CPTSW: An Efficient approach for Mining static Weighted Frequent Patterns From Data Stream

AUTHORS

Safa Marwa Moonmoon

2015-1-60-026

MD. Mominul Islam

2015-1-60-147

Tasnim Jahan Munny

2015-1-60-038

SUPERVISED BY

Jesan Ahammeds Ovi

Lecturer, Dept. of CSE, East West Univeersity

A thesis submitted in partial fulfillment for the degree of B.Sc. in Computer Science and Engineering

in the

Department of Computer Science and Engineering

EAST WEST UNIVERSITY

February 2020

Declaration of Authorship

We, Safa Marwa Moonmoon, MD Mominul Islam , Tasnim Jahan Munny declare that this thesis titled, "An Efficient approach for Mining static Weighted Frequent Patterns From Data stream" and the work presented in it are the outcomes of investigation performed by us under supervision of Lecturer Jesan Ahammed Ovi, Department of Computer Science and Engineering,East West Univercity. We also declare that no part of this thesis has been or is being submitted elsewhere for the award of any degree or-diploma.

Jesan Ahammed Ovi
Lecturue
Department of Computer Science and Engineering
East West Univercity
Thesis Supervisor

Signature and Date of Supervisor: ____________________

Safa Marwa Moonmoon	MD. Mominul Islam	Tasnim Jahan Munniy
(ID: 2015-1-60-026)	(ID: 2015-1-60-147)	(ID: 2015-1-60-038)
Signed:	Signed:	Signed:
______________	______________	______________
Date:	Date:	Date:
______________	______________	______________

"Nature loves to mystery!The mysterious human mind is more mysterious than the complexity and mystery of all the stars."

-Humayun Ahmed

EAST WEST UNIVERSITY

Abstract

Department of Computer Science and Engineering

Bachelor in Science

by

Safa Marwa Moonmoon

2015-1-60-026

MD. Mominul Islam

2015-1-60-147

Tasnim Jahan Munny

2015-1-60-038

In the recent year data mining is one of the most demanding sectors of computer science which basically deals with discovering frequent patterns by using methodologies, techniques and intelligence tools from databases. As the modern technology is growing rapidly, high volume of data with several features are generated by modern applications. When data set's flowing velocity is high but applications demand real time analyzing of data depends on immediate features then situation has become more challenging. Several researches has been made in order to assuage the challenges regarding data streams.To find the actual patterns as though the nature of data sets is streams, frequent patterns of those data may be huge and requires further mining. Today's generation are very much interest in patterns that are significant for them and not just all frequent patterns. Already many researchers work with this topic but they are not enough sufficient. In this thesis we proposing a novel tree based approach, CPTSW-growth which is able to capture the uncertain data streams depending on the importance of applications and only produces significant patterns.

Acknowledgements

First of all, We would like to thank our thesis supervisor MD. Jesan Ahmed Ovi of the Department of Computer Science Engineering at East West University. He consistently allowed this thesis to be our own work,but steered us in the right direction whenever he thought we needed it.The consistent guidance from him has made this work come into fruition. his door was always open for us whenever we came to any problems,needed any help or had any question. this accomplishment would not have been possible without him.

Finally, we must express our very profound gratitude to our parents for providing us with unfailing support and continuous encouragement throughout our years of study and through the process of researching and writing this thesis. Thank you.

Author
Safa Marwa Moonmoon
MD. Mominul Islam
Tasnim Jahan Munny

Contents

List of Figures

List of Tables

Abbreviations

GMAXW	**G**lobal **M**aximum **w**eight
LMAXW	**L**ocal **M**aximum **w**eight

Dedicated to our parents.

Chapter 1

Introduction

“Data mining is the process of extracting hidden patterns of data according to different aspects for categorization into useful, significant and interesting information, which is collected and combined in common areas, such as for efficient analysis, data warehouses, facilitating business decision making ,data mining algorithms, and other information requirements to increase revenue and cut costs”.

