Intra-Transaction Frequency Based Weighted Interesting Pattern Mining

Department of Computer Science and Engineering

East West University

Dhaka-1212, Bangladesh

September 2019

Declaration

I, hereby, declare that the work presented in this thesis is the outcome of the investigation performed by me under the supervision of the name of your supervisor, Professor, Department of Computer Science and Engineering, East West University. I also declare that no part of this thesis/project has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

.....

(Jesan Ahmed Ovi)
Supervisor

Signature

.....

(Ramisa Hassan)
2015-1-60-148
Signature

.....

(Mahmudul Hasan)
2015-1-60-155
Signature

.....

(Rupa Akter)
2015-1-60-108
Signature

Letter of Acceptance

This thesis report entitled 'Intra-Transaction Frequency Based Weighted Interesting Pattern Mining' submitted by Ramisa Hassan (ID: 2015-1-60-148), Mahmudul Hasan (ID: 2015-1-60-155), Rupa Akter (ID: 2015-1-60-108) to the Department of Computer Science and Engineering, East West University is accepted by the department in partial fulfillment of requirements for the award of the Degree of Bachelor of Science and Engineering on September, 2019.

. . . . . .

Dr. Taskeed Jabid

Associate Professor and Chairperson

Department of Computer Science and Engineering

East West University

Aftabnagar, Dhaka-1212, Bangladesh

…….....

Jesan Ahmmed Ovi

Lecturer and Supervisor

Abstract

Data mining is an integral part of Knowledge Discovery in Database (KDD), which is the overall process of converting raw data into useful information. This process consists of a series of transformation steps from data preprocessing to post-processing of data mining results. Many algorithms have developed considering support affinities like FP-growth and Apriori Growth. However, only considering support affinity sometimes does not give any proper knowledge as many of the essential items pruned due to having low support affinity. In our thesis work, we are going to propose a new tree-based approach, WIP-growth approach, which considers weight along with intra-transaction frequency. In this method, low-frequency items cannot get pruned due to having a high weight value.

Acknowledgment

As it is true for everyone, we have also arrived at this point in achieving a goal in our life through various interactions with and help from other people. However, written words are often elusive and harbor diverse interpretations even in one's mother language. Therefore, we would not like to make efforts to find the best words to express my thankfulness other than simply listing those people who have contributed to this thesis itself in an essential way. This work was carried out in the Department of Computer Science and Engineering at East West University, Bangladesh.

First of all, we would like to express my deepest gratitude to the Almighty for His blessings on us. Next, our special thanks go to our supervisor, **Jesan Ahmmed Ovi**, who gave us this opportunity, initiated us into the field of Machine Learning, and without whom this work would not have been possible. His encouragement, visionaries and thoughtful comments and suggestions, unforgettable support at every stage of our B.Sc. study were simply appreciating and essential. His ability to muddle us enough to finally answer our own question correctly is something valuable that we have learned and we would try to emulate if ever we get the opportunity.

Last but not least, we would like to thank our parents for their unending support, encouragement, and prayers.

There are numerous other people too who have shown me their constant support and friendship in various ways, directly or indirectly related to our academic life. We will remember them in our hearts and hope to find a more appropriate place to acknowledge them in the future.

Ramisa Hassan

September, 2019

Mahmudul Hasan

September, 2019

Rupa Akter

September, 2019

Table of Contents

List of Figures

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In every sphere of human life, knowledge has played a significant role. If someone wants to acquire knowledge, find the hidden information from a vast amount of data in various formats, data analysis is fundamental. However, extracting useful information from a vast amount of data is extremely challenging. Data mining supports all kinds of knowledge and information gaining process. The term "data mining" was first introduced in 1990. Data mining refers to the process that extracts new knowledge from large amounts of data. Data mining has several applications, and these various types of applications have enhanced the various fields such as (business, education, social, media, medical) of human life.

Data Mining also refers to a process of finding patterns and discovering correlations in an extensive data set from a different perspective. It aims to extract information when the data is too large. In a nutshell, the actual task of data mining is to analyze large amounts of data to extract knowledge and discover the previously unknown and interesting patterns. Machine Learning and Statistical Models are also used in data mining to uncover hidden patterns of data. Earlier data mining recognized as a sub process within a significant process known as knowledge discovery in database(KDD). However, with the increasing complexity of data sets, it has lead data mining to evolve from static data to dynamic data. Data mining includes many techniques among them; the research focuses on pattern mining. Pattern mining aims to identify rules that discover specific patterns within the data. An essential use of pattern mining is to discover sequential patterns.

Moreover, Market-basket analysis, which identifies items that typically occur together in purchase transactions, was one of the first applications of data mining. It helps to know the customer behavior and exploring unknown credible patterns that is useful for business success. There are many purposes

of data mining like detecting trends, predicting various outcomes, modeling, target audience, gathering information about the product or service used.

In this chapter, we will discuss different types of data mining tools, algorithms, and their applications in real life. We also try to give some idea about our proposed system and its importance in real life.

1.1 Overview

Knowledge discovery in databases refers to the overall process of discovering useful knowledge from data, and data mining refers to a particular step in this process. Data mining is the application of specific algorithms for extracting patterns from data. As a result, demand for efficient algorithms for mining different types of the database has also increased. Over the recent decades, many efficient frequent pattern methods were discovered in various fields like frequent pattern mining, high utility pattern mining for incremental databases, weighted interesting pattern mining. Most of those algorithms based on support constrain to prune the item from a dataset and consider the only present or absent of item in a transaction. Some of that algorithm also consider the actual amount of an item in the transaction. However, there is some situation in real life where the presence of an item in particular transaction is not absolute. That means there is some probability of the item being present in the transaction.

For this reason, we need something special to get rid of this problem. However, our target is to keep essential items of a transaction although having a low frequency.

1.2 Motivation

In recent days, frequent pattern mining, including incremental mining, has been the subject for numerous research studies. However, most of the incremental mining algorithm is Apriori based, which are not readily adaptable for the case of FP- tree based frequent pattern mining. CanTree (Canonical Order Tree) has proposed for solving the problem. This CanTree captures the content of the transactional database and orders tree nodes according to some canonical order. The CanTree can be easily maintained when the database of the transaction faces any modification such as insertion or deletion.

However, sometimes, frequent patterns may not be enough to find meaningful and interesting patterns from an extensive database. Cause it only reflects the number of transactions that contain that pattern. In many cases, different items may be different importance (weight) depending on the application. There are so many real-life situations where it is not realistic to generate just frequent patterns rather than find only those that are important to those applications.

For this reason, researchers introduce an efficient mining algorithm which is Weighted Interesting Pattern Mining (WIP). This algorithm is based on mining weighted frequent patterns. In the WIP algorithm, a new measure has introduced called weight-confidence(w-confidence). WIP finds useful patterns with weight and support affinity patterns.

1.3 Problems with Existing System

As far as we know, we are the first to propose an algorithm of mining weighted interesting pattern mining using intra-transaction frequency. As mentioned earlier, some system can mine patterns using only support affinity; some uses considerable weight. However, still due to having a low-frequency valuable item become pruned from the transaction database. Our goal is to mitigate those flaws of an existing system by giving universal methods that able to find interesting patterns from using intra-transaction frequency.

1.4 Challenges

Procedures for mining weighted interesting patterns considering only strong weight and support affinity already discovered. So, in our thesis work, we are trying to find interesting patterns considering the weight and intra-transaction frequency. We faced several significant challenges to do so. In this section, we discuss those problems shortly.

- Algorithms that deal with the weighted interesting itemsets such as WFIM (weighted frequent itemset mining) and WIP (weighted interesting pattern) maintain the downward closure property by multiplying each itemset's frequency by the maximum weight.

- Some algorithms, such as high utility pattern mining, solved the problem of downward closure property, by using MEU (mining with expected utility).
- Finally, an algorithm that works with incremental mining considers some canonical ordering but we can't as we have to consider the weight of the item. For this reason, we use the lexicographical ordering.

1.5 Our Contribution

Our primary target is to propose an efficient mining method to find an interesting weighted pattern from a transaction using intra-transaction frequency. If we look up some previous mining techniques, then we can see that only frequent patterns do not help to find actual interesting patterns. Some applications demand rare patterns that are may not so frequent but more interesting for that purpose. Our target is to find those interesting patterns.

1.6 Organization of This Book

We organize the rest of our thesis book like this, in Chapter 2, a detailed description of related algorithms. Our proposed tree construction process and mining process are explained in Chapter 3. In Chapter 4, the experimental results of our methods with a brief explanation. Finally, in Chapter 5, we conclude our thesis works.

Chapter 2

Related Work

From the very beginning of frequent pattern mining algorithms, researchers are trying to improve algorithms that can be more efficient. As it is an essential part of data analysis and data mining field, constant developments are required to tackle every situation in this era of storing and producing massive data.

In this chapter, we are going to discuss related topics and different types of procedures that help us during the implementation of our work.

2.1 Background

In this chapter, we will discuss the related topics, different types of procedures that help us to implement our work. We also focus on the essential notions, background topics of our work. We will start this chapter by explaining the essential terminologies and different types of parameters related to our work.

2.2 What is Transactional Database?

In general, a transactional database consists of a file where each record represents a transaction. A transaction typically carries a unique identification number (trans ID) and a list of the items making up the transaction (such as items purchased in a store). A transactional database may have additional tables, which contain other information related to the transaction, such as item description.

