, e\})$. So, any itemset that satisfies $minWUConf$ has a strong correlation or affinity among its items respect to individual weight or importance.

Alongside, using WUIP-tree, $wUConf^{pCap}$ of any itemset, I can be calculated at any moment. At line 22 of Figure 3.4 a single comparison of $wUConf^{pCap}$ with minimum weighted support confidence $minwUConf$ is added to prune non interesting items.

Chapter 4

Experimental Results

In this chapter we have shown the experimental results that is achieved by our proposed algorithm. Based on several performance scale/parameter we have tried to verify our algorithms efficiency and depicted the performance throughout the chapter.

4.1 Experimental Settings

We have performed a comprehensive performance study in our experiments on both synthetic and real world datasets. The synthetic and real datasets are provided in [49]. Our performance tests show that WUIPM(Weighted Uncertain Interesting Pattern Mining) is a scalable and faster pattern mining technique. It can mine correlations with larger datasets with any amount of support and confidence threshold. Since we have proposed a completely new method of correlation mining so we have compared the performance of the method with itself to judge its scalability property and faster execution capability property for any level of threshold. But we have compared it with all the well-known frequent pattern discovery algorithms to indicate the efficacy of our algorithm with respect to frequent pattern, affinity pattern and weight affinity pattern mining. And we have also compared its performance with others to ensure the efficiency of our proposed method and measure in mining correlation among uncertain databases.

All programs are written in C/C++ programming language and run on a machine having 2.53GHz Intel Core i5 64-bit processor with OS X Mountain Lion installed in it. The running times provided includes CPU, I/O s, tree construction and false positive filtering. The results shown in this section are based on the average of multiple runs for each case. In all of the experiments, the WUIP-trees were constructed in descending order of expected support of items.

4.1.1 Uncertainty Value Generation

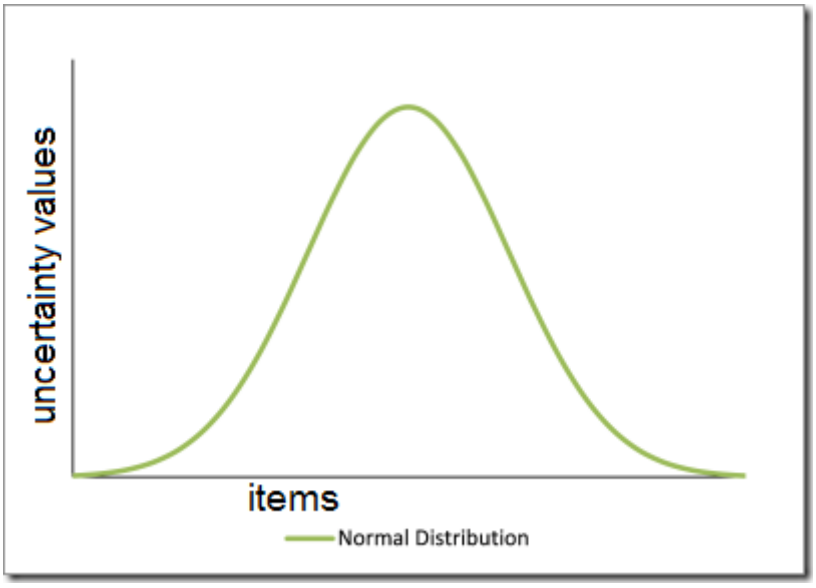


FIGURE 4.1: Probability Value Genertaion

From the frequent itemset mining repository [49] we got the synthetic and real life datasets that were collected for certain databases. To introduce uncertainty among these databases, we have used our own probabilistic approach and tools. We have generated a normal distribution for the distinct items of each dataset. The reason for choosing normal or gaussian distribution is most real life transactional data follows a gaussian distribution where extreme values are rare and other values are evenly distributed on the both side of the mean value. Then, we have associated each item of a database with its corresponding uncertainty value to generate a new uncertain database. In this way, our datasets attain attribute and tuple level uncertainty besides having an uncertainty distribution model.

4.1.2 Weight or Importance Value Generation

To generate weight values for items in databases, we have used our own probabilistic approach and tools. We have generated a normal distribution for the distinct items of each dataset. The reason for choosing normal or gaussian distribution is most real life transactional data follows a gaussian distribution where extreme values are rare and other values are evenly distributed on the both side of the mean value. For example, an item with extreme weight value is like to occur less than an item with average weight or importance. This can be described using our motivating example in Section 1.2 where disease like AIDS and Leukemia had greater importance. Although, it depends on user's requirement but the general case is described here. We have given users the flexibility

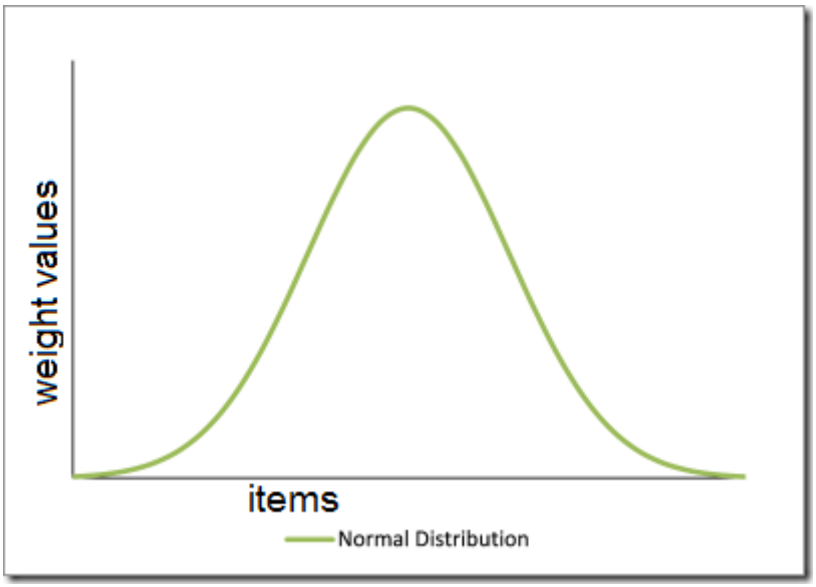


FIGURE 4.2: Weight or Importance Value Genertaion

to assign weight values according to their need. Then, we have associated each item of a database with its corresponding uncertainty value to generate a new uncertain database. In this way, our datasets attain attribute and tuple level uncertainty besides having an uncertainty distribution model.

4.2 Performance Metrics

we have considered several metrics as parameter to evaluate the performance of our proposed algorithm. The parameters are listed below:

- Processing time vs minimum expected support
- Number of false positives vs minimum expected support
- Memory in number of tree nodes vs minimum expected support
- Number of patterns vs minimum expected support
- Number of patterns vs correlation measure $uConf$
- Processing time vs correlation measure $uConf$
- Number of patterns vs correlation measure $wUConf$
- Processing time vs correlation measure $wUConf$

Dataset Name	Total Transactions	Distinct Items
mushroom	8124	120
kosarak	990002	41270
pumsb_star	49046	2088
chess	3196	75
$T10I4D100K$	100000	869

TABLE 4.1: Datasets of Experimental Results

4.3 Experimental Environment

To evaluate the performance of our algorithm we have checked it with real life and synthetic data. And in both cases our algorithm is proved to be sound and efficient. Our algorithm performs well in both type of dataset and was scalable, faster and efficient enough that it can mine frequent, affinity and weight affinity patterns with any size and any level of uncertain database.

4.3.1 Real life Dataset

For real life data sets we have used *mushroom* [49], *chess* [49], *kosarak* [49] and *pumsb_star* [49]. Among them *mushroom*, *chess* and *pumsb_star* are dense datasets. *mushroom* has 8124 transactions with 120 distinct items, *chess* has 3196 transactions with 75 distinct items and *pumsb_star* has 49046 transactions with 2088 distinct items. *kosarak* is a large and sparse data set with 990002 transactions and 41270 distinct items.

4.3.2 Synthetic Dataset

For synthetic data sets we have used $T10I4D100K$ [49]. It is an IBM generated transactional data set widely used for frequent pattern mining. It is a sparse data set with 100000 transactions and 869 distinct items.

4.3.3 Performance Analysis

To test the performance of our proposed correlation measures, we have thoroughly conducted extensive tests using two real life datasets *mushroom* and *kosraka*. Both are widely used large scale datasets and we have chosen *mushroom* as a candidate for dense dataset and *kosarak* for sparse dataset, *mushroom* has 8124 transactions with 120 distinct items and *kosarak* has 990002 transactions with 41270 distinct items. The results of the experiments are given below:

4.3.3.1 Performance analysis of uConf measure

The proposed $uConf$ measure helps to mine the affinity patterns. It drastically improves pruning of weak affinity patterns because of having anti-monotone property. The tests of $uConf$ have been conducted for both number of patterns and processing time. In all the cases, it has successfully proved to be worthy of mining strong affinity patterns within a short time.

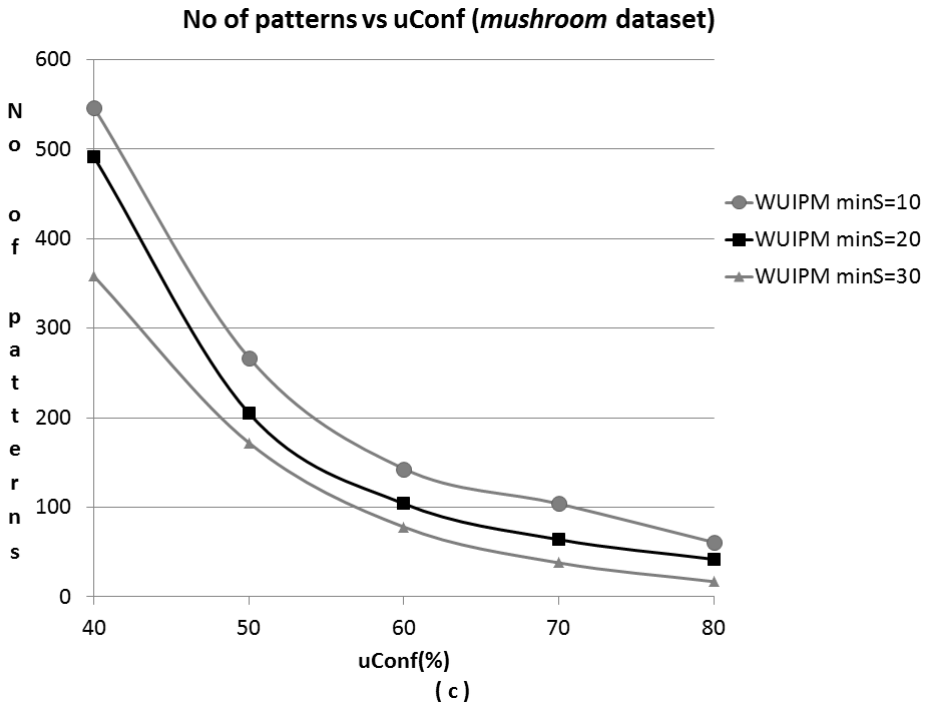


FIGURE 4.3: number of patterns vs uConf for mushroom database

Number of affinity patterns

In Figure 4.3, the number of generated patterns for various $uConf$ values are shown with different minimum support values for *mushroom* dataset. Here, number of patterns gradually decreased with increasing values for $uConf$ which was expected i.e., stronger affinity patterns was generated. These tests were done for several minimum support value. For a high $minSup$ and $uConf$ value, the number of patterns was the lowest because it extracted only the most correlated frequent patterns.