The methods of data mining are classification, regression, clustering and pattern mining. Pattern mining is the most important role in data mining. Pattern mining covers the maximum field such as frequent pattern mining, sequential pattern mining, graph mining et cetera [1–5]. But in the real world, pattern mining is not enough for knowledge discovering. Because in the real scenario different items has different value. For an example, a shop has different type of dress. But all dresses are not same price. If frequent pattern mining is applied in this scenario then it will give the unexpected value. That’s why weighted frequent pattern mining is applied in this filed. Weighted frequent pattern mining [6–11] is more useful for extracting the useful knowledge and interesting pattern because different items has different weight. In real world weighted pattern mining is applied in different type of area such as web traversal and market value analysis. And the most important field is biomedical data analysis because there are many types of gene and those are not same, any disease are not caused by single gene but also it is caused by a sequence of gene where different gene is carried different information. In recent decade, automatic measurement such as different type of sensors value is produced a large amount of data and constantly highly increased. Resultants, this large amount of data is flooded. That’s why Data Steam [12–15] algorithm was introduced. Data Stream algorithm used for the knowledge extracting from continuous data, rapid-data, continuous record. Many data stream algorithm is used for predict the class or new instances. Examples of data streams include phone conversation , ATM transactions, Web searches and some sensors data is continuous and unbounded[16–20]. But the problem is if weighted frequent mining pattern mining is applied here it would be more critical and generate large amount of false pattern. And the important thing is maximum time very old data is not needed. For an example, weather sensor gives the continuous data, when the mining algorithm is applied on this data then it gives so many unnecessary patterns those are not interesting. Because there has so many old data and time has changed so old data is not needed anymore. That's why sliding base data streams is applied in this scenario and this algorithm find the latest information from the data stream.

1.1 Motivation

In recent decade, automated measurement and collection of data has produced large amounts of data. Resultantly, development in technology have led to a flood of data. For examples of data streams contain sensor database, call center data records , network traffic so on. This erect volume and high speed posture breed great challenge to mine data sets for the data mining researcher association . Stream of data exhibit several unique characteristics: concept-drift, infinite length, feature-evolution concept-evolution, and confined labeled data. The underlying concept of data changes over time generates Concept-drift in data streams. Concept-evolution happens when new classes develop in streams. When features vary with time in data streams Feature-evolution occurs. Scarcity of labeled data is another challenges in data stream so it is not probable to manually label each and every data points in the stream. Each of these exorcism adds a challenge to mine data streams. Researchers have introduced weighted frequent dataset mining algorithms which reflect the worth of items. Their main focus of weighted frequent dataset mining suspense fulfil the downward closure property. Every weighted association rule mining algorithms based on the Apriori algorithm. however algorithms of pattern growth are more feasible than Apriori based algorithms.

1.2 Challenges

In this research pattern growth mining approach is proposed to solve weighted frequent pattern mining for data dream mining, which supports the downward closer property. This takes time and memory. The main challenges is to compact the prefix tree of the previous work weighted frequent pattern mining for Data stream (WFPMDS) algorithm. This needs to be done by restructuring the prefix tree by swapping the nodes according to the frequency count value.

1.3 Objective

Pattern mining is a form of data mining, for example itemset mining in retail market analysis, has been proved to be the most essential and useful scenario in daily life. But in real life problems are not simple. Real life problems are much more difficult and complex than item set mining. For representing complex objects graph is the most useful tool, so graph mining has become the most important part of data mining. Graph has an effective representational power that has been promoted by authorizing weights on relations in objects. Goals of this thesis is to:

1. Finding interesting weighted frequent patterns from databases.
2. Reduce memory use by compacting the prefix tree.
3. Ensure that proposed approach better performance than previous approaches.
4. Explain the scalability of the proposed approach.

1.4 Contribution

Motivated by real world scenarios, sliding window based technique WFPMDS (weighted Frequent pattern mining over Data Stream) is introduced. By using single scan, it can extract useful recent wisdom from data streams. However, this algorithm requires less mining time and uses less compact prefix tree. Comparing with [1, 17, 21**?** –27] mining from traditional static databases and mining from continuous data flow is more demanding problem. Our main focus is to solve all this problem together. Our main contributions of this paper include:

- A highly compact tree structure – compact pattern tree for static weights (CPTSW) is introduced to mine frequent patterns with static weights that significantly flourishes the performance with single database scan.
- A single scan mining algorithm is exhibited that can be wielded for finding weighted frequent patterns over data stream.
- Path adjusting method that works base on phase by phase tree restructuring method which enhances the degree of prefix in the tree formation.
- Through comprehensive experimental study pattern growth mining approach were observed for weighted frequent pattern mining.

Our technique transacts a pattern growth mining approach to fudge the level-wise candidate generation and test problem. Our technique can be applied to mining web path traversal patterns as well as retail market data. It can be useful in biomedical research and DNA analysis. As different web pages has different importance values, this algorithm can extract very useful knowledge about weighted frequent web path traversals using only one database scan in real time. Moreover for detecting certain disease,drugs and medicine that can be used by detecting the combination and amalgamation of weighted gene patterns. Stock market, sensor networks, telecommunication data are the other application of our algorithm.