TID	Items
1	Bread, Milk
2	Bread, Sugar, Butter, Egg
3	Milk, Sugar, Butter, Cheese
4	Bread, Milk, Butter, Egg
5	Bread, Milk, Cheese, Sugar

Table 2.1: Transactional Database

2.3 What is Frequent Pattern Mining?

Frequent patterns are patterns that frequently occur in data. There are many kinds of frequent patterns, including frequent itemsets, frequent subsequences (also known as sequential patterns), and frequent substructures. A frequent itemset typically refers to a set of items that often appear together in a transactional database. Suppose, milk and bread, which are frequently bought together in grocery stores.

Frequent Pattern Mining (also known as Association Rule Mining) is an analytical process that finds frequent patterns, associations, or causal structures from data sets found in various kinds of databases such as relational databases, transactional databases. If there is a given set of transactions, this process aims to find the rules that enable us to predict the occurrence of a specific item based on the occurrence of other items in the transaction.

This method of analysis can be useful in evaluating data for various business functions and industries and is useful in determining the frequent patterns in buying behavior for various products and services, and in analyzing the relationships among various data points and to understand targeted audiences.

2.4 Different types of Frequent Pattern Mining Algorithm

Different types of algorithms can be used for frequent mining patterns. A short description of the algorithms is describing below.

2.4.1 Apriori Algorithm

R. Agrawal and R. Srikant give Apriori algorithm in 1994 for finding frequent itemsets in a dataset for the Boolean association rule. The name of the algorithm is Apriori because it uses prior knowledge of frequent itemset properties. The fundamental concept of the Apriori algorithm is its anti-monotonicity of support measures. Apriori assumes that all subsets of a frequent itemset must be frequent. If an item set is infrequent, then all its supersets will be infrequent.

Apriori uses a “bottom-up" approach, where various subsets are extended one item at a time (it is a step known as candidate generation), and groups of candidates prepared for testing against the data. The algorithm terminates when no further successful extensions found. Apriori uses “breadth-first search” and a “hash tree” structure to count candidate itemsets efficiently.

In many cases, the Apriori algorithm reduces the size of candidate itemsets significantly and provides an excellent performance gain. However, there are still some limitations. One of the essential limitations of this algorithm is that the entire database goes under several scanning repeatedly and a vast set of candidate items are required to be verified using the technique of pattern matching.

2.4.2 Frequent Pattern Growth Algorithm

Han proposes a frequent Pattern (FP) Growth algorithm. It is an improvement of the apriori algorithm. It uses for finding frequent itemset in a transaction database without any candidate generation.

It implements a divide-and-conquer technique to compress the numerous components into an FP-Tree that recollects the association information of the various items. The FP-Tree is further divided into a set of Conditional FP-Trees for each frequent item for mining them distinctly.

FP growth algorithm solves the problem of identifying long patterns by searching through smaller conditional FP- tree repeatedly. However, constructing the FP- Tree is time-consuming if the data set is extensive.

2.5 Interesting Pattern

Some researchers define an interesting pattern as a pattern that frequently appears in any database. In the field of data mining, so many traditional algorithms are developed to find frequent patterns. However, these traditional algorithms, researchers did not consider the weight of the item. As a result, some patterns may not be interesting to the users. Consequently, there are so many items that may not be frequent but much more attractive to the user. It is especially true when mining at a low support threshold or mining for long patterns.

For example, if we consider a transactional database that contains items like gold, silver, platinum. Here gold may not be frequent because it is the most expensive product. So if we are looking for frequent items gold may be pruned, but it is more interesting as it the most profitable product which implies that, if we equally treat every product there is a chance that interesting patterns may be excluded and so many uninteresting patterns may be generated.

So we can say that a particular item may not be frequent, but when it occurs with a high weighted item, it may become frequent.

2.6 Important Terminologies

In this section, we are going to explain some essential ideas that are related to our topic.
Let us consider a transactional database table 2.1. This table will help us explaining our terminologies.

Item	Price	Support	Weight(Normalized)
Q	600	800	0.6
R	700	750	0.4
S	500	700	0.07

Table 2.2: Transactional Database with Support and Weight

2.6.1 Weight of an Item

Weight of an item is a non-negative real number that is assigned to reflect the importance of each item in the transactional database.

Given a set of items, then the weight of a pattern can be defined as follows,

$$\text{Weight(P)} = \frac{\sum_{i=1}^{length(P)} Weight(x_i)}{length(P)}$$

For example, from table 2.2, the weight of pattern QRS is (0.6+0.4+0.07)/3 =0.356

2.6.2 Weighted Support of a Pattern

Weighted support of a pattern can be defined as the resultant value of multiplying the pattern's support with the weight of the pattern. That is, given a pattern P, the weighted support is

$$WSupport(P) = Weight(P) * Support(P)$$

For example, from the given table 2.2,

$$WSupport(Q) = Weight(Q) * Support(Q)$$

$$= 0.6 * 800$$

$$= 480$$

2.6.3 Weighted Frequent Pattern

A pattern is called weighted frequent pattern if the weighted support of the pattern is not less than a minimum threshold.

If we consider 400 as the minimum support threshold, then S cannot be frequent because of WSupport(S) = 700*0.07=49, which is very much less than the minimum support threshold.

2.7 Some Important Data Mining Techniques and Procedures

In our previous section (section 2.3), we tried to give a little hint on the differences between previously build methodologies and our proposed methodology. However, it is essential to discuss the already build in methodologies that encourage us. In this particular section, we try to discuss the methodologies widely.

2.7.1 Incremental and Iterative Mining

Numerous researches have been done for developing the techniques for incremental and interactive mining approaches in the field of traditional frequent pattern mining methodologies. CanTree. CP-tree, FUFP-tree, is the resultant prefix tree structure of the research.

After that, some efficient dynamic database updating algorithms in short EDUA designed for mining the databases when data deletion is performed quietly frequently in any database subset.

IncWTP and WssWTP are the algorithms designed for mining incremental and interactive Web traversal patterns. However, these algorithms do not apply to incremental and interactive high utility pattern mining. Because, in high utility pattern mining, any deletion or modification cannot be done. Moreover, none of the data structures have the "build once mine many" property.

2.7.2 CanTree (Canonical Order Tree)

CanTree (canonical order tree) introduced by Carson Kai-Sang Leung, Quamrul I. Khan, Zhan Li, Tariqul Hoque for incremental frequent pattern mining. In this tree, items arranged according to canonical order. In the case of the construction of the CanTree, it requires only one database scan. This quality differs CanTree from FP-Tree, as two database scans required for FP-Tree. Notably, items in CanTree can arrange in lexicographic order or alphabetical order.

CanTree solves the problems of the FELINE or AFPIM algorithms. If we compare CanTree with the CATS tree of a FELINE algorithm, then we will see that the CATS tree keeps items in local frequency order which is quite costly and challenging for the corresponding FELINE algorithm to locate transactions. Whereas CanTree keeps items in canonical order. So locate transactions is quite easy.

Again if we compare CanTree with the AFPIM algorithm, then we will see that the AFPIM algorithm arranges items according to descending (global) frequency. If the frequency order of items gets changed, then one needs to adjust the tree node using the bubble sort algorithm. On the other hand, the CanTree is unaffected by any changes in frequency due to deletions or insertions of transactions.

2.7.2.1 Construction of CanTree

CanTree is designed mainly for the incremental mining approach. The tree captured the content of the database and arranged those items according to some canonical order. It is important to note that construction of the CanTree requires only one database scan. This quality defers it from the construction of an FP-Tree. Because, in FP-Tree, two database scans required for the construction (one scan for obtaining item frequencies and another one for arranging items in descending frequency order). Items in CanTree can arrange in lexicographic order or alphabetical order. While the orderings are frequency-independent, items can also arrange according to some fixed frequency-related ordering.

Now let us consider a transactional database below:

	TID	Contents
Original Database(DB)	t1	a, d, b, g, e, c
	t2	d, f, b, a, e
	t3	a
	t4	d, a, b
The First Group of Insertions(db1)	t5	a, c, b
	t6	c, b, a, e
The Second Group of Insertions(db2)	t7	a, b, c
	t8	a, b, c

Table 2.3: Transactional Database

In the above table, items arranged in lexicographical order. Items can also arrange according to fixed frequency-related ordering, and it can be either ascending or descending. In the above table' original database (DB)' is in descending order. It is important to note that, once the order is determined, the rest of the items must follow that ordering.

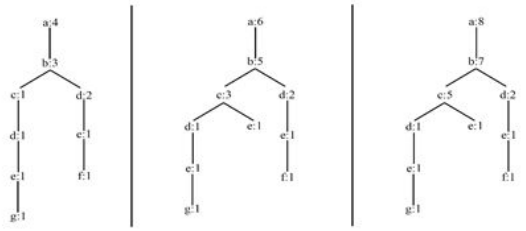

Figure 2.1: the CanTree after each transaction is added

The above figure reflects CanTree, how it looks after adding each transaction in the tree. Transactions t1–t4 can easily add to the tree. As canonical order is unaffected by the frequency order of items at runtime, any changes in frequency caused by incremental updates do not affect the ordering of items in the CanTree. As well as, the swapping of tree nodes, which often leads to merging and splitting of tree nodes, is not needed. Once the CanTree is constructed, one can mine frequent patterns from the tree in a divide-and-conquer manner which is similar to the FP growth algorithm.