Similar experiments were conducted for *kosarak* dataset which is shown in Figure 4.4. Number of patterns gradually decreased with increasing values for $uConf$ like before but the difference between two minimum support values were more conspicuous than *mushroom* dataset. This happened because *kosarak* is a sparse data set.

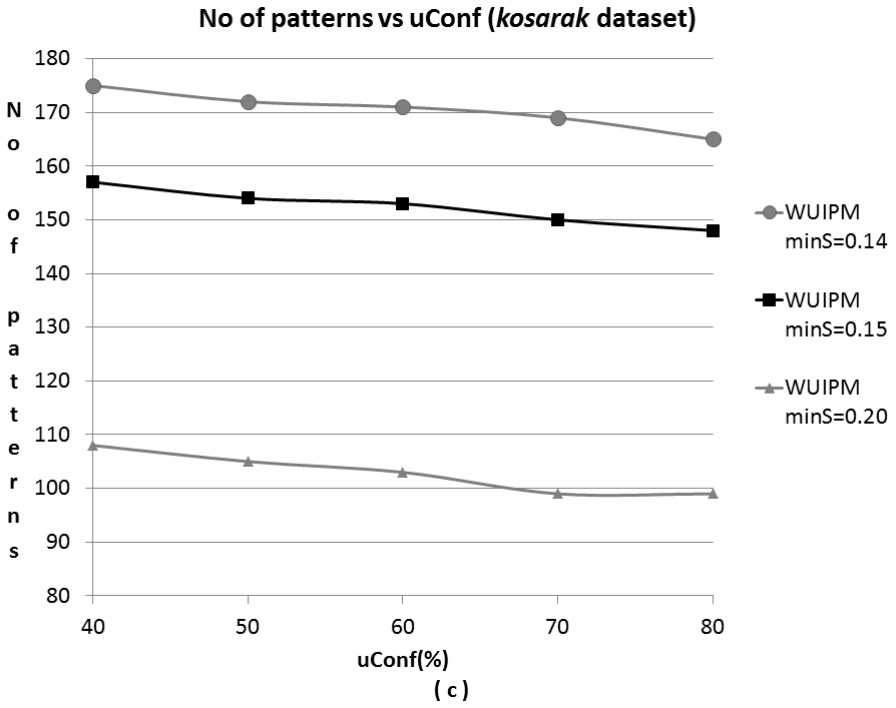


FIGURE 4.4: number of patterns vs uConf for kosarak database

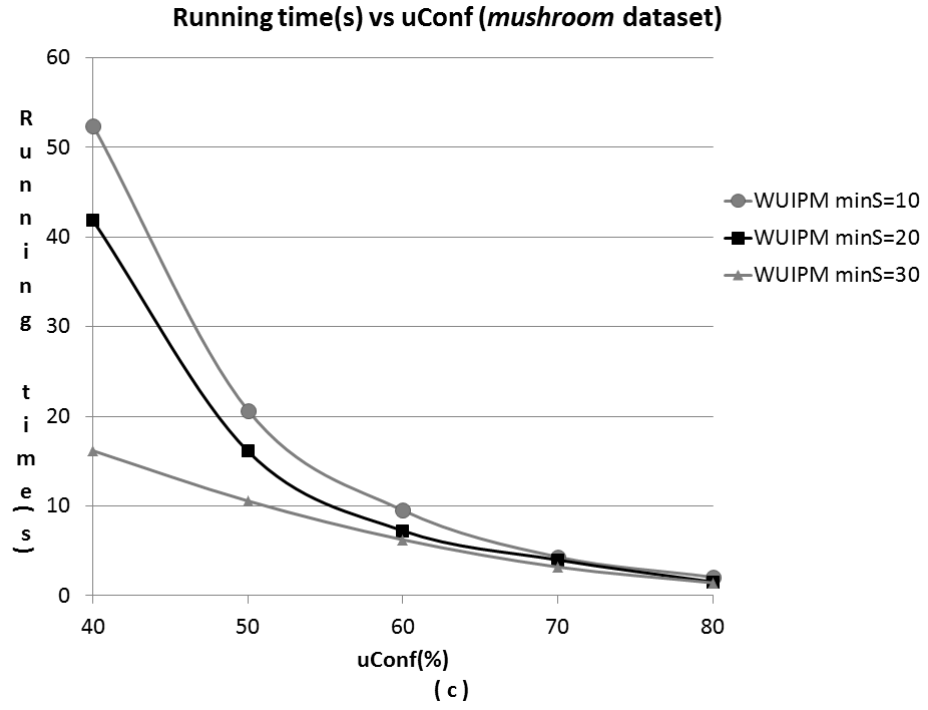


FIGURE 4.5: processing time vs uConf for mushroom database

Processing Time

For different minimum support values and $uConf$ values, processing time were recorded as shown in Figure 4.5 and 4.6. Processing time gradually decreased with increasing

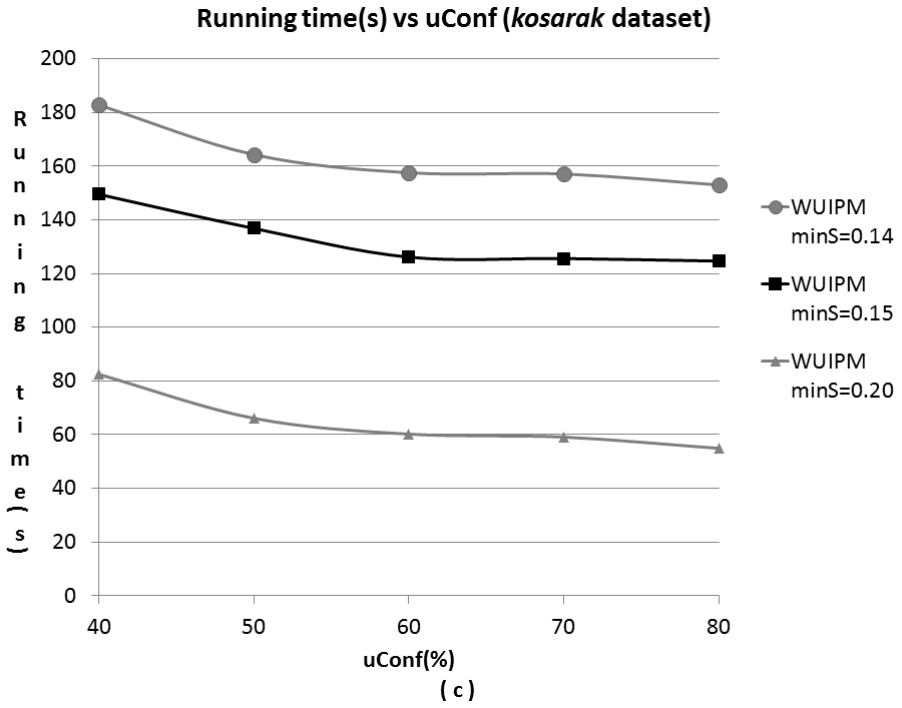


FIGURE 4.6: processing time vs uConf for kosarak database

$uConf$ values for *mushroom* and the difference among various minimum support values were more visible for low $uConf$ values.

Similarly, processing time gradually decreased with increasing $uConf$ values for *kosarak* but the difference among various minimum support values were significantly noticeable here even for a high $uConf$ because *kosarak* has more transactions than *mushroom* and it is sparse. Overall, $uConf$ passed the tests with significant achievement.

4.3.3.2 Performance analysis of wUConf measure

The proposed $wUConf$ measure helps to mine the weight affinity patterns. It remarkably improves pruning of non weight affinity patterns because of having anti-monotone property. The tests of $wUConf$ have been conducted for both number of patterns and processing time. For each, it has successfully mined weight affinity patterns within very fast.

Number of affinity patterns

In Figure 4.7, the number of generated patterns for various $wUConf$ values are shown with different minimum support values for *mushroom* dataset. Here, number of patterns