1.5 Outline

Rest of the report is organized as follows:

- Chapter 2: Relevant background studies and weighted pattern mining for data stream mining related works.
- Chapter 3: Complete simulation on sample database of our proposed method.
- Chapter 4: Description, experimental result and comparison with existing models.
- Chapter 5: Summery of the whole research work and future research scope of this thesis.

Chapter 2

Related Works

this chapter we will discuss about the related topics, procedures of different types that will help us to implement our works and solve our problems. Background topics of our works and important concepts will also focused. This chapter will be start by giving illustration of important technicalities. At first we will discuss about frequent pattern mining , weighted frequent pattern mining , data streams and different types of parameters related to such data.

2.1 Background study

To set the platform for this work, we start by introducing the notion of frequent pattern mining, incremental and interactive pattern mining, weighted frequent pattern mining and data streams. Subsequently, we narrate the main demanding problem in weighted frequent pattern mining for data streams.

2.1.1 Frequent pattern mining

Pattern mining is a formation of using/developing data mining algorithms to discover interesting, unexpected and useful patterns in databases. Finding frequent patterns, casual structures, associations from data sets such as relational databases, transactional databases are known as frequent pattern mining. This process helps us to find the rules that enable us to calculate or predict the occurrence of specific item based on the frequency or occurrence in the dataset. For example, a set of items, such as new born baby milk and diaper that come frequently together in a transaction data set is a frequent item set. A substructure can prescribe to different structural formation, such as sub graphs, sub trees, sub lattices, which may be joined with item sets. I case a substructure appears frequently in a graph database, it is called a frequent structural pattern. A sub sequence like buying first a computer, then a memory card and then a digital camera, in case it occurs frequently in a purchasing memoir database, is frequent sequential pattern. Further more, it helps in classification, data indexing, clustering, and several data mining tasks. Frequent pattern mining is an important data mining persuasive in data mining research. Abundant literature has been introduce scalable and efficient algorithm for frequent pattern mining in transactional database to mainfold research such as sequential pattern mining, correlation mining, frequent pattern-based clustering, structured pattern mining, associative classification and correlation mining. Basic solution of frequent pattern mining is Appriori algorithm which is very effective and useful in association rule mining. Several database scan and level-wise candidate generation-and-test problem are corns of Appriori algorithm. Fp-growth algorithm solves this problem by applying two database scan and Fp-tree-based solution without any candidate formation

2.1.2 Incremental and interactive pattern mining

Data mining is an interactive procedure in nature. That method run modify the mined dataset , or the parameter of the previous mined database in each recurrences when users issues continuous series of similar type of data mining queries. The results of previous queries are required by Incremental mining algorithms. Incremental mining is such a data mining technique which can be applied for dynamic conditions or environment where database grows frequently. By utilizing the same data structure or prior mining results, with different minimum support threshold interactive data mining can be possible. Interactive data mining encourages users' learn, improve and understand of the problem to be solved, and stimulate users to investigate creative possibilities. Some research works [28] have developed incremental and interactive mining algorithms based on traditional frequent pattern mining algorithm. AFPIM has been improved incremental mining performance readjusts an Fp-tree using path adjusting mechanism. While database is growing and shrinking CanTree builds (1st book references) transactions in canonical order and sustains the whole database information in a tree. It has proposed the" build once mine many "property. CanTree is improved by CP-tree[28] by restructuring the incremental prefix-tree according to frequency descending order. That shows that their incremental prefix tree are quite enable and effective using currently available memory size in gigabyte range. Inc- WTP and WssWTP [28] are developed for incremental an interactive mining of web traversal patterns.

2.1.3 Weighted frequent pattern mining

A weight of an item is a non-negative real number which is assigned to reect the importance of each item in the transaction database[6, 8]. For a set of items I = i1,i2,......in, weight of a pattern Px1,x2,.......x2 is given as follows:

$$Weight(P) = \frac{\sum_{q=1}^{length(p)} Weight(X_q)}{length(P)} \ldots\ldots (1)$$

A weighted support or frequency of a pattern is defined as the value that results from multiple pattern support with the weight of the pattern [6, 8]. So , the weighted support of a pattern, P, is given as follows.