2.7.3 Weighted Frequent Pattern Mining

2.7.3.1 Anti Monotone Property

If a pattern is defined as an infrequent pattern, then all super patterns of the pattern must be infrequent.

Weighted interesting pattern mining or weighted frequent pattern mining is used to mine correlated patterns with weight or support affinity. The weight of a pattern P is the ratio of the sum of all its items weight value and the length of P. Though considering weight affinity during mining is a new concern for the researchers, but still, much research has been done on this topic.

Researchers showed that the main challenge for both Weighted Frequent Itemset Mining(WFIM) and Weighted Interesting Pattern Mining(WIP) is that the weighted frequency of a pattern does not follow the 'downward closure' property.

Let us consider two items 'r' and 's,' and their weight is 0.6 and 0.2 respectively. Moreover, their frequency is 4 and 5, respectively. Also, the itemset 'rs' has frequency 3. Then the weight of itemset 'rs' will be (0.6+0.2)/2=0.4, and weighted frequency is 0.4*3=1.2. Then the weighted frequency of 'r' and 's' will be 0.6*4=2.4 and 0.2*5=1.0, respectively. If we consider 1.2 as the minimum weighted frequency threshold, the pattern's' is weighted infrequent, but its sub-pattern is weighted frequent, which proves that the downward closure property is not satisfied here.

If both the WFIM and WIF maintain the 'downward closure' property, then the scenario will be different from the above. For maintaining the property, we will multiply the item set's frequency with the global maximum weight. From the above example, we see that item 'r' has the maximum weight of frequency 0.6. Now if we multiply it with the frequency of 's' which is 5, then the resulting value will be 3.0. Then at the first stage, the pattern's' will not be pruned, and pattern 'rs' will not miss. However, the pattern's' is overestimated, which will be pruned later by using actual frequency.
Anti-monotone property is another primary concern in the case of weighted frequent pattern mining. Because after inserting different weights to the items and the property break down, which causes weighted frequent to the super pattern of an infrequent weighted pattern.

Mining Association Rules with Weighted Items(MINWAL) is designed to maintain the anti-monotone property with the help of K-support bound. MINWAL defined as weighted support that calculates by multiplying support of a pattern with an average weight of a pattern. For maintaining the anti-monotone property, any super pattern of the infrequent weighted pattern must be weighted infrequent.

2.8 Conclusion

Some essential data-mining tools, techniques, procedures had discussed throughout this chapter which has a tremendous impact on different sectors in data-mining for example, medical database, stream data such as sensor network data. These methods and procedures not only significantly change the data mining sector but also help us to think about new approaches, search for new sectors and their applications in real life. With the help of the knowledge of this chapter in the next chapter, we will discuss how we find new applications, new sector and their solutions in the field of data-mining.

Chapter 3

Proposed Model

In our thesis work, we basically incorporate important and most demanding topics of data-mining those are Weighted patterns for building a new methodology focusing on the recent demand of users. Our target is simple but important and useful, is to introduce some techniques that help to perfectly capture data stream where data items are uncertain and also propose mining techniques that ensure only find interesting patterns depending on users need. There are many recent applications in where data is uncertain for example medical database and also data sets are growing faster in size, can come extremely fast, as a result, we can't afford to capture whole the data stream and mine them by scanning several time like some traditional data-mining techniques. Moreover, as the datasets are huge it is maybe unnecessary to generate all the frequent patterns which may also be huge in size without considering the applications and users need. It is being more effective and less time consuming to generate only those patterns that actually have an impact on the applications. Several researchers proposed many approaches to find interesting patterns from databases and also from data streams. But their works only deal with certain data. As our prior knowledge we are the first to deal with uncertain data streams to generate only interesting patterns which are less time consuming, appropriate for large dataset and also use less memory resources.

3.1 Our Goal

In our thesis, we propose a new approach to mine frequent interesting items considering external weight. We named our approach Weighted Interesting Patterns Mining (WIPM)-growth approach. In the later sections, we will explain our WIPM-growth approach with proper examples.

3.2 Important Terminologies

In this section, we are going to introduce some concepts which have connections to our work. For this, we need to use a couple of tables that are given below.

Transactions	items
T1	a(5), b(3), c(7)
T2	b(7), c(2), d(2), e(3)
T3	a(3), e(2)
T4	b(2), d(1), e(3)
T5	c(3), d(5), e(2)

Table 3.1 Transactional Database

Table 3.1 represents a transactional database table that contains transaction ID and items with intra-transaction frequency. Internal frequency varies from transactions to transactions. We consider each value randomly for our tree building purpose.

Item	External Weight
a	3
b	5
c	2
d	1
e	2

Table 3.2 Items with Internal Weight

Table 3.2 represents the external weight of items. We also here consider external weights randomly for tree building purposes.

3.2.1 Intra-Transaction Frequency

The total number of frequency of a particular item in a given set of transactions. For example, in our given transactional database item frequency of ‘a’ is 5 in T1 and 3 inT3. So the intra-transaction frequency of ‘a’ is 5+3=8.

3.2.2 Combined Weighted Support

We can calculate CWS by multiplying the sum of intra-transaction frequency with external weight. For example, the intra-transaction frequency of ‘a’ is 8 and the external weight of ‘a’ is 3. So CWS for ‘a’ is 8*3=24.

Suppose, using tables 3.1 and 3.2, we want to calculate the projected database for pattern ‘cde’ then the calculating equation will be,

$$\text{cde}= \sum \frac{liength\ (c)*weight\ (c)+length\ (de)*weight\ (de)}{total\ length}$$

3.3 WIPM Growth Approach

Like most traditional data-mining procedures WIPM-growth approach consists of two major phases one is tree construction and another one is mining interesting patterns.

In our next section, we are going to discuss the construction process of our proposed tree.

3.3.1 Tree Construction

For constructing our proposed WIP tree, we use both the information of tables 3.1 and 3.2.

Step 1:

In the very beginning step, the first item of transaction T1 will be inserted into the tree.

Step 1.1:

The first item of T1 is ‘a’. Intra-transaction frequency and external weight of a is 5 and 3 respectively. So the value of ‘a’ will be 15 which is the multiplication of intra-transaction frequency (5) and external weight (3).

Figure 3.1: step 1_insertion of T1 items

In the beginning, ‘a’ is the only node that going to connect directly with the root. Figure 3.1 shows the initial stage after inserting ‘a’ to root.

Step 1.2:

Now, the second item of transaction T1 which is ‘b’ will be inserted into the tree.

Figure 3.2: step 2_insertion of T1 items

The value of item ‘b’ is 15 and it is calculated just like item ‘a’ that is the multiplication of intra-transaction frequency of b (3) and external weight of b (5). And ‘1’ is added in the bracket after the value of item ‘b’. This ‘1’ indicates the first child of item ‘a’.

Step 1.3:

After the addition of the first two items of transaction T1 into the tree, now er are going to add the last value of T1 which is ‘c’. Figure 3.3 represents the tree condition after adding T1 into the tree.

Figure 3.3: step 3_insertion of T1 items

The value of item ‘c’ is 14 and it is calculated just like item ‘a’ and ‘b that is the multiplication of intra-transaction frequency of c (7) and external weight of c (2). And again ‘1’ is added in the bracket after the value of item ‘c’. This ‘1’ indicates the first child of item ‘a’.

Step 2:

Now we are in the second phase of the tree construction process. In this phase, items of transaction T2 will be added into the tree. Transaction T2 has four items and it starts with item ‘b’. So item ‘b’ will connect with the root first and the rest of the items will be added gradually. And each of the values will be the first child of item ‘b’.

Step 2.1:

‘b’ is the first item of T2. So just like item ‘a’ of T1, it connects directly with the root.

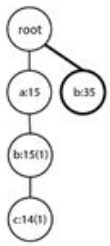

Figure 3.4: step 4_insertion of T2 items

35 is the value of item ‘b’ and it is calculated by multiplying intra-transaction frequency of b (7) and external weight of b (5).

Step 2.2:

The second item of T2 is ‘c’. It goes after by item ‘b’ and it is the first child of ‘b’. As per rule, 1 will add in the bracket after ‘c:4’.

Figure 3.5: step 5_insertion of T2 items

4 is the value of item ‘c’ and it is calculated by multiplying intra-transaction frequency of c (2) and external weight of c (2).

Step 2.3:

After insertion of the second item, now the third item ‘d’ of T2 is going to be added in the tree.

Similarly, it goes after by items ‘b’ and ‘c’. Unlike ‘c’, it is also the first child of ‘b’. So, 1 will add in the bracket after ‘d:2’.

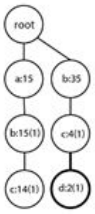

Figure 3.6: step 5_insertion of T2 items

The value of item ‘d’ is 2 and calculated by multiplying intra-transaction frequency of d (2) and external weight of d (1).

Step 2.4:

Finally, the last item of T2 which is ‘e’ will be added with the node ‘b’ of our tree. Serially it added after ‘b’, ‘c’, and ‘d’.

The value of item ‘b’ is 4 and calculated by multiplying intra-transaction frequency of d (4) and external weight of d (1).