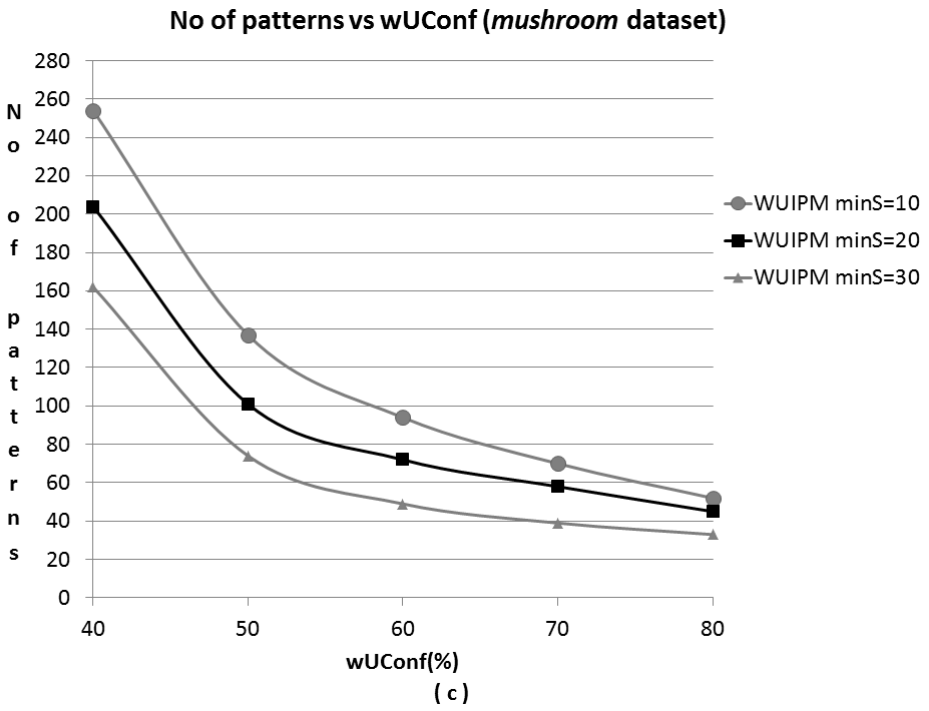


FIGURE 4.7: number of patterns vs wUConf for mushroom database

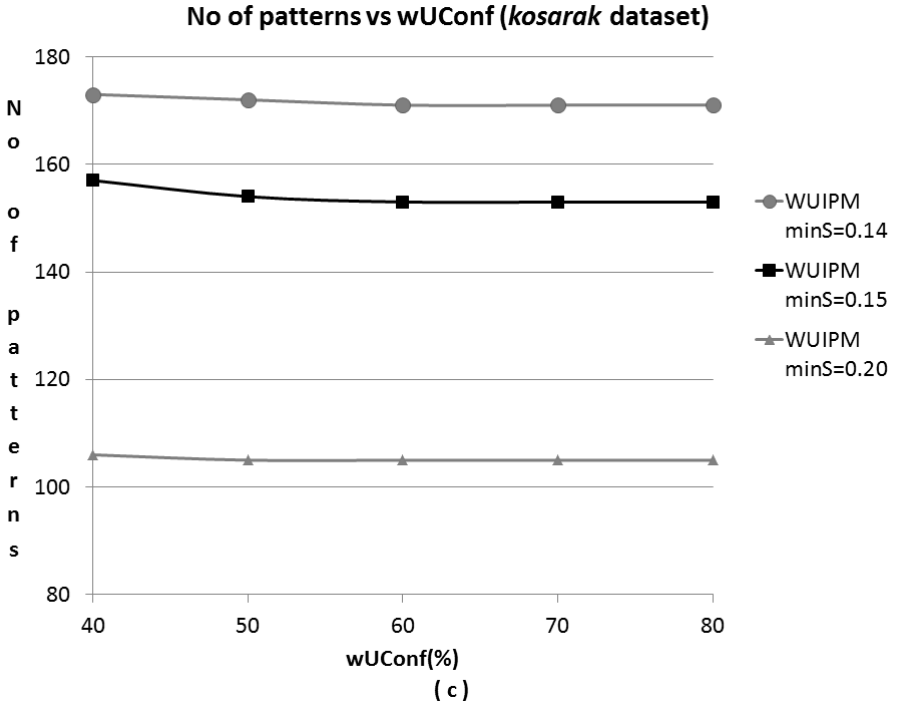


FIGURE 4.8: number of patterns vs wUConf for kosarak database

gradually decreased with increasing values like Figure 4.3 for $wUConf$ values i.e., patterns with more weight similarity were generated. These experiments were conducted for several minimum support values. For a high $minSup$ and $wUConf$ value, the number

of patterns was the lowest because it extracted only the most similar patterns regrading individual weight

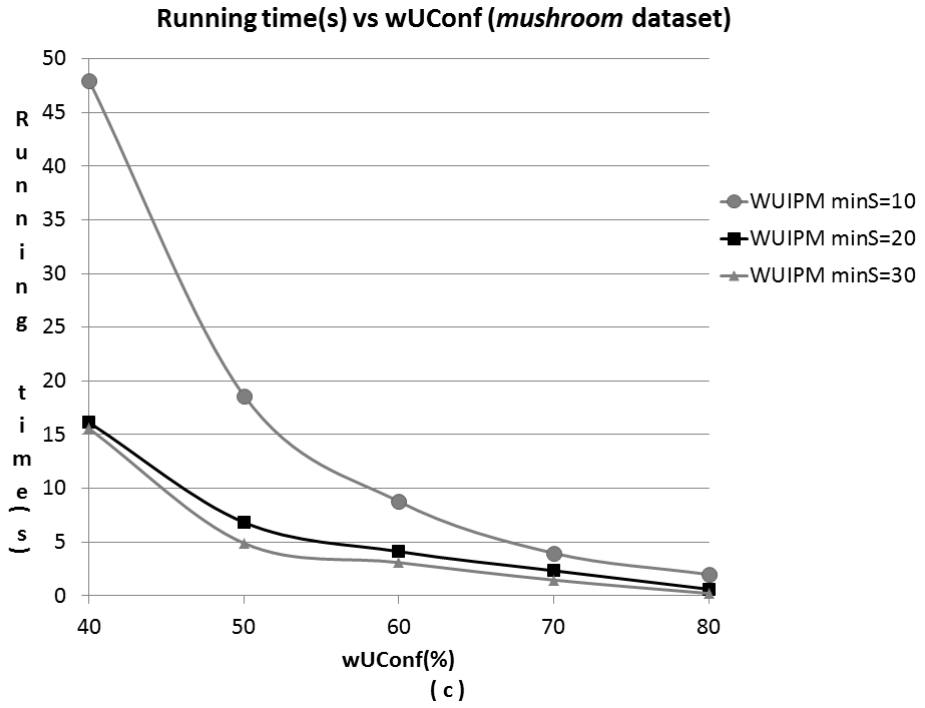


FIGURE 4.9: processing time vs wUConf for mushroom database

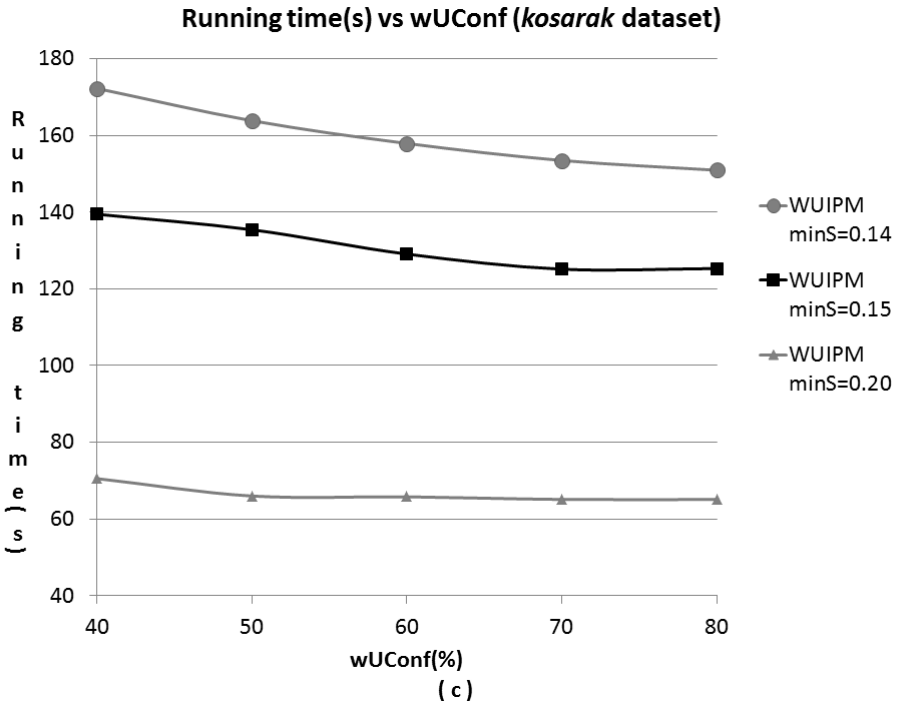


FIGURE 4.10: processing time vs wUConf for kosarak database

Processing Time

In Figure 4.9 and 4.10, processing time were recorded as shown for different minimum support values and $wUConf$s, . Processing time gradually decreased with increasing $wUConf$ values for *mushroom* and the difference among various minimum support values were more visible for low $wUConf$ values.

Similarly, processing time gradually decreased with increasing $wUConf$ values for *kosarak* but the difference among various minimum support values were significantly noticeable here even for a high $wUConf$ because *kosarak* has more transactions than *mushroom* and it is sparse. Altogether, $wUConf$ was successful to achieve its goal.

4.3.4 Comparison With Existing Algorithms

To compare WUIPM with other approaches we chose two of the most recent and efficient algorithms, PUF-growth [32] and CUF-growth* [31]. Extensive experiments were conducted to compare number of patterns, number of false positives, running time and number of memory as tree nodes on *mushroom*, *kosarak*, *chess*, *pumsb_star* and $T10I4D100K$.

WUIPM is capable of mining frequent, affinity and weight affinity patterns altogether which is unique and new. Other approaches e.g., PUF-growth [32] and CUF-growth* [31] can only mine frequent patterns. Hence, while comparing we have included the result of WUIPM compared to others when correlation measures were not applied i.e., $minUConf$ and $minWUConf$ values were 0. Alongside, we conducted experiments including correlation measures. This helps to understand the frequent pattern mining power of WUIPM besides correlation mining.

4.3.4.1 Number of Interesting Patterns Comparison

In Figure 4.11 and 4.12 the number of generated patterns are shown for PUF-growth [32], CUF-growth* [31], WUIPM without $uConf$ and $wUConf$, WUIPM with $uConf$ and $wUConf$ on *kosarak* and *mushroom* dataset. Here, it showed that WUIPM was much better than existing algorithms even when affinity measures were not present. It generated less number of patterns i.e, less number of false positives. Only WUIPM performed better in frequent pattern mining because of associated $pProxy$ value that helped to prune infrequent patterns and when $uConf$ and $wUConf$ were applied, it mined the most interesting patterns which are frequent and correlated.