$$Wsupport(P) = Weight(P) \times Support(P) \ldots\ldots (2)$$

A pattern is called a weighted frequent pattern if the weighted support of the pattern is greater than or equal to theminimum threshold [6, 8]. A weighted support of a pattern is defined as the resultant value of multiplying the pattern's support with the weight of the pattern [6, 8]. A pattern is called a weighted frequent pattern if the weighted support of the pattern is greater than or equal to the minimum threshold [6, 8]. In the very beginning some weighted frequent pattern mining algorithms WARM [10], WAR [29] have been developed based on the Apriori algorithm [1, 6] candidate generation-and-test paradigm. WFIM [6] is the first FP-tree based weighted frequent pattern mining algorithm using two database scans over a static database. They have used a minimum weight and a weight range. Items are given fixed weights randomly from the weight range. They have arranged the FP-tree [3]in weight ascending order and maintained they downward closure property [3] on that tree. To extract the more interesting weighted frequent patterns, WIP [8] algorithm introduces a new measure of weight-confidence to measure strong weight anity of a pattern. "The WFIM [6] and WIP [8] algorithms shows that the main challenges of weighted frequent pattern mining is that weighted support of an item set does not have the downward closer property . Consider that item "a "has weight 0.6 and frequency 4, item "b" has weight 0.2 and frequency 5 ,and item set "ab" has frequency 3. According to equation (1) , the weight of item set "ab" will be (0.6=0.2)/2 =0.4 , and according to equation (2) it's weighted frequency will be 0.4 * 3= 1.2. The weighted frequency of "a" is 0.6 * 4 = 2.4, and of "b" is 0.2 *5 =1.0. If minimum threshold is 1.2, then pattern "b" is weighted infrequent but " ab " is weighted frequent. WFIM and WIP maintain the downward closure property by multiplying each item set's frequency by the maximum weight. In the above example, if item "a" has the maximum weight 0.6, then by multiplying it with the frequency of item "b", the result will be 3.0. so , pattern “b” will not be pruned at the early stage, and pattern “ab” will not be missed. At the final stage, this overestimated pattern “b” will finally be pruned by using its actual weighted frequency.” Some research works[5, 30–32] proposed single-pass mining algorithm based on traditional frequent pattern mining. Research works[12–14] developed algorithm for finding frequent pattern mining over data streams.[15, 33, 34] research works has developed frequent patterns from data streams using sliding windows. However, these research works, not applicable for weighted frequent pattern mining.

2.1.4 Data streams

Data stream means that data flows in a system in vast volumes, change dynamically and apparently infinite. This kind of data also can comprise multidimensional features. For this characteristics, data streams cannot be stored in traditional database system. Most of the system can be able to read streams once in sequential order. Data flows quickly in data streams, so there is no such luxury to scan or read data base multiple time for mining like other traditional database systems. That’s why building effective methods of mining data streams is more challenging than mining traditional database mining. To develop effective methods of mining such data streams database sequential research is effective. Different procedures, methods and algorithms are implemented by several researchers in order to develop frequent pattern from data streams [15, 35, 36]. This subsumes, collecting information from data stream in sliding windows or title windows that has been explored techniques like limited aggregation, micro-clustering and approximation. There are many application of data streams such as real time detection of anomalies in computer traffic, video stream, web search, text stream, sensor networks and so on.

2.1.5 Batch Size

Batch size basically means the number of transactions in one group[37]. Which denotes how many transaction we consider to store their item value in one partition of each node. In table 2.1 the batch size is 2 that means every 2 transaction is capture in separate partition of tree node.

2.1.6 Window size

As we can not hold the whole tree rather than summery of a given time, window size show us how many batch we have capture in a node[38]. If window size 3 so just 3 batch we can be captured when 4th batch has come 1st batch will be deleted from the node and new batch is stored. The tree in Figure 2.1 is an example to denote the window size

Batch	TID	Transactions
1	T_1	a,b,c,g,h
	T_2	a,d,f
2	T_3	a,b,e,g
	T_4	a,d,f
3	T_5	a,b,d,e
	T_6	a,b,c,d,g
4	T_7	a,b,c
	T_8	a,g

Table 2.1: Simple Transaction Table

in 3, so 3 there is 3 partition in every node one for each batch. Where new batch comes older one is deleted.

2.2 Important data mining Techniques and Procedures

We have already develop our field for discussing some data-mining techniques related to our works. In this section we will try to illustrate different methodologies and their differences with our work in short.In the prior section some important differences between our proposed approach and other related techniques are tried to be given. But it is essential to discuss about some methods that are actually the main motivation of our works. In this Section detail portrayal of those topics have to be discussed.

2.2.1 Data Stream Tree (DS-tree)

DS-Tree approach is one of the most efficient technique that deals with uncertain database. It is based on Fp-growth based tree structure that effectively hold uncertain data and effectively mine the tree to generate all the frequent patterns. Like other tree based approaches DS-Tree works in two steps 1) Tree construction phase 2) Mining frequent patterns from tree.