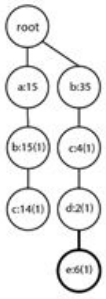

Figure 3.7: step 5_insertion of T2 items

The value of item ‘e’ is 4 and calculated by multiplying intra-transaction frequency of e (3) and external weight of e (2).

Step 3:

Now we are in the third phase of the tree construction process. In this phase, items of transaction T3 will be added into the tree. Transaction T3 has two items and it starts with item ‘a’. As ‘a’ is already added in the tree so we will combine the second item of that transaction with the existing node ‘a’. And also the value of ‘a’ in this transaction which is 9 (intra-transaction frequency of a (3) and external weight of a (3)) placed just after the previous value 15.

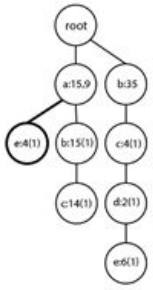

Figure 3.8: step 6_insertion of T3 items

The value of item ‘e’ is 4 and calculated by multiplying intra-transaction frequency of e (2) and external weight of e (2). As ‘a’ has already a child before and now another child id added on the tree so ‘e’ becomes the second child of ‘a’ and (2) is added to indicate a number of children.

Step 4:

In this step, we start our fourth phase of tree construction process. Transaction T4 has three items ‘b’, ‘d’ and ‘e’. The process that has been completed in the third phase will be repeated here. As ‘b’ is already presented in the tree so no need to add it again. We just merge them. And also value of ‘b’ in this transaction which is 10 (intra-transaction frequency of b (2) and external weight of b (5)) placed just after the previous value 35.

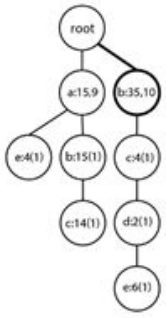

Figure 3.9: step 7_insertion of T4 items

Step 4.1

The second item of T4 is ‘d’. It goes after by item ‘b’ and it is the second child of ‘b’. As per rule, 2 will add in the bracket after ‘d:1’.

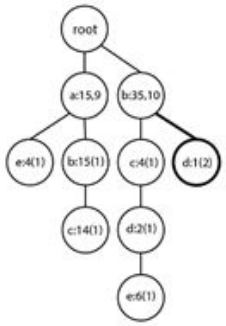

Figure 3.10: step 8_insertion of T4 items

The value of item ‘d’ is 1 and calculated by multiplying intra-transaction frequency of d (1) and external weight of d (1). As ‘b’ has already a child before and now another child is added on the tree so ‘d’ becomes the second child of ‘b’ and (2) is added to indicate the number of children.

Step 4.2

The last item of T4 is ‘e’ and it goes after by item ‘d’ and it is the second child of ‘b’. As per rule, 2 will add in the bracket after ‘e:6’.

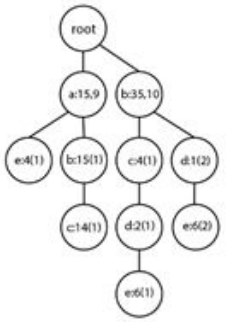

Figure 3.11: step 9_insertion of T4 items

The value of item ‘e’ is 6 and calculated by multiplying intra-transaction frequency of e (3) and external weight of e (2). As ‘b’ has already a child before and now another child is added on the tree so ‘e’ becomes the second child of ‘b’ and (2) is added to indicate the number of children.

Step 5:

Now we are on the final phase of our tree construction process. Transaction T5 has three items ‘c’, ‘d’ and ‘e’. Item ‘c’ is new in this tree. So it will connect directly with the root. And items will be added gradually right after ‘c’. Value of ‘c’ in this transaction which is 6 (intra-transaction frequency of c (3) and external weight of c (2)).

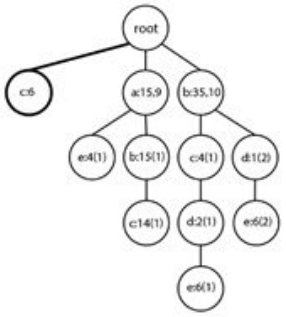

Figure 3.12: step 10_insertion of T5 items

Step 5.1:

Now second item of T5 will be added in the tree. This item will consider as the first child of ode ‘c’. So 1 will be added right after the value of ‘d’.

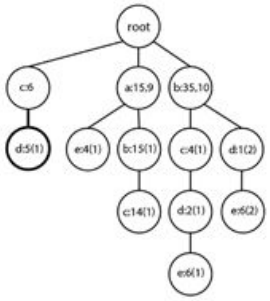

Figure 3.13: step 11_insertion of T5 items

The value of item ‘d’ is 5 and calculated by multiplying intra-transaction frequency of d (5) and external weight of d (1).

Step 5.2:

It is the last item of transaction T5 and also tree construction process. This item will also consider as the first child of ode 'c'. So 1 will be added right after the value of 'e'.

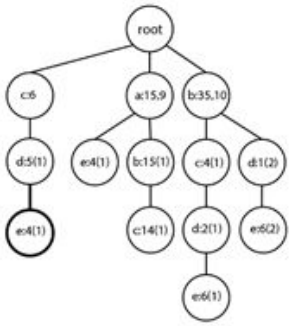

Figure 3.14: step 12_insertion of T5 items

The value of item 'e' is 4 and calculated by multiplying intra-transaction frequency of e (2) and external weight of e (2).

Final Step:

When all the items of the transaction database are added in the tree, it will look like as follows:

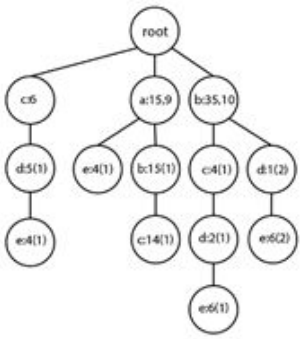

Figure 3.15: Final Tree

3.3.2 Tree Mining Approach

Now for our WIPM-growth mining approach, we are going to use table 3.1. And also we need to build a header table maintaining lexicographic canonical order.

Header Table
a
b
c
d
e

Table 3.3: Header Table

One of the essential characteristics of using lexicographic canonical order is that while mining, it uses the bottom-up approach that is mining starts from item ‘e’.

Also, we set 10 as a threshold value, which will help to prune patterns during our mining process. Pattern value or CWS value should be more significant than the threshold value to avoid mining from the tree.

When ‘e’ projected into the database, then we get four patterns which are ‘ae’. ‘be’, ‘ce’, and ‘de’. Now steps of calculating CWS calculation for all patterns are given below:

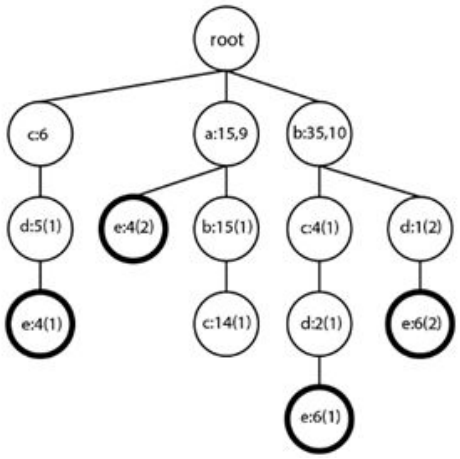

Figure 3.16: 'e' projected database

Step 1:

At first CWS calculation of pattern 'ae' will take place.

Here the combined weight (CW) of a=9 and the combined weight (CW) of e=4. Length(a)=1 and length (e)=1.

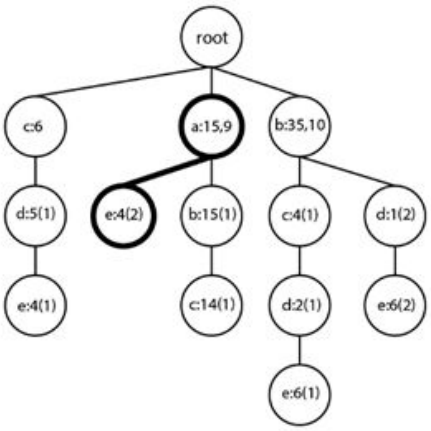

Figure 3.17: CWS calculation of 'ae'

The CWS value of 'ae' $= \frac{9(CW\,of\,a)*1(length\,of\,a)+4(CW\,of\,e)*1(length\,of\,e)}{1(length\,of\,a)+1(length\,of\,e)} = \frac{(9*1)+(4*1)}{1+1} = \frac{13}{2} = 6.5$

Step 2:

There are two paths to connect with ‘e’ from ‘b’. Separate CWS calculation for both the paths will take place at first, and then we will combine both of the CWS values to get the final CWS value for the pattern ‘be’.