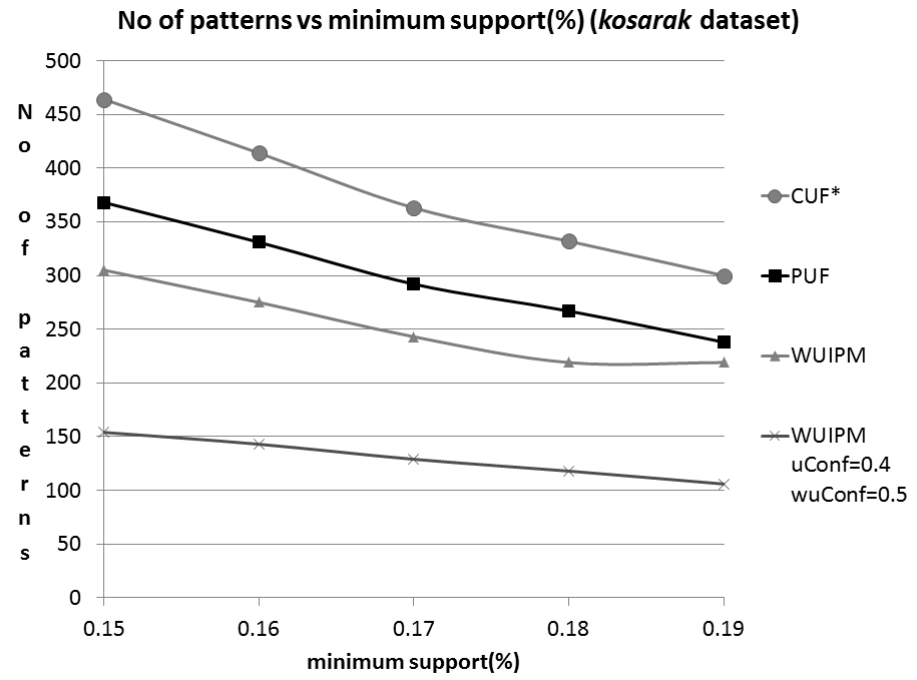


FIGURE 4.11: Number of patterns comparison for kosarak database

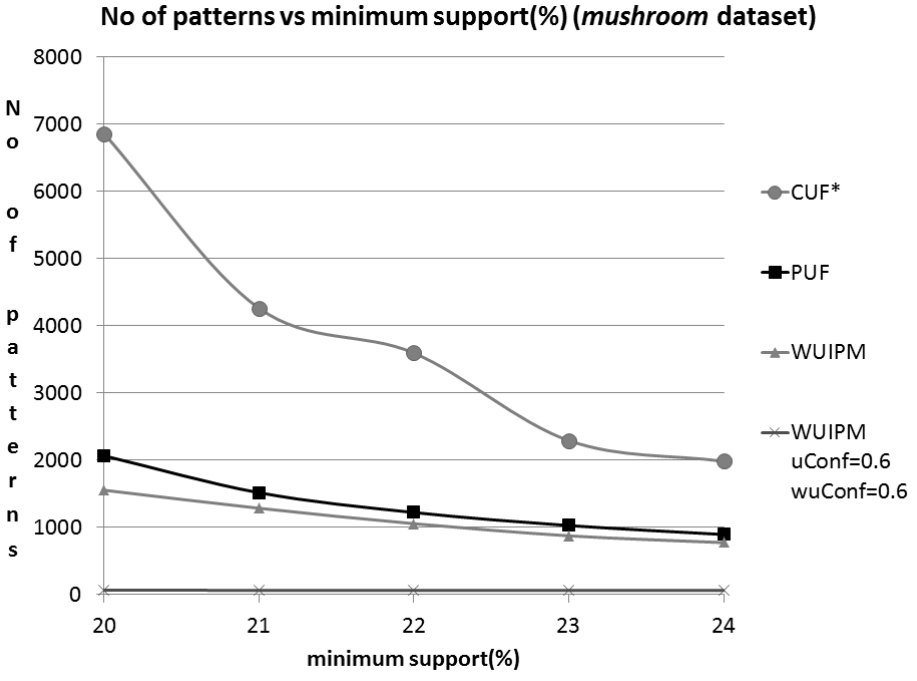


FIGURE 4.12: Number of patterns comparison for mushroom database

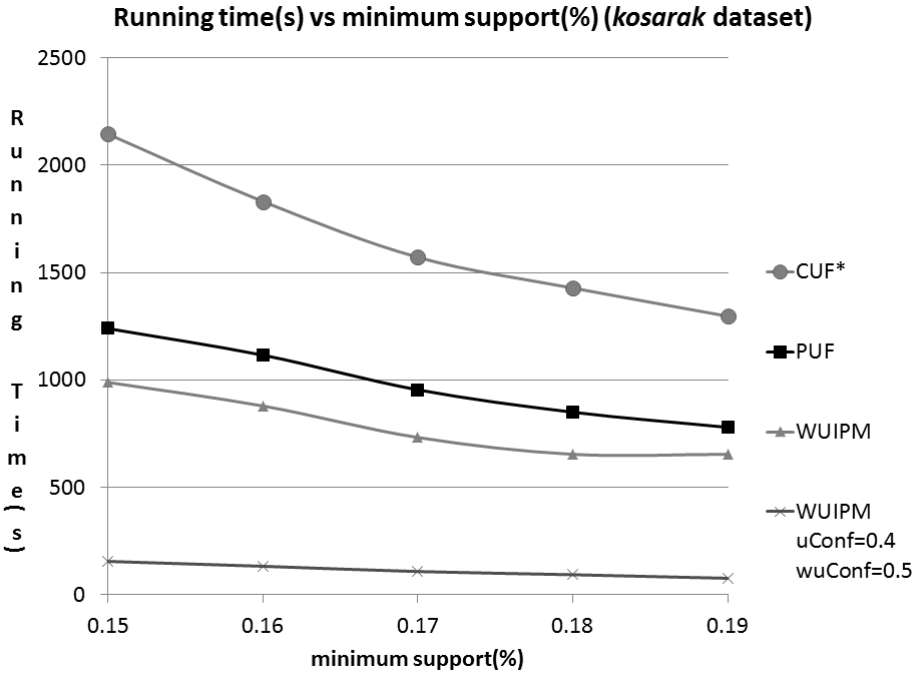


FIGURE 4.13: Running time comparison for kosarak database

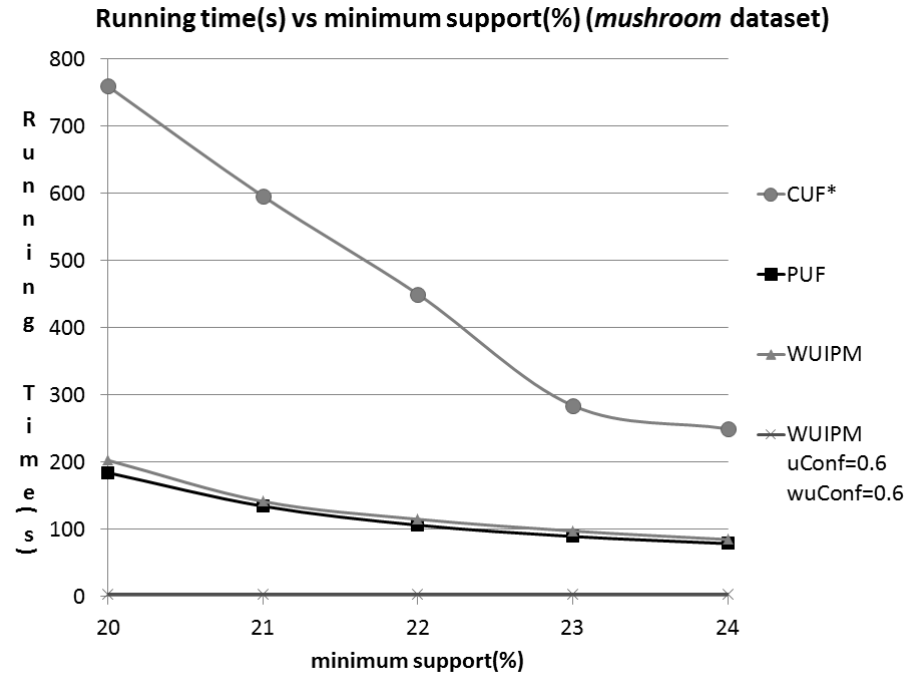


FIGURE 4.14: Running time comparison for mushroom database

4.3.4.2 Running Time Comparison

In Figure 4.13 and 4.14 running times are shown for PUF-growth [32], CUF-growth* [31], WUIPM without $uConf$ and $wUConf$, WUIPM with $uConf$ and $wUConf$ on

kosarak and *mushroom* dataset. Here, it showed that WUIPM was much faster than existing algorithms even when affinity measures were not present. It generated less number of false positives that lead to faster running time. Only WUIPM performed faster in frequent pattern mining because of associated $pProxy$ value that helped to prune infrequent patterns and when $uConf$ and $wUConf$ were applied, it pruned the non interesting patterns before hand. Besides, for following anti-monotone property, these measures were integrated with existing WUIP-tree and growth mining approach without any extra computational effort.

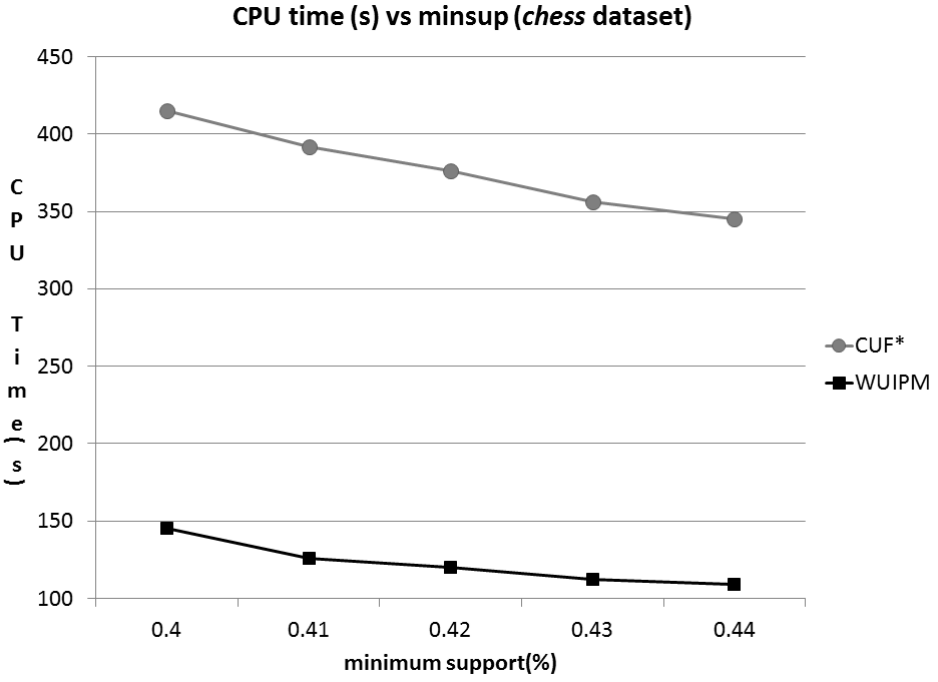


FIGURE 4.15: Running time comparison for chess database

Running time comparison on *chess* [49], *pumsb_star* [49] and $T10I4D100K$ [49] is shown in Figure 4.15, 4.16 and 4.17 for mining frequent patterns. In each dataset WUIPM was far better than others while extracting frequent itemsets because WUIPM pruned the infrequent ones comprehensively.

4.3.4.3 Number of False Positives Comparison

Because of using estimated values instead of actual expected support as other existing approaches, WUIPM had false positives too i.e., there were some patterns estimated to be frequent patterns but proved to be infrequent while filtering process of false positives. Experiments conducted on various types of data sets showed that WUIPM generated