To gain a better understanding of DS-tree, let us consider the following example.

2.2.1.1 Tree construction (DS-Tree)

DS-Tree is designed for data flow or data stream. The construction of the DS-tree only requires one scan of the streaming data. The tree capturers the contents of transactions in each batch of streaming data. Data stream has dynamic nature, frequencies of items are incessantly influenced by the most recent batches and the removal of old batches.

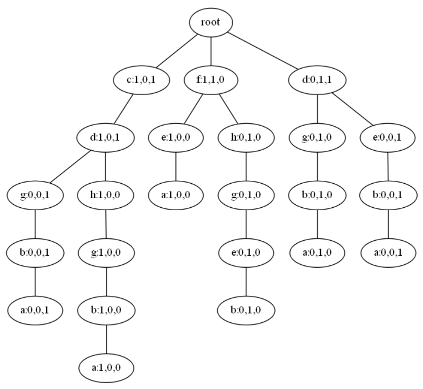


Figure 2.1: A simple tree for capturing data stream

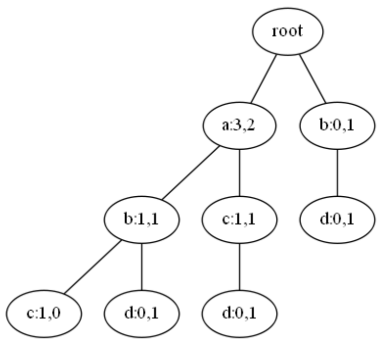
Adorning items in frequency dependent order conduct to swapping that causes merging and splitting when frequencies change[15].

At first In DS-Tree items are arranged according to some canonical order, For example,fig 2.2(A) items can be consistently arranged in lexicographic order. After that items can be arranged according to particular order depending on items properties such as items values or validity. The next issue is updating the frequency information at each tree node when new batches are inserted and old batches are deleted.DS-Tree keeps a list of frequency counts at each node. When a new batch flow in, it is appended to this list at each node frequency count in the current batch. In other words the last entry of the list at node A shows the frequency count of A in the current batch. Whenever next batch comes in, the list is shifted forward. The last entry shifts and becomes the second last entry. At the same time the frequency count assembling to the oldest batch in the window is removed fig 2.2(B).

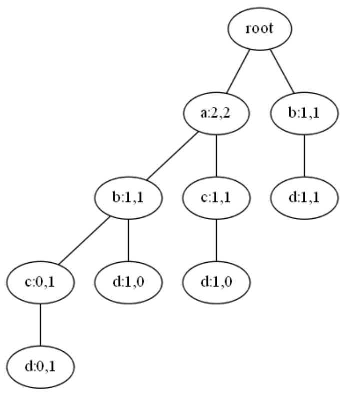
Batch	TID	Transactions
first	T_1	{a,b,c}
	T_2	{a}
	T_3	{a,c}
second	T_4	{a,c,d}
	T_5	{b,d}
	T_6	{a,b,d}
third	T_7	{b,d}
	T_8	{a,b,c,d}
	T_9	{a,c}

Table 2.2: Transaction Table.

Here In fig 2.2 minimum support is 3 and the window size(W) is 2. That indicates only two batches of transactions are kept. When first two batches of transactions flows in, they are inserted into the DS-Tree and frequency counts in a list of W entries at each node is kept . For example the node b: 0,1 indicates that the frequency of b is 0 in the



(A) After 1st two batch insertion.



(B) After 3rd batch insertion.

FIGURE 2.2: DS-Tree after each batch of transactions is added for stream mining.

first batch and 1 in the second batch. When the third batch of stream data flows in, it is inserted in the DS-Tree. The lists frequency counts sifts, the frequency counts for oldest batch are removed.

2.2.1.2 Mining process of DS-Tree

The usual FP-tree based mining process can be applied to the DS-Tree to find all frequent item sets. This is because DS-Tree only captures the W = 2 most recent batches of transactions at that time. If we call the mining process at time 'T', we get frequency itemsets a:4 , a,c :3, a,d: 3, b:4, c:3 and d:5.

2.2.2 Weighted frequent pattern mining over Data streams (WFP-MDS)

Weight frequent pattern becomes an important research topic in pattern mining and data discovery area. Weighted Frequent Pattern Mining over Data Stream (WFPMDS) is one of the important contribution to data mining field [39]. .This approach exploits a pattern growth mining approach to avoid level-wise candidate generation and test problem. Extensive performance analyses show that this technique is very efficient for WFP mining over data stream.