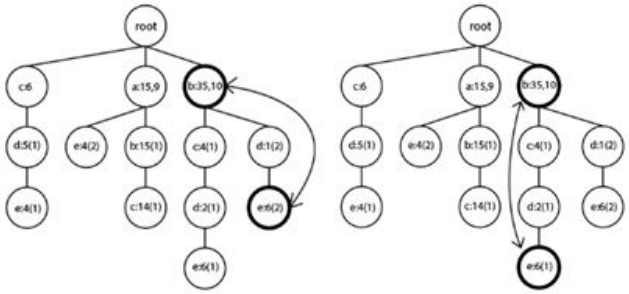

Figure 3.18: CWS calculation of ‘be’

At first, for the path (1) we calculate the CWS calculation of ‘b_1e_1’ (first child). Here the combined weight (CW) of b_1=35 and the combined weight (CW) of e_1=6. Length(b_1) =1 and length (e_1) =1.

$$\text{The CWS value of } `b_1e_1' = \frac{35(CW \text{ of } b_1)*1(\text{length of } b_1)+6(CW \text{ of } e_1)*1(\text{length of } e_1)}{1(\text{length of } b_1)+1(\text{length of } e_1)} = \frac{(35*1)+(6*1)}{1+1} = \frac{41}{2} = 20.5$$

And now for the path (2), we calculate CWS calculation of ‘b_2e_2’ (second child). Here the combined weight (CW) of b_2=10 and the combined weight (CW) of e_2=6. Length(b_2) =1 and length (e_2) =1.

$$\text{The CWS value of } `b_2e_2' = \frac{10(CW \text{ of } b_2)*1(\text{length of } b_2)+6(CW \text{ of } e_2)*1(\text{length of } e_2)}{1(\text{length of } b_2)+1(\text{length of } e_2)} = \frac{(10*1)+(6*1)}{1+1} = \frac{16}{2} = 8$$

CWS of ‘be’ $= b_1e_1 + b_2e_2 = 20.5 + 8 = 28.5$

‘be’ will be considered as an interesting pattern because the CWS values of ‘be’ are higher than threshold values.

Step 3:

Again, there are two paths to connect with 'e' from 'c'. Separate CWS calculation for both the paths will take place at first, and then we will combine both of the CWS values to get the final CWS value for the pattern 'ce'.

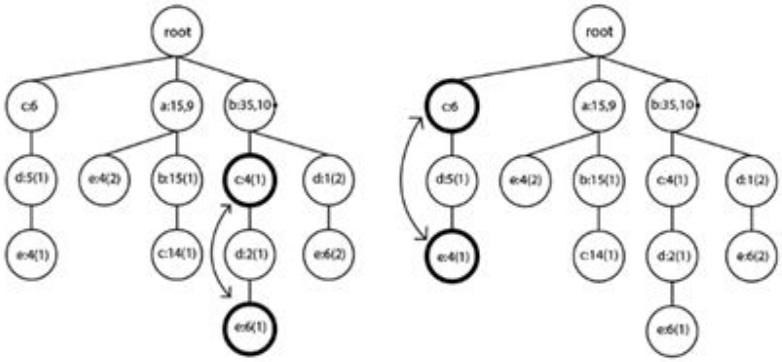

Figure 3.19: CWS calculation of 'ce'

For path (1), here the combined weight (CW) of c_1=4 and the combined weight (CW) of e_1=6. Length(c_1) =1 and length (e_1) =1.

$$\text{The CWS value of '}c_1e_1\text{'} = \frac{4(CW\,of\,c_1)*1(length\,of\,c_1)+6(CW\,of\,e_1)*1(length\,of\,e_1)}{1(length\,of\,c_1)+1(length\,of\,e_1)} = \frac{(4*1)+(6*1)}{1+1} = \frac{10}{2} = 5$$

And for path (2), the combined weight (CW) of c_2=6 and the combined weight (CW) of e_2=4. Length(c_2) =1 and length (e_2) =1.

$$\text{The CWS value of '}c_2e_2\text{'} = \frac{6(CW\,of\,c_2)*1(length\,of\,c_2)+4(CW\,of\,e_2)*1(length\,of\,e_2)}{1(length\,of\,c_2)+1(length\,of\,e_2)} = \frac{(6*1)+(4*1)}{1+1} = \frac{10}{2} = 5$$

So total CWS of 'ce' = $c_1e_1 + c_2e_2$ 5+5 = 10

'ce' will be considered as an interesting pattern because the **CWS** values of 'ce' are equal to the threshold value.

Step 4:

Finally, we will calculate the CWS of pattern ‘de’. Unlike previous patterns like ‘be’ and ‘ce’ pattern ‘de’ has three different paths. Now separate CWS calculations for the three paths are given below.

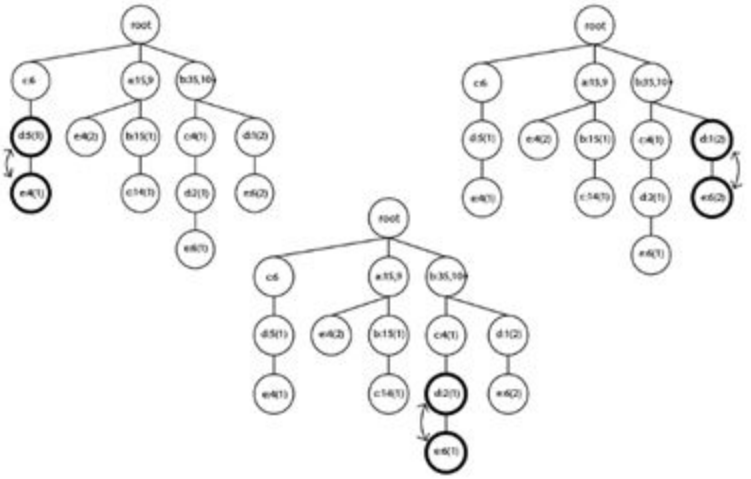

Figure 3.20: CWS calculation of ‘de’

For path (1), here the combined weight (CW) of d_1=2 and the combined weight (CW) of e_1=6. Length(d_1) =1 and length (e_1) =1.

$$\text{The CWS value of } `d_1e_1' = \frac{2(CW\ of\ d_1)*1(length\ of\ d_1)+6(CW\ of\ e_1)*1(length\ of\ e_1)}{1(length\ of\ d_1)+1(length\ of\ e_1)} = \frac{(2*1)+(6*1)}{1+1} = \frac{8}{2} = 4$$

For path (2), here the combined weight (CW) of d_2=1 and the combined weight (CW) of e_2=6. Length(d_2) =1 and length (e_2) =1.

$$\text{The CWS value of } `d_2e_2' = \frac{1(CW\ of\ d_2)*1(length\ of\ d_2)64(CW\ of\ e_2)*1(length\ of\ e_2)}{1(length\ of\ d_2)+1(length\ of\ e_2)} = \frac{(1*1)+(6*1)}{1+1} = \frac{7}{2} = 3.5$$

For path (3), here the combined weight (CW) of d_3=5 and the combined weight (CW) of e_3=6. Length(d_3) =1 and length (e_3) =1.

$$\text{The CWS value of } `d_3e_3' = \frac{5(CW of\ d_3)*1(length\ of\ d_3)+4(CW\ of\ e_3)*1(length\ of\ e_3)}{1(length\ of\ d_3)+1(length\ of\ e_3)} = \frac{(5*1)+(4*1)}{1+1} = \frac{9}{2} = 4.5$$

$$\text{So total CWS of 'de'} = d_1e_1 + d_2e_2 + d_3e_3 = 4+3.5+4.5 = 12$$

'de' will be considered as an interesting pattern because the **CWS** values of 'de' is greater than the threshold value.

Step 5:

Now we have four patterns 'ae', 'be', 'ce', 'de'. According to our algorithm (lexicographic canonical order) 'de' will be projected into the database and then we would find that two patterns 'bde' and 'cde'. So, following previous steps separate CWS calculations of 'bde' pattern are given below.

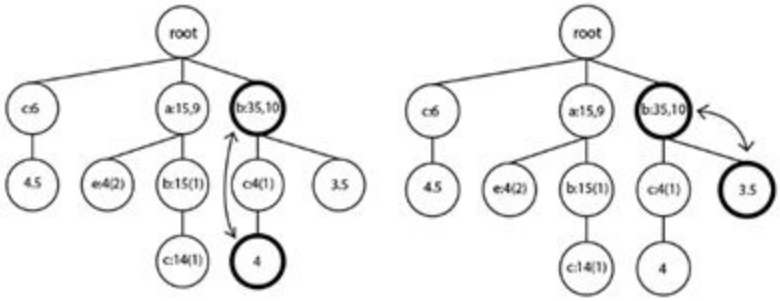

Figure 3.21: CWS calculation of 'bde'

Again, here we have found two paths. Now separate CWS calculations for the two paths are given below.

For path (1), here the combined weight (CW) of b_1=35 and the combined weight (CW) of de_1=4. Length(b_1) =1 and length (de_1) =2.

$$\text{The CWS value of '}b_1de_1\text{'} = \frac{35(CW\,of\,b_1)*1(length\,of\,b_1)+6(CW\,of\,de_1)*2(length\,of\,de_1)}{1(length\,of\,b_1)+2(length\,of\,de_1)} = \frac{(35*1)+(4*2)}{1+2} = \frac{43}{3} = 14.33$$

And for the path (2), here the combined weight (CW) of b_2=10 and the combined weight (CW) of de_2=3.5. Length(b_2) =1 and length (de_2) =2.

$$\text{The CWS value of '}b_2de_2\text{'} = \frac{10(CW\,of\,b_2)*1(length\,of\,b_2)+3.5(CW\,of\,de_2)*2(length\,of\,de_2)}{1(length\,of\,b_2)+2(length\,of\,de_2)} = \frac{(10*1)+(2*3.5)}{1+2} = \frac{17}{3} = 5.66$$

So total CWS of b_1de_1 + b_2de_2 = 14.33+5.66 = 20

'bde' will be considered as an interesting pattern because the **CWS** values of 'bde' are higher than the threshold value.