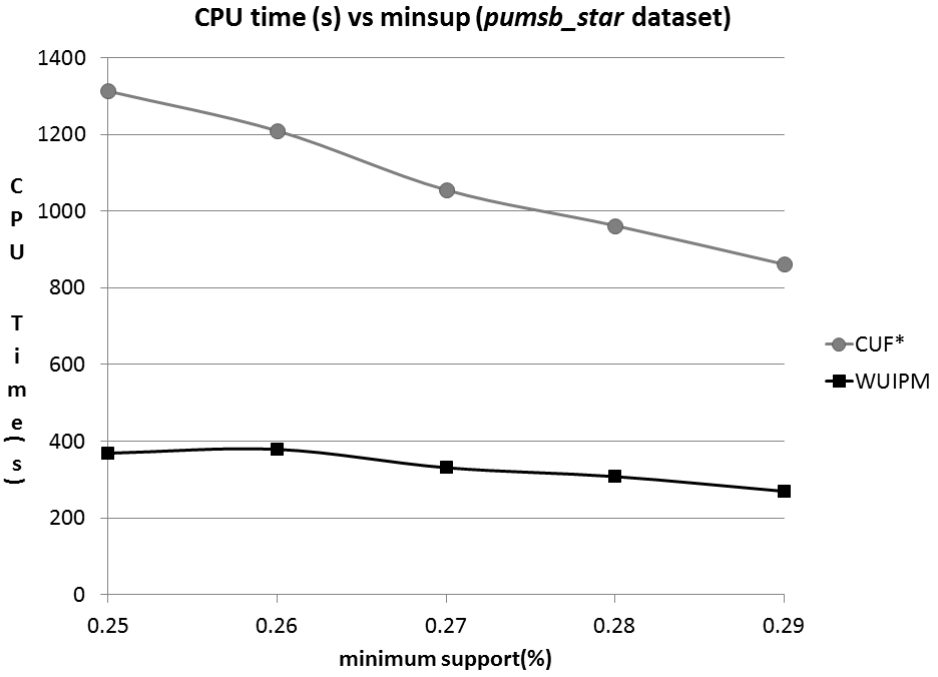


FIGURE 4.16: Running time comparison for pumsb star database

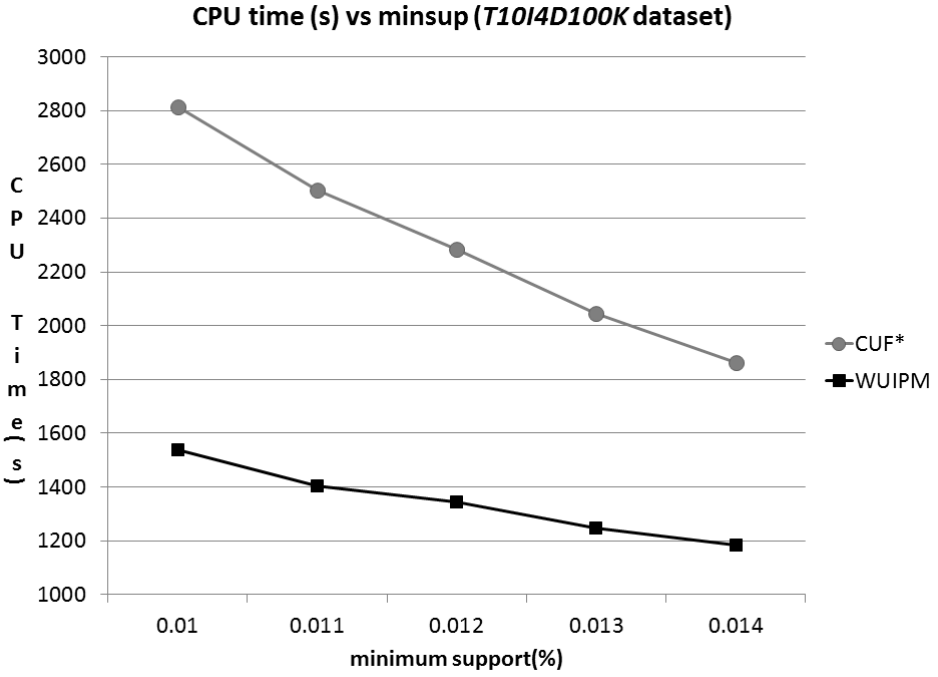


FIGURE 4.17: Running time comparison for T10I4D100K star database

fewer false positives than others for using $pProxy$ value to calculate the boundary values and had a tighter limit.

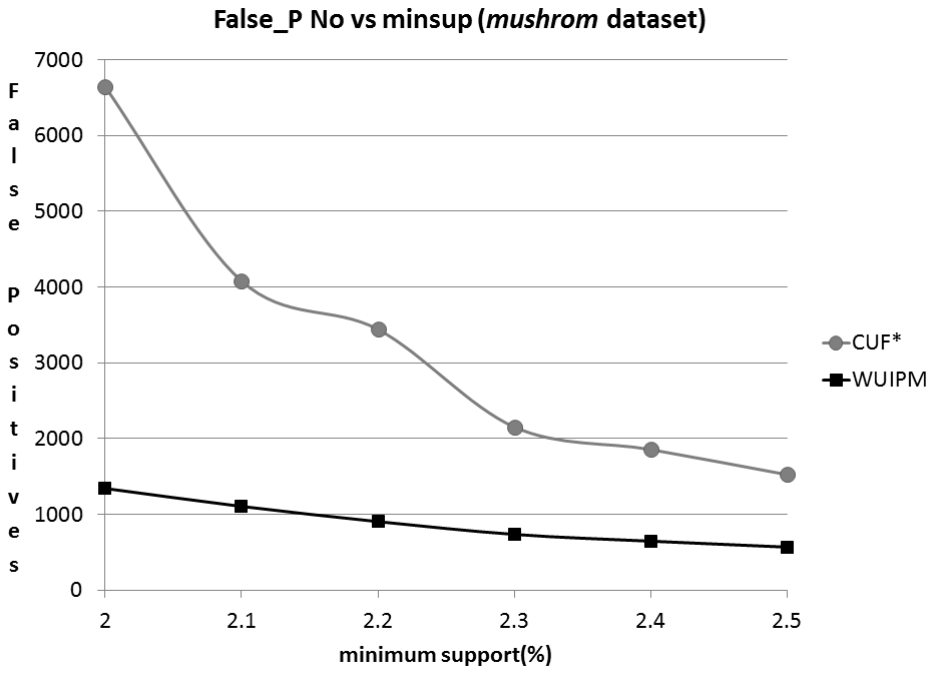


FIGURE 4.18: Number of false positives comparison for mushroom database

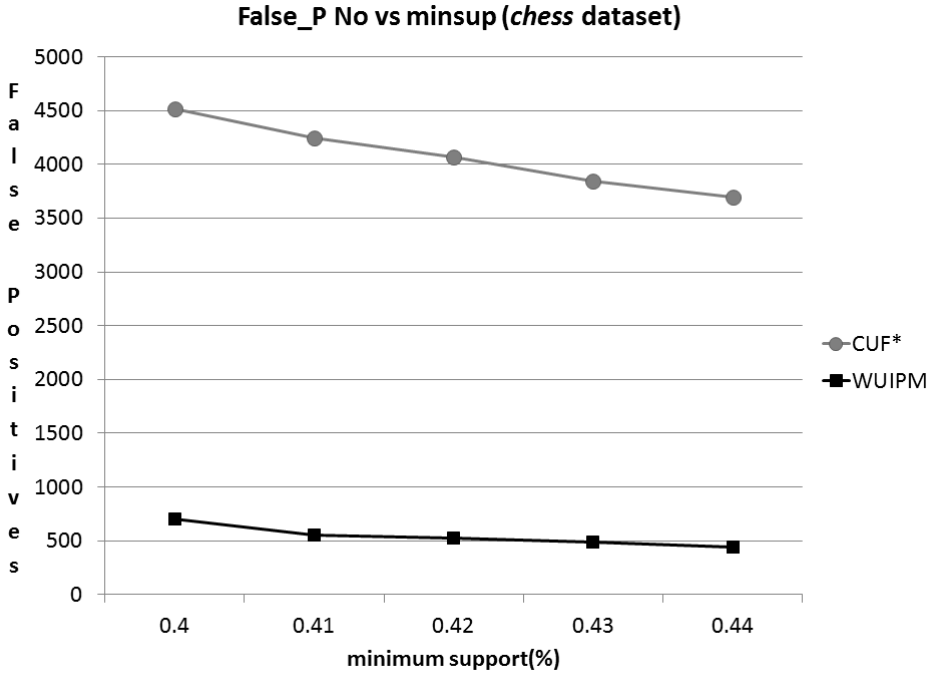


FIGURE 4.19: Number of false positives comparison for chess database

Figures 4.23, 4.24 and 4.20 show the number of false positives comparison for three dense and real life datasets i..e., *mushroom* [49], *chess* [49] and *pumsb_star* [49]. The difference in number of false positives was significant and this dissimilarity increased for

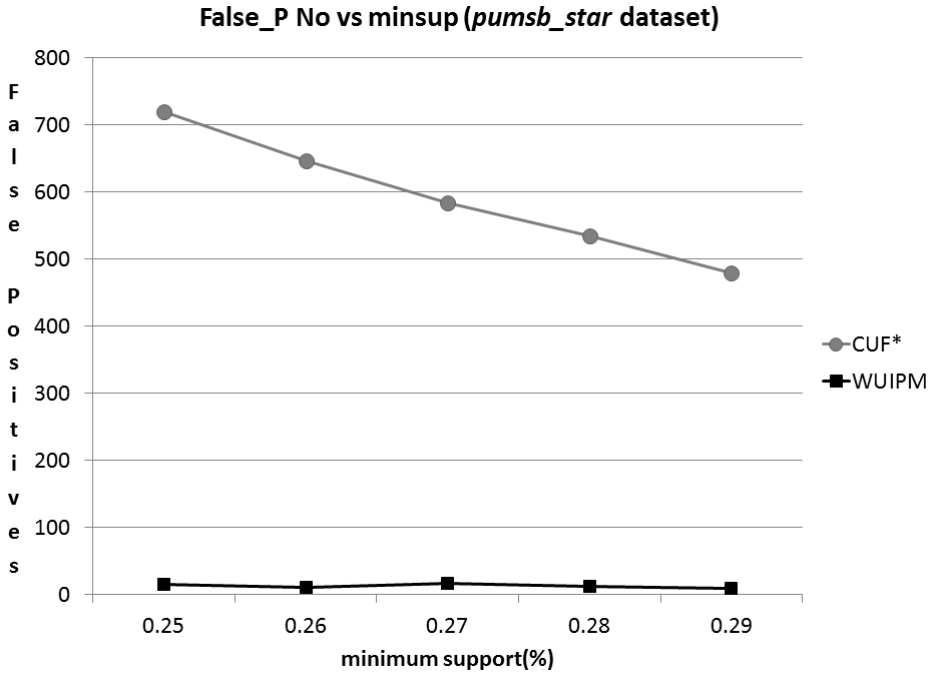


FIGURE 4.20: Number of false positives comparison for pumsb star database

lower minimum support.