Key concepts of WFPMDS: WFPMDS starts by creating a tree from the data stream using a single pass. Tree construction phase is same as DStree with one exception here items are sorted according to the weight to take advantage of GlobalMax weight.

Definition GlobalMax Weight: The global maximum weight, denoted by GMAXW, is the maximum weight of all the items in the current window.

First challenge of this task is that whether a particular item is frequent or not. It can't be decided immediately as a item may infrequent now but may be frequent in next batch. Second, Weighted is considered here so downward property doesn't hold as a infrequent item may be frequent when it appeared with higher weighted item.

Batch	TID	Transactions
first	T_1	{a,b,c,d,g,h}
	T_2	{a,e,f}
second	T_3	{b,e,f,g,h}
	T_4	{a,d,d,g}
third	T_5	{a,b,d,e}
	T_6	{a,b,c,d,g}
fourth	T_7	{a,b,g}
	T_8	{a,c}

(A) Transaction Stream

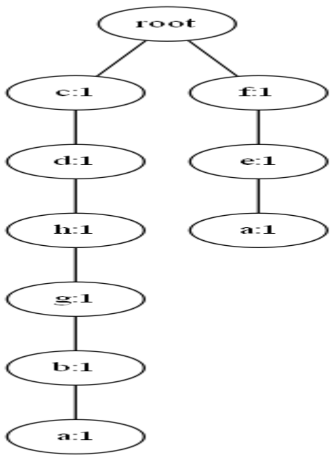
Items	Weight
a	0.6
b	0.5
c	0.2
d	0.35
e	0.5
f	0.3
g	0.4
h	0.38

(B) Weight Table

Table 2.3: Example of a transaction data stream with weight table (WFPMDS).

Item	Weight	Frequency
c	0.2	1
f	0.3	1
d	0.35	1
h	0.38	1
g	0.4	1
e	0.5	1
b	0.5	1
a	0.6	2

(A) Header-Table.

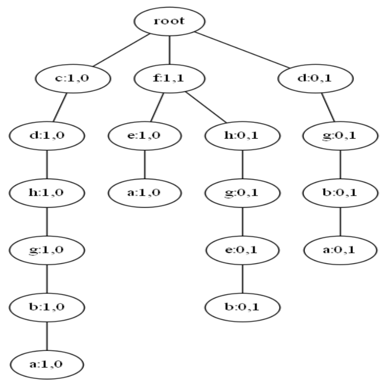


(B) Tree Construction.

Figure 2.3: WFPMDS tree after first batch of transactions is added for stream mining.

Item	Weight	Frequency
c	0.2	1
f	0.3	2
d	0.35	2
h	0.38	2
g	0.4	3
e	0.5	2
b	0.5	3
a	0.6	3

(A) Header-Table.

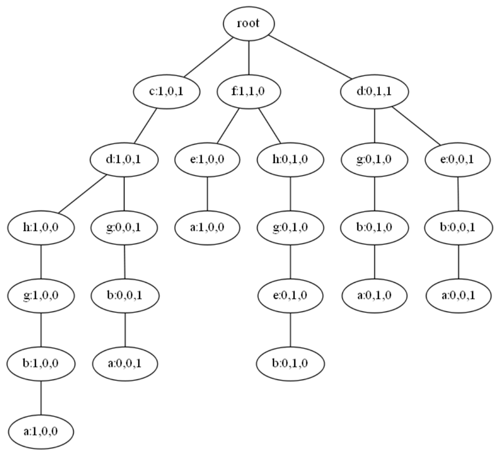


(B) Tree Construction.

FIGURE 2.4: WFPMD tree after second batch of transactions is added for stream mining.

Item	Weight	Frequency
c	0.2	2
f	0.3	2
d	0.35	4
h	0.38	2
g	0.4	4
e	0.5	3
b	0.5	5
a	0.6	5

(A) Header-Table.

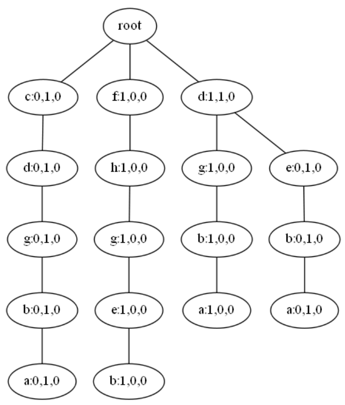


(B) Tree Construction.

FIGURE 2.5: WFPMDS Tree after third batch of transactions is added for stream mining.