Step 6:

Again, we found two paths to calculate the CWS value of 'cde'. Now separate CWS calculations for these two paths are given below.

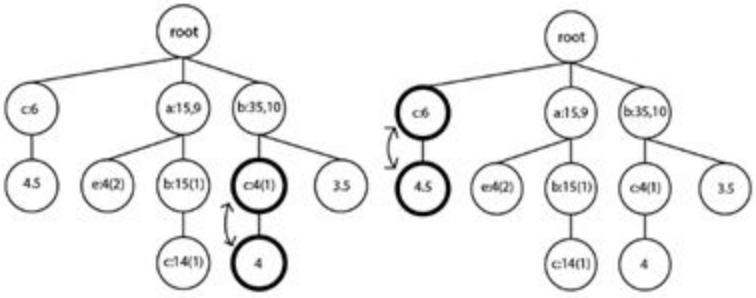

Figure 3.22: CWS calculation of 'cde'

For path (1), here the combined weight (CW) of c1=4 and the combined weight (CW) of de1=4. Length(c1) =1 and length (de1) =2.

$$\text{The CWS value of } `c_1de_1' = \frac{4(CW\,of\,c_1)*1(length\,of\,c_1)+4(CW\,of\,de_1)*2(length\,of\,de_1)}{1(length\,of\,c_1)+2(length\,of\,de_1)} = \frac{(4*1)+(4*2)}{1+2} = \frac{12}{3} = 4$$

And for the path (2), here the combined weight (CW) of c_2=6 and the combined weight (CW) of de_2=4.5. Length(c_2) =1 and length (de_2) =2.

$$\text{The CWS value of } `c_2de_2' = \frac{6(CW\,of\,c_2)*1(length\,of\,c_2)+4.5(CW\,of\,de_2)*2(length\,of\,de_2)}{1(length\,of\,b_2)+2(length\,of\,de_2)} = \frac{(6*1)+(4.5*2)}{1+2} = \frac{15}{3} = 5$$

$$\text{So total CWS of 'cde'} = c_1de_1 + c_2de_2 = 4+5 = 9$$

'cde' will not be considered as an interesting pattern because the CWS values of 'cde' are less than the threshold value.

Step 7:

Now we have two patterns 'bde' and 'cde'. According to our algorithm (lexicographic canonical order) 'cde' will be projected into the database and then we would find that only one pattern 'bcde'. Now we will calculate CWS for the pattern 'bcde'.

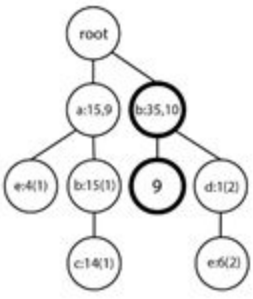

Figure 3.23: CWS calculation of 'bcde'

Here the combined weight (CW) of b=35 and the combined weight (CW) of cde=9. Length(b) =1 and length (cde) =3.

$$\text{The CWS value of 'bcde'} = \frac{35(CW\,of\,b)*1(length\ of\ b)+9(CW\ of cde)*3(length\ of cde)}{1(length\ of\ b)+3(length\ of\ cde)} = \frac{(35*1)+(9*3)}{1+3} = \frac{37}{4} = 15.5$$

'bcde' will be considered as an interesting pattern because the **CWS** values of 'bcde' are higher than the threshold value.

3.4 WIPM Growth Analysis

Algorithm 1 is the basic procedure of our work. Database with internal weight, weight table are taken as input at the beginning of our procedure. This procedure generates all the weighted pattern as output.

First of all, using header table HT is build according to weight descending order. For inserting each transaction we use the outer loop. Then we sort them according to the header table (HT) and calculate CWS (combined weighted support).

Initial if conditions in the outer loop check whether the window is filled or not. If window is overload it simple delete the oldest batch and shift the rest of the batch. After that, inner loop basically insert each transaction from the batch into the tree by calculating TAMP value for the transaction. By this time if any mining request comes from user minimum weighted threshold is taken as input. Then algorithm starts mining using the threshold from the current state of the tree which is done by the last if condition.

Input: Database with internal weight, Weight table

Result: List of Weighted Frequent Patterns

initialization;

Create the Header table HT and store the items according to

weight descending order.;

foreach Transaction $T_j \in B_i$ **do**

Sort the items in T_j according to HT;

Calculate CWS;

Insert T_j into Tree;

end

if *Any Mining Request From User* **then**

Input: δ from user

Calculate GMAXW;

foreach *item I from Bottom of HT* **do**

if TAMP(I) * GMAXW ≥δ **then**

Call CandidateTesting (I, TAMP(I));

Create Prefix Tree PT_I and Header Table HT_I

for item I;

Call MiningTree (PT_I, HT_I, I, $Weight_I$);

end

end

end

Algorithm 1: WIPM Growth Approach Algorithm

At the beginning of mining process GMAX is calculated from the header table and mining is started from the bottom of the table. For a particular item-set first whether the WexpSup of the item-set cross the threshold is checked.

After that CandidateTesting procedure is called to generate the frequent patterns. Than prefix tree is built for that item-set and recursively mine frequent patterns from prefix tree.For each items in the header table, algorithms generate prefix tree and recursively find frequent patterns from prefix tree. CandidateTesting procedure basically finds the actual frequent patterns by calculating actual WeightedExpected support and comparing with the threshold provided by the users. The procedure takes item-setandTAMP for that item-setas input and determines whether the item-set is frequent or not. Loop in the procedure calculate the total weight of all items belong to the tem-setand find the actual weight of the item-set by simply calculating the average weight that is by dividing the total weight by the length of item-set. Than check whether the actual weighted expected support is more than the threshold or not.

Input: Itemset I, Total Maximum Probability of I ($TAMP_I$)

W_I is the actual weight of I;

Set $W_I = 0$;

foreach *Item* $x_i \in I$ **do**

$W_I = W_I$ + Weight (x_i);

end

$W_I = W_I \div$ Length (I);

if $W_I * TAMP_I > \Sigma$ **then**

Add I to the weighted frequent List;

end

Algorithm 1: WIPM Growth Approach Algorithm

Loop in the procedure calculate the total weight of all items belong to the item-set and find the actual weight of the item-set by simply calculating the average weight that is by dividing the total weight by the length of item-set.Than check whether the actual weighted expected support is more than the threshold or not.

Chapter 4

Experimental Analysis

A proper description of different datasets, followed by the performance of the WIP-growth approach is the main content of this chapter. A comparison of performance between our approach and WFPM-growth. The most similar approach to our works in order to clarify the advantages of WIP-growth.

To evaluate the performance of our proposed method, we use several datasets that are commonly used by data-mining scientists to determine the performance of their techniques.

4.1 Datasets

Data mining is all about discovering and extracting knowledge accurately from raw data. This useful information matters to different types of applications and transforming data into some logical structure for further uses.

Some data or datasets are needed to evaluate our work performance and indicate the usefulness of our proposed method, in order to find interesting patterns from data. In this work, several real lives, for example, Chess, Mushroom datasets and synthesis dataset likes T1014D100K are used to measure the performance of WIP-growth.

Details of each dataset are discussed in the following sections.

4.1.1 Mushroom

This data set includes descriptions of hypothetical samples corresponding to 23 species of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500-525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended.

This latter class was combined with the poisonous one. So it is clear that there is no simple rule for determining the edibility of a mushroom; no rule like "leaflets three, let it be" for Poisonous Oak and Ivy.

Data Set Characteristics	**Multivariate**	**Number of Instances**	**8124**
Attribute Characteristics	Categorical	Number of Attributes	22
Associated Tasks	Classification	Missing Values	Yes

Table 4.1: Mushroom Dataset

Table 4.1 shows the fundamental characteristic of Mushroom data, which reflects that the Mushroom dataset is multivariate, and analysis is based on more than two variables per observation. The attribute is categorical implies that each variable has more than one category, but there is no intrinsic ordering to the categories. Here is a brief description of the attributes giving below.

Attribute Information: (classes: edible=e, poisonous=p)

• cap – shape: bell=b, conical=c, convex=x, flat=f, knobbed=k, sunken=s

• cap-surface: fibrous=f, grooves=g, scaly=y, smooth=s

• cap-color:

brown=n, buff=b, cinnamon=c, gray=g, green=r, pink=p, purple=u, red=e, white=w, yellow=y

• bruises?: bruises=t, no=f

• odor: almond=a, anise=l, creosote=c, fishy=y, foul=f, musty=m, none=n, pungent=p, spicy=s

• gill-attachment: attached=a, descending=d, free=f, notched=n

• gill-spacing: close=c, crowded=w, distant=d

• gill-size: broad=b, narrow=n

• gill-color: black=k, brown=n, buff=b, chocolate=h, gray=g, green=r, orange=o, pink=p, purple=u, red=e, white=w, yellow=y

• stalk-shape: enlarging=e, tapering=t

• stalk-root: bulbous=b, club=c, cup=u, equal=e, rhizomorphs=z, rooted=r, missing=?