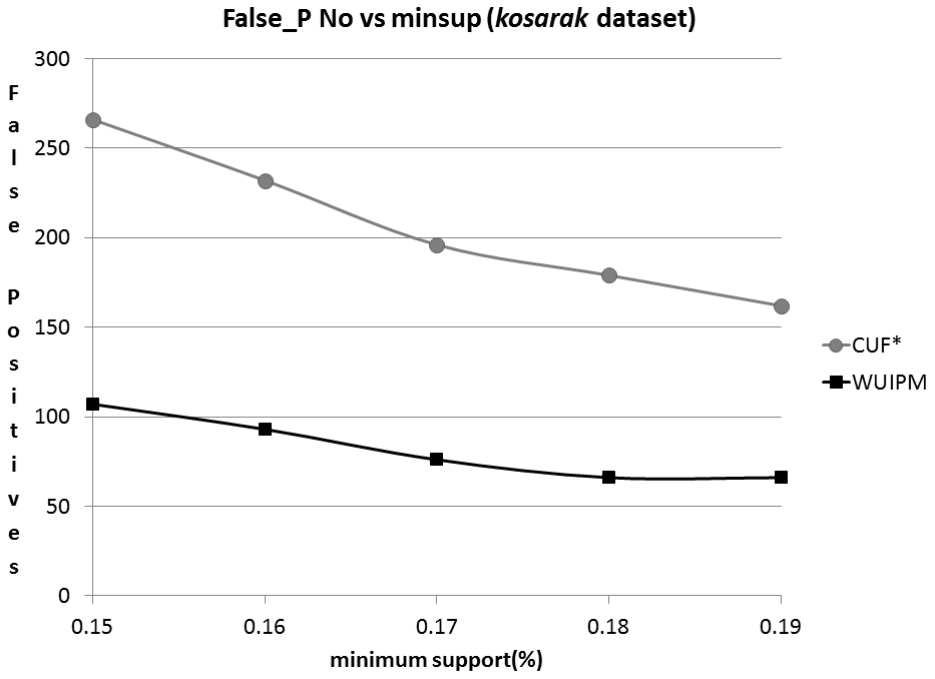


FIGURE 4.21: Number of false positives comparison for kosarak star database

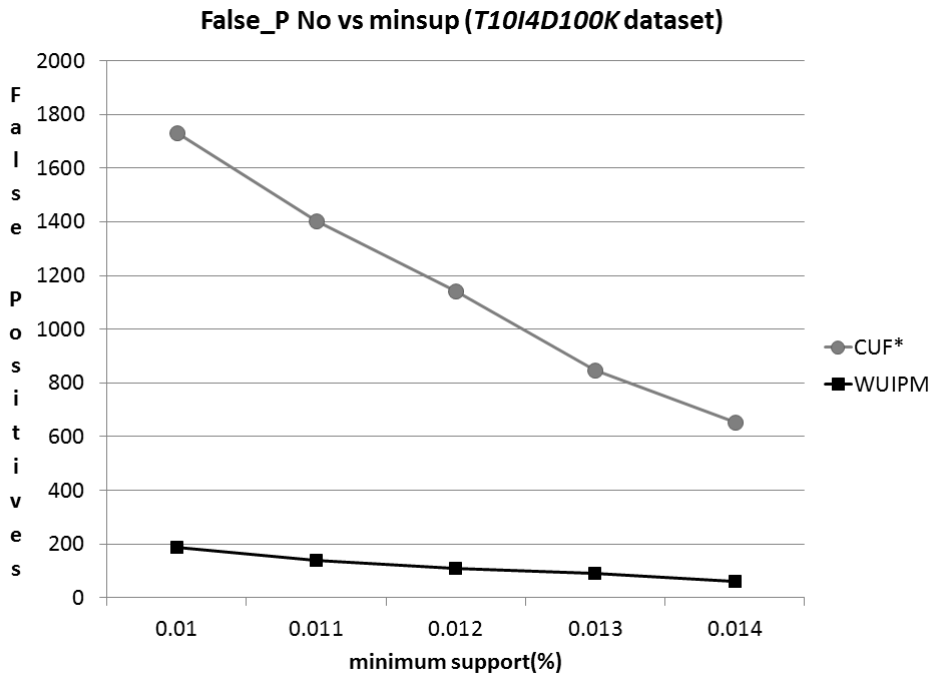


FIGURE 4.22: Number of false positives comparison for T10I4D100K star database

Figures 4.21 and 4.22 show the number of false positives comparison for two sparse datasets i..e., *kosarak* [49] and $T10I4D100K$ [49]. *kosarak* [49] is a real life dataset and $T10I4D100K$ [49] is synthetic dataset generated by IBM as a test database for frequent pattern mining. WUIPM generated less number of false positives for the sparse datasets too.

4.3.4.4 Memory Comparison

In a growth mining algorithm which generally uses a tree structure can be evaluated in maximum number of required tree nodes at any instance while mining process is running. This indicates the required memory for the particular mining algorithm. So, we compared WUIPM with others in terms of number of tree nodes and WUIPM was significantly better.

Figures **??**, **??** and 4.25 show the number of false positives comparison for three dense and real life datasets i..e., *mushroom* [49], *chess* [49] and *pumsb_star* [49]. The difference in required memory was remarkable because in the intermediate steps of mining WUIPM pruned a lot of unnecessary nodes.

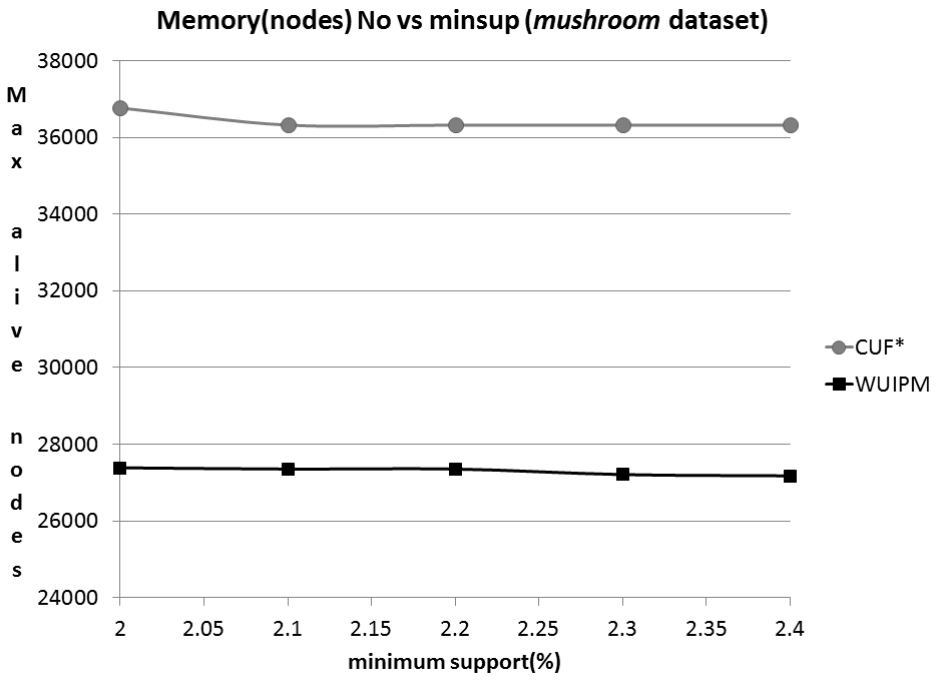


FIGURE 4.23: Memory comparison for mushroom database

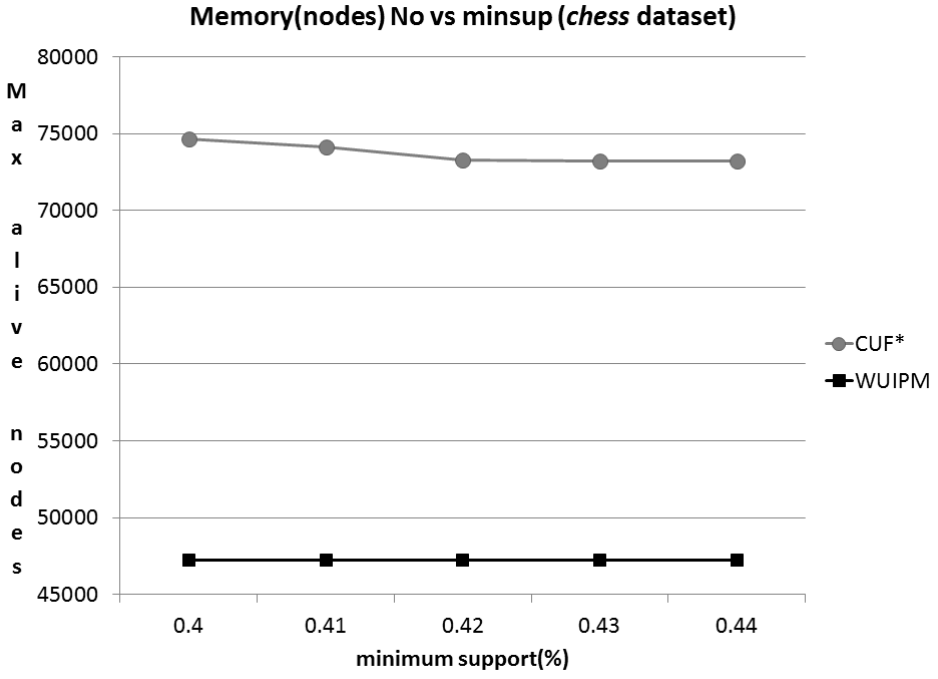


FIGURE 4.24: Memory comparison for chess database

Figures 4.26 and 4.27 show the memory comparison for two sparse datasets i..e., *kosarak* [49] and $T10I4D100K$ [49]. *kosarak* [49] is a real life dataset and $T10I4D100K$ [49] is synthetic dataset generated by IBM as a test database for frequent pattern mining.