Item	Weight	Frequency
c	0.2	1
f	0.3	1
d	0.35	3
h	0.38	1
g	0.4	3
e	0.5	2
b	0.5	4
a	0.6	3

(A) Header-Table.

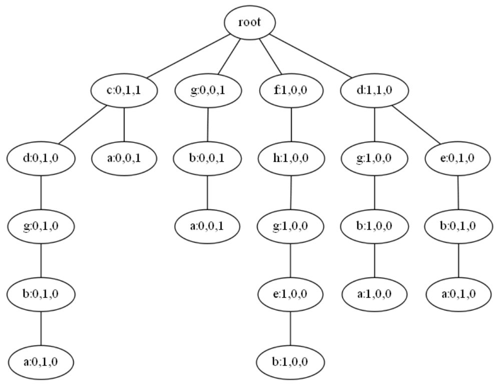


(B) Tree Construction.

FIGURE 2.6: WFPMDS after After Deleting Batch1.

Item	Weight	Frequency
c	0.2	2
f	0.3	1
d	0.35	3
h	0.38	1
g	0.4	4
e	0.5	2
b	0.5	5
a	0.6	5

(A) Header-Table.

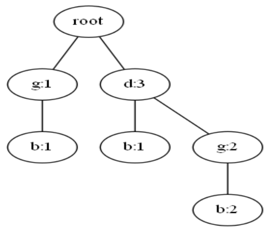


(B) Tree Construction.

FIGURE 2.7: WFPMDS tree and Header-Table After Inserting Batch 4.

Item	Weight	Frequency
d	0.35	3
g	0.4	3
b	0.5	5

(A) Header-Table.

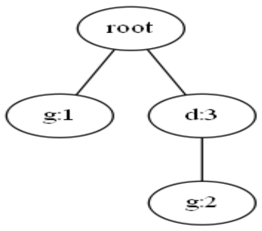


(B) Tree Construction.

FIGURE 2.8: WFPMDS Conditional tree for item 'a'.

Item	Weight	Frequency
d	0.35	3
g	0.4	3

(A) Header-Table.

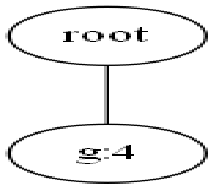


(B) Tree Construction.

FIGURE 2.9: Mining Process of WFPMDS Conditional tree for item-set 'ab'.

Item	Weight	Frequency
g	0.4	3

(A) Header-Table.



(B) Tree Construction.

FIGURE 2.10: Mining Process of WFPMDS Conditional tree for item 'b'.

2.2.2.1 Tree Construction (WFPMDS)

In the process of tree construction, the tree structure is to capture data stream using a single pass. In tree structure maintain header table to keep an item order. A predefined value for batch size and window size are fixed this value may come from users. Every node is built so that it can store as many value as window size. Each node only maintain item-id and frequency information for each batch. Each partition of the node contains the value of each batch. When every partition is filled and new batch come older one is deleted, each value is shifted by one partition and new batch is inserted to new partition.

First, items in the transactions are sorted according to their weight in weight ascending order. Then transactions are inserted batch-wise into the tree. Each node hold the frequency of each item alone with the item itself. Here is example when batch1 is inserted, each node store the frequency of the respected item in the first partition fig 2.3(B). Similarly when second batch come their values are stored in 2nd partition fig 2.4. In the same way batch3 is inserted into tree fig 2.5. When batch4 come as mention earlier batch1 is deleted fig 2.6, batch2 is shifted in the place of batch1. Also batch 3 is shifted in the place of batch 2. After that new batch inserted into the third partition.

2.2.2.2 Mining Process of WFPMDS

Consider mining request has come in the current window. Here for each node, value of a certain node is summed to calculate the value of the node. As weighted pattern does not follow downward closure property, WFPMDS method does not prune item immediately by only looking its weighted support. It uses GMAXW to calculate maximum weighted support of items. The highest probability of an item being frequent is reflected by this value. The item is pruned if the maximum weighted support is less than threshold otherwise it as candidate of frequent pattern.

Now we explain mining process for item a. here figure 2.8 item c and e is pruned as their maximum weighted frequency is less than the threshold limit. But items d,g,b cross the limit of threshold so they are considered as candidate of higher pattern. The actual weighted support of ad = 3*.475 <threshold limit hence pruned

Similarly weighted support of ag = 3*0.5 = 1.5 <threshold so pruned. But for ab = 4*0.55 = 2.2 >threshold hence frequent. Thus mining process is originated to generate all the weighted frequent patterns from data stream.