• stalk-surface-above-ring: fibrous=f, scaly=y, silky=k, smooth=s

• stalk-surface-below-ring: fibrous=f, scaly=y, silky=k, smooth=s

• stalk-color-above-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w

• stalk-color-below-ring: brown=n, buff=b, cinnamon=c, gray=g, orange=o, pink=p, red=e, white=w, yellow=y

• veil-type: partial=p, universal=u

• veil-color: brown=n, orange=o, white=w, yellow=y

• ring-number: none=n, one=o, two=t

• ring-type: cobwebby=c, evanescent=e, flaring=f, large=l, none=n, pendant=p, sheathing=s, zone=z

• spore-print-color: black=k, brown=n, buff=b, chocolate=h, green=r, orange=o, purple=u, white=w, yellow=y

• population: abundant=a, clustered=c, numerous=n, scattered=s, several=v, solitary=y

• habitat: grasses=g, leaves=l, meadows=m, paths=p, urban=u, waste=w, woods=d

Missing attributes value 2480 of them denoted by"?".

Class description: edible 4208 that is 51% and poisonous 3916 that is 49%.

4.1.2 Chess

Chess is a real-life dataset built by domain theories for generating the legal moves of the chess game. It is one of the widely used datasets by data scientists to evaluate the performance of their methods. There are different types of domain theory of chess games such as Employs a geometric representation for states, with each square designated by an X, Y coordinate and square connectivity computed by vectors. Generates legal moves by first generating pseudo moves then eliminating those that result in the moving player being in check. Some general information on Chess dataset is given in Table 4.2

Data Set Characteristics	**Domain Theory**	**Number of Instances**	**1000**
Attribute Characteristics	**NA**	**Number of Attributes**	**NA**
Associated Tasks	**NA**	**Missing Values**	**NA**

Table 4.2: Chess dataset

4.1.3 General Characteristics of Dataset

Some well-known synthesis datasets like T1014D100K, Kosarak, etc. are also used to test the WIP growth algorithm. T1014D100k is generated using the generator from the IBM Almaden Quest research group is sparse in nature. Some important characteristics of datasets that are used in our methods to be tested are given in Table 4.3.

Data Set	No. of Transaction	No. of Distinct Value(D)	Average Transaction Length(A)	Dense/Sparse Characteristics Ratio R=(A/D)*100
Mushroom	8124	119	23	19.327
Chess	3196	75	37	49.33
Kosarak	990002	41270	8.1	0.0196
T101D100K	100000	870	10.1	1.16

Table 4.3: Characteristics of Dataset

4.2 Experimental Result Analysis

All the datasets which are collected from the UCI machine learning and FIMI repository do not contain any weight associated with each item. However, in our WIP-growth approach, we need weights for each item. So some preprocessing process has to be performed on the dataset before analysis the result like generating weight.

However, we follow some probabilistic distribution to generate weight depending on the demand for the application. Analyzing real-life situation patterns whose frequency is average is more demanding rather than extremely frequent patterns or those patterns that are cross the frequent boundary.

That is why we follow normal distribution to assign the probability of items by giving the highest probability to those items whose frequency is average and gradually assign a lower probability to less frequent items.

For the experiment, the weight of the items also generates randomly following the normal distribution. We perform an analysis of our algorithm in terms of runtime and memory consumed by the algorithms.

Distribution = Normal
External Weight : Mean= 0.7 and Standard Deviation = 0.2
Intra-transaction Occurrence : Mean = 20 and Standard Deviation = 4

Table 4.4: Data used in result analysis

Information that provides in table 4.4 is used for our experimental analysis. The normal distribution method has used for calculating ‘external weight’ and ‘intra-transaction occurrences’.

4.2.1 Effects of Threshold and Time in T1014D100K

To perform the runtime analysis of our methods we conducted several experiments on dataset T1014D100K, Chess and Mushroom.

Threshold	Time(min)
5	453
7	147
9	102
11	86
13	67

Table 4.5: T1014D100K

In this section, we discuss the effect of threshold and time in the synthesis dataset T1014D100K. Table 4.5 contains threshold value and time in respect of each threshold.

$$Threshold = mean\ value\ of\ intra_{transaction} occurrence * mean\ value\ of\ external\ weight * selected\ percentage\ from\ total\ data$$

The above equation is used for calculating the threshold value. For example, if we want to calculate the threshold value for 50% of total data then it will be 20*0.7*0.5=7.

Table 4.5 also shows a systematic relationship between threshold and time. If we increase the threshold value, then the number of frequent items will be less so less amount of time is needed for finding frequent patterns.

Figure 4.1: Continuous representation of threshold vs time (T1014D100K)

Fig. 4.1 demonstrates the systematic relationship of threshold and time for the synthetic data in a continuous way. At the beginning when threshold value is low, a large amount of time needed to find frequent items. And after that the required amount of time falls down drastically while increasing the threshold value.

Fig. 4.2 demonstrates the systematic relationship of threshold and time for the synthetic data in a discrete way. Idea behind this representation is the same but it helps to understand the change in an easy way. At the beginning when threshold value is low, a large amount of time needed to find frequent items. And after that the required amount of time falls down drastically while increasing the threshold value.

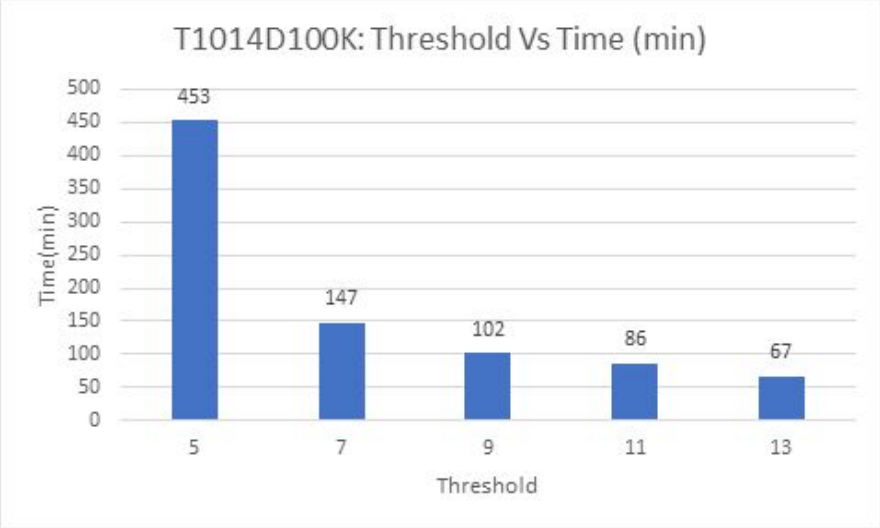

Figure 4.2: Discrete representation of threshold vs time (T1014D100K)

4.2.2 Effects of Threshold and Time for Mushroom Dataset

In this section, we discuss the effect of threshold and time in real life dataset which is Mushroom. Table 4.6 contains threshold value and time in respect of each threshold.

Threshold	Time(min)
5	3723
7	2777
9	1356
11	1156
13	1023

Table 4.6: Mushroom Dataset

$$Threshold = mean\ value\ of\ intra_{transaction} occurrence * mean\ value\ of\ external\ weight * selected\ percentage\ from\ total\ data$$

The above equation is used for calculating the threshold value. For example, if we want to calculate the threshold value for 50% of total data then it will be 20*0.7*0.5=7.

Table 4.6 also shows a systematic relationship between threshold and time. If we increase the threshold value, then the number of frequent items will be less so less amount of time is needed for finding frequent patterns.

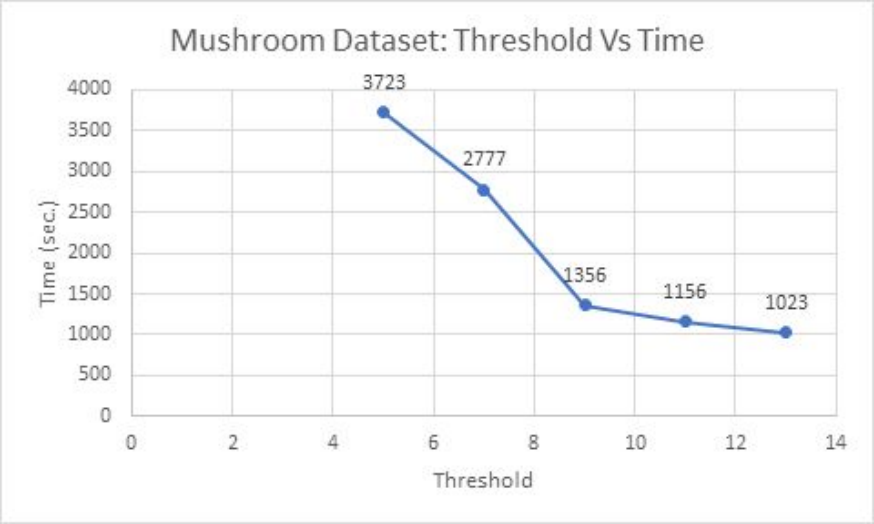

Figure 4.3: Continuous representation of threshold vs time (Mushroom)

Fig. 4.3 demonstrates the systematic relationship of threshold and time for the real life data in a continuous way. At the beginning when threshold value is low, a large amount of time needed to find frequent items. And after that the required amount of time falls down gradually while increasing the threshold value.