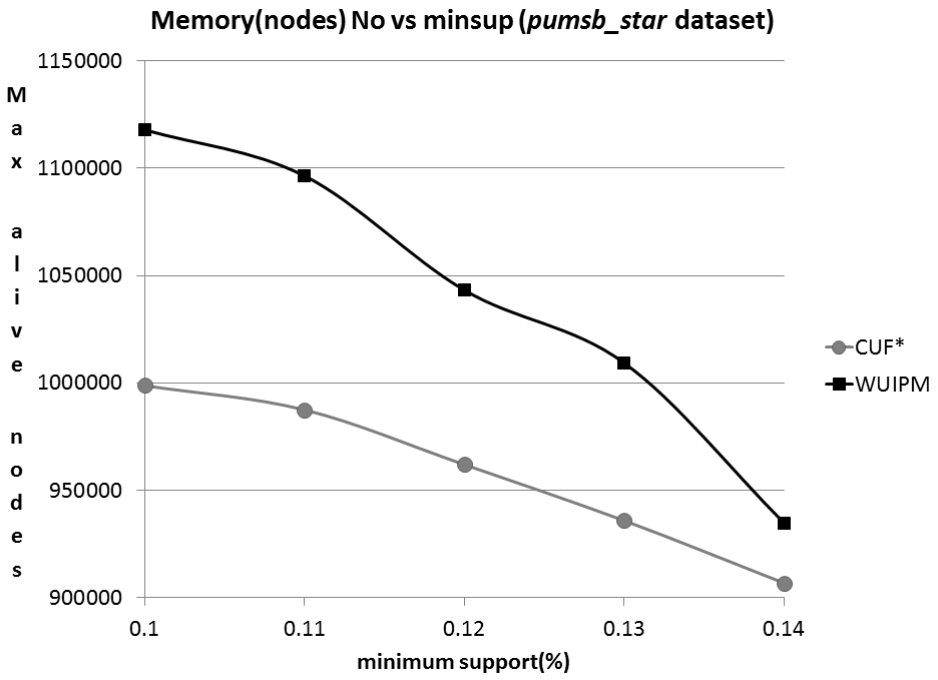


FIGURE 4.25: Memory comparison for pumsb star database

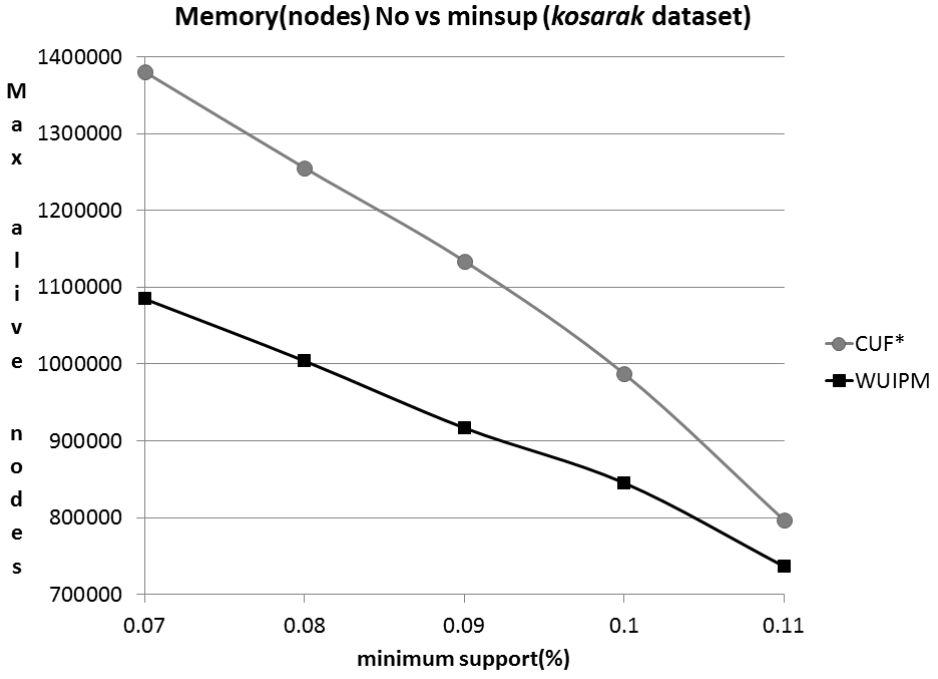


FIGURE 4.26: Memory comparison for kosarak star database

WUIPM required less memory for the sparse datasets too.

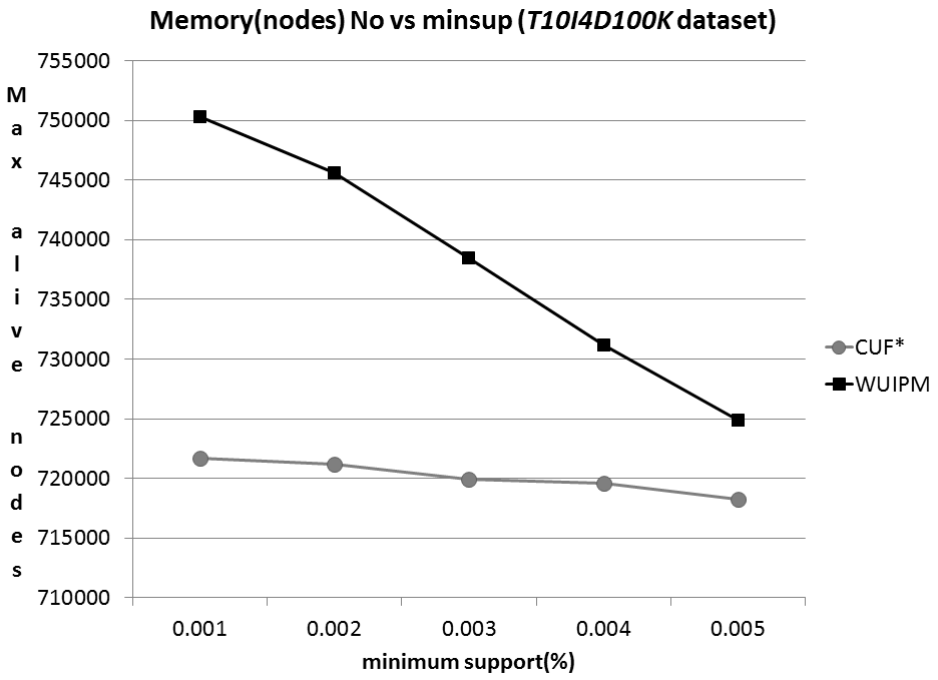


FIGURE 4.27: Memory comparison for T10I4D100K star database

4.4 Summary

To summarize, WUIPM algorithm stores uncertain database very efficiently using WUIP-tree and mines interesting (frequent, affinity, weight affinity) patterns from WUIP-tree. Overall, WUIPM was faster and required less memory than others. Alongside, WUIPM provided features to mine correlated interesting patterns both for weight similarity and support similarity.

Chapter 5

Applications

Most of the real life datasets have uncertainty within the collected data. The emerging modern applications such as information integration, scientific databases, sensor data management, information extraction on the web, entity resolution requires to manage uncertain data. A few examples of uncertain data mining can be:

5.1 Underground Coal mine Monitoring with Sensor Networks

Environment monitoring in coal mines is an important application of wireless sensor networks (WSNs) [50] that has commercial potential. It has a design of a Structure-Aware Self-Adaptive WSN system, SASA. By regulating the mesh sensor network deployment and formulating a collaborative mechanism based on a regular beacon strategy, SASA is able to rapidly detect structure variations caused by underground collapses. It further develops a sound and robust mechanism for efficiently handling queries under instable circumstances. A prototype is deployed with 27 mica2 motes in a real coal mine. This kind of system generates huge amount of uncertain data from sensor networks which is required to be mined efficiently to deliver the correct results.

5.2 Privacy Preserving Data Mining

The importance of privacy preservation in data mining [40] has been recognized recently. Privacy preserving data mining algorithms aim at discovering accurate knowledge/patterns while avoiding actual access to sensitive individual information in data. To achieve this, a common approach is to randomize/distort the real dataset, so that the true value

for a particular instance can not be inferred from its randomized counterpart with probabilities better than a predefined threshold, and the data mining algorithms are performed on the randomized/distorted dataset instead. These distorted datasets are uncertain in nature.

5.3 Trajectory or Moving Object Search

Predicting path [51] for cyclones or missiles and searching for moving objects (e.g., rescue or military operations, searching objects in space using image data) produces data that is uncertain. So, efficient uncertain data mining techniques are required to give them the desired outcomes.

Overall, many real life applications exist where our proposed approach would serve as a valuable tool. It can also be modified to cope with all the problems related to uncertain data mining.

Chapter 6

Conclusions

In this thesis we proposed a new mining approach for uncertain databases with a growth mining algorithm, WUIPM and a tree data structure WUIP-tree. We proposed several new measures for reducing memory and running time. $pProxy$ 7 is one of them. We have shown the importance of correlation mining in uncertain data bases and to our knowledge, we are the first to address this issue for uncertain databases. Then, we have given several new measures for mining correlation among itemsets. $uConf$ 9 and $wUConf$ 11 were among the correlation measures. This chapter provides the summary of this thesis and future scopes.

6.1 Research Summary

In this thesis, we discussed the emerging of uncertain databases in data mining. We elaborately described how uncertain comes to our regualr databases and the importance of mining uncertain data. We analyzed the existing approaches for mining uncertain data and discovered their faults. We proposed a new data structure WUIP-tree to store uncertain databases and an efficient algorithm, WUIPM to mine using the WUIP-tree. We proposed several new measure for reducing memory and running time of frequent pattern mining.

After mining frequent patterns, we moved to correlation mining in uncertain databases. To our knowledge, we were the first to mine correlations among patterns in uncertain databases. We proposed new measures and algorithm for mining support affinity patterns. Alongside, we discovered and analyzed the importance of weight for individual items in uncertain databases. Then, we proposed new measures and technique for mining weight affinity patterns.

In the experimental results, we have compared the performance of our proposed approach, WUIPM respect to existing approaches and we also analyzed the proposed measures with varying parameters. We conducted experiments on various(e.g., rela life, synthetic, dense, sparse etc.) types of data sets and in all the cases WUIPM performed remarkably better. After presenting the experimental results we showed some real life applications that use uncertain databases and its mining approaches to help understand the importance of uncertain data mining and the applicability of our approach.

6.2 Future Scopes

As the domain of uncertain data mining is very new, there are definite scopes to extend and use our proposed approach as a tool for further research.
First, we build our approach to work around *expected support* which can be modified to work with the *probablilistic support*.

Second, as a base step we improved the frequent pattern mining process alongside extracting correlations in uncertain databases. This work can be extended for mining association rules, closed frequent itemsets etc. for uncertain databases.

In this chapter, we have discussed the research summary alongside some future scopes for our proposed approach. The future scopes stated above shows a vast opportunity to contribute further in this domain. We have strong belief and hope to explore further and enrich the field of uncertain data mining.

Bibliography

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules in large databases," in *Proceedings of the 20th International Conference on Very Large Data Bases*, ser. VLDB '94. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1994, pp. 487–499. [Online]. Available: http://dl.acm.org/citation.cfm?id=645920.672836

[2] J. Han, J. Pei, Y. Yin, and R. Mao, "Mining frequent patterns without candidate generation: A frequent-pattern tree approach," *Data Min. Knowl. Discov.*, vol. 8, no. 1, pp. 53–87, Jan. 2004. [Online]. Available: http://dx.doi.org/10.1023/B:DAMI.0000005258.31418.83

[3] R. Srikant and R. Agrawal, "Mining sequential patterns: Generalizations and performance improvements," in *Research Report RJ 9994, IBM Almaden Research*, 1995.

[4] J. Pei, J. Han, B. Mortazavi-Asl, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu, "Mining sequential patterns by pattern-growth: The prefixspan approach," *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 11, pp. 1424–1440, 2004.

[5] C. K.-S. Leung and Q. I. Khan, "Dstree: A tree structure for the mining of frequent sets from data streams," in *ICDM*, 2006, pp. 928–932.

[6] S. K. Tanbeer, C. F. Ahmed, and B.-S. Jeong, "Mining regular patterns in data streams," in *DASFAA (1)*, 2010, pp. 399–413.

[7] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "Efficient tree structures for high utility pattern mining in incremental databases," *IEEE Trans. Knowl. Data Eng.*, vol. 21, no. 12, pp. 1708–1721, 2009.

[8] X. Yan and J. Han, "gspan: Graph-based substructure pattern mining," in *ICDM*, 2002, pp. 721–724.

[9] K. Gade, J. Wang, and G. Karypis, "Efficient closed pattern mining in the presence of tough block constraints," in *Proceedings of the Tenth ACM SIGKDD*

International Conference on Knowledge Discovery and Data Mining, ser. KDD '04. New York, NY, USA: ACM, 2004, pp. 138–147. [Online]. Available: http://doi.acm.org/10.1145/1014052.1014070

[10] J. Pei, J. Han, and R. Mao, "Closet: An efficient algorithm for mining frequent closed itemsets." in *ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, 2000, pp. 21–30. [Online]. Available: http://dblp.uni-trier.de/db/conf/dmkd/dmkd2000.html#PeiHM00

[11] J. Wang, J. Han, and J. Pei, "Closet+: searching for the best strategies for mining frequent closed itemsets," in *KDD '03: Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*. New York, NY, USA: ACM Press, 2003, pp. 236–245.

[12] J. Wang, J. Han, Y. Lu, and P. Tzvetkov, "Tfp: An efficient algorithm for mining top-k frequent closed itemsets." *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 5, pp. 652–664, 2005. [Online]. Available: http://dblp.uni-trier.de/db/journals/tkde/tkde17.html#WangHLT05

[13] U. Yun, "Mining lossless closed frequent patterns with weight constraints." *Knowl.-Based Syst.*, vol. 20, no. 1, pp. 86–97, 2007. [Online]. Available: http://dblp.uni-trier.de/db/journals/kbs/kbs20.html#Yun07

[14] M. J. Zaki and C.-J. Hsiao, "Charm: An efficient algorithm for closed itemset mining." in *SDM*, R. L. Grossman, J. Han, V. Kumar, H. Mannila, and R. Motwani, Eds. SIAM, 2002. [Online]. Available: http://dblp.uni-trier.de/db/conf/sdm/sdm2002.html#ZakiH02

[15] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," *SIGMOD Rec.*, vol. 22, no. 2, pp. 207–216, Jun. 1993. [Online]. Available: http://doi.acm.org/10.1145/170036.170072

[16] P.-Y. Hsu, Y.-L. Chen, and C.-C. Ling, "Algorithms for mining association rules in bag databases." *Inf. Sci.*, vol. 166, no. 1-4, pp. 31–47, 2004. [Online]. Available: http://dblp.uni-trier.de/db/journals/isci/isci166.html#HsuCL04

[17] Y. Li, S. Zhu, X. S. Wang, and S. Jajodia, "Looking into the seeds of time: Discovering temporal patterns in large transaction sets," 2002.

[18] B. Liu, W. Hsu, and Y. Ma, "Mining association rules with multiple minimum supports," in *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '99. New York, NY, USA: ACM, 1999, pp. 337–341. [Online]. Available: http://doi.acm.org/10.1145/312129.312274

[19] C.-Y. Wang, S.-S. Tseng, and T.-P. Hong, "Flexible online association rule mining based on multidimensional pattern relations." *Inf. Sci.*, vol. 176, no. 12, pp. 1752–1780, 2006. [Online]. Available: http://dblp.uni-trier.de/db/journals/isci/isci176.html#WangTH06

[20] J. Liu, Y. Pan, K. Wang, and J. Han, "Mining frequent item sets by opportunistic projection." in *KDD*. ACM, 2002, pp. 229–238. [Online]. Available: http://dblp.uni-trier.de/db/conf/kdd/kdd2002.html#LiuPWH02

[21] U. Yun, "A new framework for detecting weighted sequential patterns in large sequence databases," *Know.-Based Syst.*, vol. 21, no. 2, pp. 110–122, Mar. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.knosys.2007.04.002

[22] "Unil yun, "wis: Weighted interesting sequential pattern mining with a similar level of support and/or weight," etri journal, vol. 29, no. 3, jun. 2007, pp. 336-352. http://dx.doi.org/10.4218/etrij.07.0106.0067."

[23] F. Bonchi and C. Lucchese, "Pushing tougher constraints in frequent pattern mining." in *PAKDD*, ser. Lecture Notes in Computer Science, T. B. Ho, D. W.-L. Cheung, and H. Liu, Eds., vol. 3518. Springer, 2005, pp. 114–124. [Online]. Available: http://dblp.uni-trier.de/db/conf/pakdd/pakdd2005.html#BonchiL05

[24] C. Bucil, J. Gehrke, D. Kifer, and W. White, "Dualminer: A dual-pruning algorithm for itemsets with constraints," *Data Mining and Knowledge Discovery*, vol. 7, no. 3, pp. 241–272, july 2003. [Online]. Available: http://dx.doi.org/10.1023/A:1024076020895

[25] K. Gade, J. Wang, and G. Karypis, "Efficient closed pattern mining in the presence of tough block constraints." in *KDD*, W. Kim, R. Kohavi, J. Gehrke, and W. DuMouchel, Eds. ACM, 2004, pp. 138–147. [Online]. Available: http://dblp.uni-trier.de/db/conf/kdd/kdd2004.html#GadeWK04

[26] A. J. T. Lee, W. chuen Lin, and C. sheng Wang, "Mining association rules with multi-dimensional constraints." *Journal of Systems and Software*, vol. 79, no. 1, pp. 79–92, 2006. [Online]. Available: http://dblp.uni-trier.de/db/journals/jss/jss79.html#LeeLW06

[27] Y. ling Cheung and A. W. chee Fu, "Mining frequent itemsets without support threshold: with and without item constraints," *In IEEE Trans. on Knowledge and Data Engineering*, vol. 16, pp. 1069–2004, 2004.

[28] Z. Zhao, D. Yan, and W. Ng, "Mining probabilistically frequent sequential patterns in uncertain databases," in *Proceedings of the 15th International Conference on*

Extending Database Technology, ser. EDBT '12. New York, NY, USA: ACM, 2012, pp. 74–85. [Online]. Available: http://doi.acm.org/10.1145/2247596.2247606

[29] C. K.-S. Leung, M. A. F. Mateo, and D. A. Brajczuk, "A tree-based approach for frequent pattern mining from uncertain data," in *PAKDD*, 2008, pp. 653–661.

[30] C. C. Aggarwal, Y. Li, J. Wang, and J. Wang, "Frequent pattern mining with uncertain data," in *KDD*, 2009, pp. 29–38.

[31] C. K.-S. Leung and S. K. Tanbeer, "Fast tree-based mining of frequent itemsets from uncertain data," in *DASFAA (1)*, 2012, pp. 272–287.

[32] ——, "Puf-tree: A compact tree structure for frequent pattern mining of uncertain data." in *PAKDD (1)*, ser. Lecture Notes in Computer Science, J. Pei, V. S. Tseng, L. Cao, H. Motoda, and G. Xu, Eds., vol. 7818. Springer, 2013, pp. 13–25. [Online]. Available: http://dblp.uni-trier.de/db/conf/pakdd/pakdd2013-1.html#LeungT13

[33] U. Yun, "Efficient mining of weighted interesting patterns with a strong weight and/or support affinity." *Inf. Sci.*, vol. 177, no. 17, pp. 3477–3499, 2007. [Online]. Available: http://dblp.uni-trier.de/db/journals/isci/isci177.html#Yun07

[34] H. Xiong, P.-N. Tan, and V. Kumar, "Mining strong affinity association patterns in data sets with skewed support distribution." in *ICDM*. IEEE Computer Society, 2003, pp. 387–394. [Online]. Available: http://dblp.uni-trier.de/db/conf/icdm/icdm2003.html#XiongTK03

[35] C. Gyrdi and R. Gyrdi, "A comparative study of association rules mining algorithms," Proc. of CONTI 2002, 2002.

[36] O. Benjelloun, A. D. Sarma, A. Y. Halevy, and J. Widom, "Uldbs: Databases with uncertainty and lineage." in *VLDB*, 2006, pp. 953–964.

[37] P. Agrawal, O. Benjelloun, A. D. Sarma, C. Hayworth, S. Nabar, T. Sugihara, and J. Widom, "Trio: A system for data, uncertainty, and lineage," in *Proceedings of the 32Nd International Conference on Very Large Data Bases*, ser. VLDB '06. VLDB Endowment, 2006, pp. 1151–1154. [Online]. Available: http://dl.acm.org/citation.cfm?id=1182635.1164231

[38] S. Ganguli and A. Nerode, "Effective completeness theorems for modal logic," *Annals of Pure and Applied Logic*, vol. 128, no. 13, pp. 141 – 195, 2004. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168007203001349

[39] L. Antova, T. Jansen, C. Koch, and D. Olteanu, "Fast and simple relational processing of uncertain data." in *ICDE*, G. Alonso, J. A. Blakeley,

and A. L. P. Chen, Eds. IEEE, 2008, pp. 983–992. [Online]. Available: http://dblp.uni-trier.de/db/conf/icde/icde2008.html#AntovaJKO08

[40] Y. Xia, Y. Yang, and Y. Chi, "Mining association rules with non-uniform privacy concerns." in *DMKD*, G. Das, B. L. 0001, and P. S. Yu, Eds. ACM, 2004, pp. 27–34. [Online]. Available: http://dblp.uni-trier.de/db/conf/dmkd/dmkd2004.html#XiaYC04

[41] C. K. Chui, B. Kao, and E. Hung, "Mining frequent itemsets from uncertain data." in *PAKDD*, ser. Lecture Notes in Computer Science, Z.-H. Zhou, H. Li, and Q. Y. 0001, Eds., vol. 4426. Springer, 2007, pp. 47–58. [Online]. Available: http://dblp.uni-trier.de/db/conf/pakdd/pakdd2007.html#ChuiKH07

[42] C.-K. Chui and B. Kao, "A decremental approach for mining frequent itemsets from uncertain data," in *Proceedings of the 12th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, ser. PAKDD'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 64–75. [Online]. Available: http://dl.acm.org/citation.cfm?id=1786574.1786585

[43] C. K.-S. Leung, C. L. Carmichael, and B. Hao, "Efficient mining of frequent patterns from uncertain data," in *ICDM Workshops*, 2007, pp. 489–494.

[44] C. K. Chui, B. Kao, and E. Hung, "Mining frequent itemsets from uncertain data," in *PAKDD*, 2007, pp. 47–58.

[45] C. C. Aggarwal, Y. Li, J. Wang, and J. Wang, "Frequent pattern mining with uncertain data," in *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, ser. KDD '09. New York, NY, USA: ACM, 2009, pp. 29–38. [Online]. Available: http://doi.acm.org/10.1145/1557019.1557030

[46] Y. Tong, L. Chen, Y. Cheng, and P. S. Yu, "Mining frequent itemsets over uncertain databases," *CoRR*, vol. abs/1208.0292, 2012. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1208.html#abs-1208-0292

[47] H. Xiong, P.-N. Tan, and V. Kumar, "Mining strong affinity association patterns in data sets with skewed support," in *In Proceedings of the 3rd IEEE International Conference on Data Mining*, 2003, pp. 387–394.

[48] E. R. Omiecinski, "Alternative interest measures for mining associations in databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 15, no. 1, pp. 57–69, 2003. [Online]. Available: http://www.computer.org/portal/web/csdl/doi/10.1109/TKDE.2003.1161582

[49] http://fimi.ua.ac.be/data/. (2013) Frequent itemset mining dataset repository @ONLINE. [Online]. Available: http://fimi.ua.ac.be/data/

[50] M. Li and Y. Liu, "Underground coal mine monitoring with wireless sensor networks." *TOSN*, vol. 5, no. 2, 2009. [Online]. Available: http://dblp.uni-trier.de/db/journals/tosn/tosn5.html#LiL09

[51] L. C. 0002 and R. T. Ng, "On the marriage of lp-norms and edit distance." in *VLDB*, M. A. Nascimento, M. T. zsu, D. Kossmann, R. J. Miller, J. A. Blakeley, and K. B. Schiefer, Eds. Morgan Kaufmann, 2004, pp. 792–803. [Online]. Available: http://dblp.uni-trier.de/db/conf/vldb/vldb2004.html#ChenN04