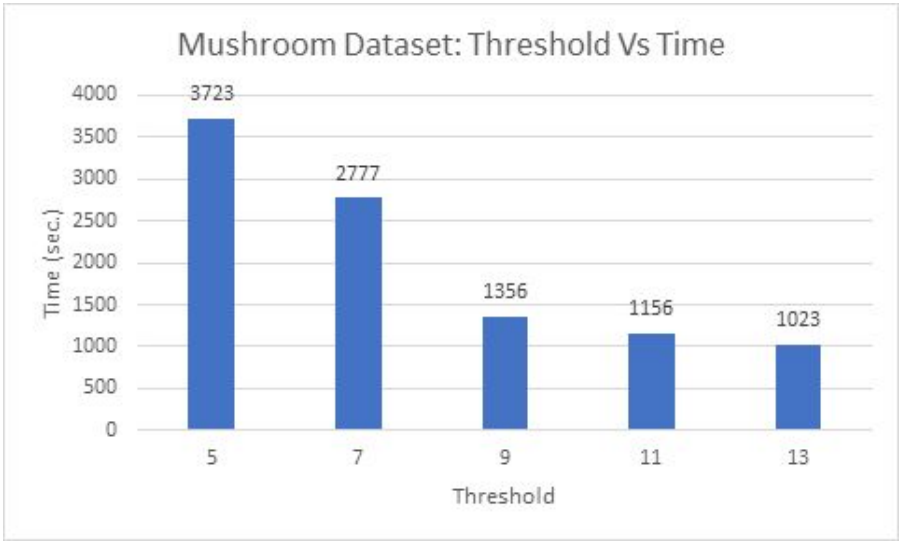

Figure 4.4: Discrete representation of threshold vs time (Mushroom)

Fig. 4.4 demonstrates the systematic relationship of threshold and time for the synthetic data in a discrete way. Idea behind this representation is the same but it helps to understand the change in an easy way. At the beginning when threshold value is low, a large amount of time needed to find frequent items. And after that the required amount of time falls down drastically while increasing the threshold value.

4.2.3 Effects of Threshold and Time for Chess Dataset

In this section, we discuss the effect of threshold and time in real life dataset which is Mushroom. Table 4.7 contains threshold value and time in respect of each threshold.

Threshold	Time(min)
5	2023
7	1807
9	1695
11	1452
13	512

Table 4.7: Chess Dataset

$$Threshold = mean\ value\ of\ intra_{transaction} occurrence * mean\ value\ of\ external\ weight * selected\ percentage\ from\ total\ data$$

The above equation is used for calculating the threshold value. For example, if we want to calculate the threshold value for 50% of total data then it will be 20*0.7*0.5=7.

Table 4.7 also shows a systematic relationship between threshold and time. If we increase the threshold value, then the number of frequent items will be less so less amount of time is needed for finding frequent patterns.

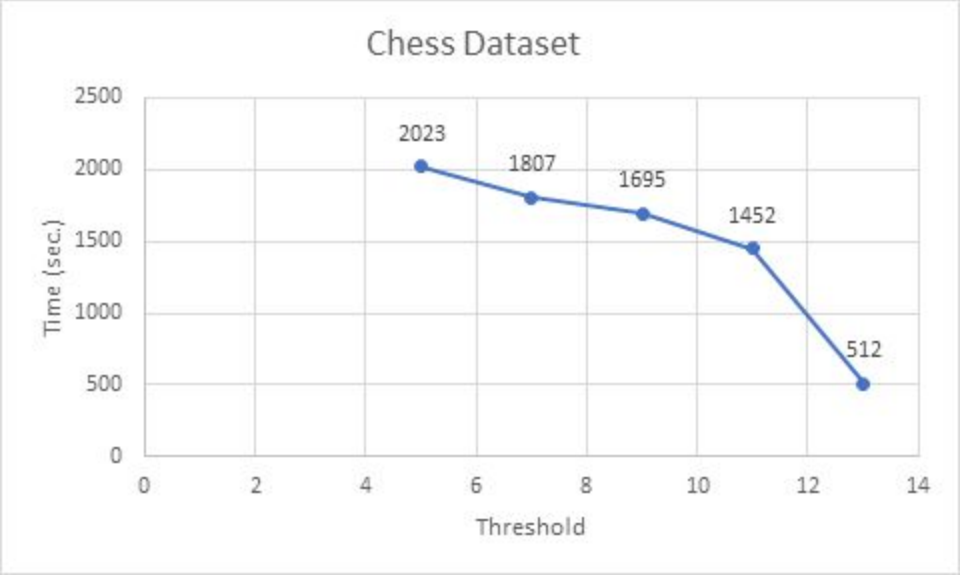

Figure 4.5: Continuous representation of threshold vs time (Chess)

Fig. 4.5 demonstrates the systematic relationship of threshold and time for the real life data in a continuous way. At the beginning when threshold value is low, a large amount of time needed to find frequent items. And after that the required amount of time falls down gradually while increasing the threshold value.

Figure 4.6: Discrete representation of threshold vs time (Chess)

Fig. 4.6 demonstrates the systematic relationship of threshold and time for the synthetic data in a discrete way. Idea behind this representation is the same but it helps to understand the change in an easy way. At the beginning when threshold value is low, a large amount of time needed to find frequent items. And after that the required amount of time falls down drastically while increasing the threshold value.

4.3 Conclusion

In this thesis work we basically propose a novel tree structure that successfully and efficiently captures the uncertain data stream and a mining algorithm that extracts only significant patterns from most recent window. Our main contribution is to mine only important patterns rather than all frequent data without considering their necessity. Recent world applications have hungered for only relevant patterns for which they actually mine large datasets, not just frequent patterns. Focusing on this demand we developed a novel approach called WIPM - growth which solves the problem of finding significant patterns from uncertain data stream. As our algorithm generates only important patterns it is less time consuming and also uses fewer memory resources.

Chapter 5

Conclusion

In our thesis work, we developed an intra-transaction weight-based mining strategy. Our proposed WIPM-growth is a complete method to generate interesting patterns by calculating combined weighted support (CWS) from a transactional database. WIPM-growth just doesn't generate all frequent patterns with considering weight but also considering intra-transaction frequency.

5.1 Research Summary

Throughout this book, we organize our works like in Chapter 1 we introduced our research topic by discussing the current situation of data mining, why uncertain data mining is required in today's world. Especially in section 1.2, we specify the need for mining large dataset with the help of weighted interesting pattern. Problems of the existing systems that deal with frequent patterns and weighted interesting patterns are discussed in section 1.3. In section 1.4 challenges are discussed in detail that we faced during our work. Section 1.5 represents our contribution to data mining research.

In chapter 2 we discussed some previous works that help us. Section 2.4 introduces two important algorithms of data mining. Section 2.2, 2.3, 2.5, 2.6 briefly described some important terminologies. Details description of some important data mining techniques and procedures are given in section 2.7. Chapter 2 is mainly about those important methods that basically inspired us.

From chapter 3 we basically start describing our actual works. First of all, in section 3.2 we set our goals. We introduced several concepts that shape our work in section 3.3. Finally in section 3.4 we described our tree constructing methods, mining techniques, and analysis in detail.

Chapter 4 basically shows the experimental results, analysis of results, several comparisons with existing methods and observing different characteristics of datasets that used to analyze our methods.

5.2 Limitation

While we are constructing our tree, if any leaf node item repeats, we place the value of the repeated item in the same leaf node. And the second child of that leaf node starts from there. We could not manage to find a solution to that problem. And also we could not manage to merge our tree at the end.

5.3 Future Work

For further work, we will develop a framework for constructing a tree and provide a solution for merging it. In the future, we will try to apply some other constrain to find more strong and interesting patterns that have more significant applications and become more realistic to users.

Bibliography

[1] C. K.-S. Leung, Q. I. Khan, Z. Li, and T. Hoque, "Cantree: a canonical- order tree for incremental frequent-pattern mining," *Knowledge and Infor- mation Systems*, vol. 11, no. 3, pp. 287–311, 2007.

[2] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "Huc-prune: an efficient candidate pruning technique to mine high utility patterns," *Applied Intelligence*, vol. 34, no. 2, pp. 181–198, 2011.

[3] ——, "Efficient tree structures for high utility pattern mining in incremen- tal databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 12, pp. 1708–1721, 2009.

[4] U. Yun, "Efficient mining of weighted interesting patterns with a strong weight and/or support affinity," *Information Sciences*, vol. 177, no. 17, pp. 3477–3499, 2007.

[5] U. Yun and J. J. Leggett, "Wfim: weighted frequent itemset mining with a weight range and a minimum weight," in *Proceedings of the 2005 SIAM international conference on data mining*. SIAM, 2005, pp. 636–640.

[6] A. Inokuchi, T. Washio, and H. Motoda, "An apriori-based algorithm for mining frequent substructures from graph data," in *European conference on principles of data mining and knowledge discovery*. Springer, 2000, pp. 13–23.

[7] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM sigmod record*, vol. 29, no. 2. ACM, 2000, pp. 1–12.

[8] E. Spyropoulou, T. De Bie, and M. Boley, "Interesting pattern mining in multi-relational data," *Data Mining and Knowledge Discovery*, vol. 28, no. 3, pp. 808–849, 2014.

[9] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "Mining weighted frequent patterns in incremental databases," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2008, pp. 933–938.

[10] W. Cheung and O. R. Zaiane, "Incremental mining of frequent patterns without candidate generation or support constraint," in *Seventh Interna- tional Database Engineering and Applications Symposium, 2003. Proceed- ings.* IEEE, 2003, pp. 111–116.