2.2.3 Dynamic weighted Frequent pattern mining (DWFPM)

Weighted Frequent Pattern Mining over Data Stream (DWFPM) is one of the important contribution to data mining field [40]. DWFPM algorithm is used to prune weighted frequent pattern from Data streams where weights are dynamic. In our real life dynamic weighted pattern is so much demanding concept. By the way this DWFPM approach has significant importance in data-mining as this the first method which can handle dynamic weighted pattern mining effectively.

2.2.3.1 key Ideas

The main challenges of DWFPM approach is to handle the dynamically changing item weights. For generate Dynamic weighted support of a pattern p, researchers introduced called DWsupport(p). This is the value which is actually stored in the tree and use to generate frequent patterns.

Definition of DWsupport(P):

$$DWsupport(P) = \sum_{q=1}^{N} Weight(P) \times Support(P) \ldots\ldots (3)$$

Batch	TID	Transactions	Weight				
1st	T_1	{a,b,d}	a	b	c	d	e
	T_2	{c,d}	0.45	0.9	0.2	0.3	0.5
	T_3	{a,b}					
2nd	T_4	{b}	a	b	c	d	e
	T_5	{b,c,d}	0.6	0.7	0.4	0.5	0.4
	T_6	{c,e}					
3rd	T_7	{a, c, e}	a	b	c	d	e
	T_8	{a}	0.5	0.3	0.7	0.4	0.45
	T_9	{a,c}					

TABLE 2.4: An example of transaction database with dynamic weights.

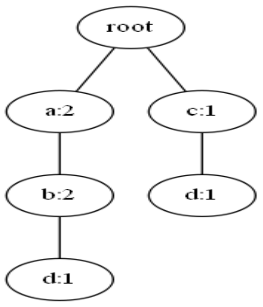
Here (table 2.4) the number of batchesis N, m $\text{Weighted}_j(P)$ and $\text{Support}_j(P)$ are weight and support of in the jth batch . The value of $\text{Weighted}_j(P)$ can be calculated by applying Eq.(3). For example, in the first, second and third batches the DWsupport(P) of pattern “bd” are ((0.09+0.03)/2)*1 =0.6, ((0.07+0.05)/2)*1 = 0.6 and ((0.03+0.04)/2)*0 = 0, respectively (table 2.4). So, the total DWsupport of “bd” is (0.6+ 0.6+0) = 1.2. If the dynamic weighted support is greater than or equal to the minimum threshold, a pattern is called a dynamic weighted frequent pattern. For example, when the minimum threshold is 1.2 then “bd” is a dynamic weighted frequent pattern.

2.2.3.2 Tree construction(DWFPM)

Throughout this section, the construction process of DWFPM tree structure will be described that captures transactions having items containing dynamic weights. Like Fp-tree a header table is sustained.The item id is the first value in the header table .With in the header table in a batch by batch fashion the frequency and weight fact associated with an item is kept. An items id and its batch by batch frequency information are contained by the tree nodes. Consider the database shown in (table 2.4). First the items are sorted according to lexicographical order in figure 2.11(A), and then they are inserted into the tree, after pursuing a transaction from the database. Figure 2.11(B) the transactions from batch 1 is shown in the tree after capturing .After the first batch insertion in both the header table and tree Separate transaction count information must be kept for each batch.In the header table only weight information is kept. Figure 2.12(B) presents the tree after inserting the first and second batches of transaction. We can easily find which transactions have occurred in which batch, as frequency information is kept separately. For example, from figure 2.13(A) the batch numbers of transactions can be easily determined. Frequency of “c,d” are 1 and 2 respectively in first batch. Figure 2.13(B) shows the tree after inserting the first, second and third batches.

Item	Weight	Frequency
a	0.45	2
b	0.9	2
c	0.2	1
d	0.3	2

(A) Header-Table.

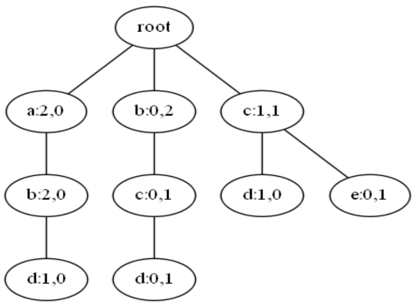


(B) Tree Construction.

Figure 2.11: DWFPM after after inserting first Batch.

Item	Weight	Frequency
a	0.45, 0.6	2, 0
b	0.9, 0.7	2, 2
c	0.2, 0.4	1, 2
d	0.3, 0.5	2, 1
e	0.5, 0.4	0, 1

(A) Header-Table.

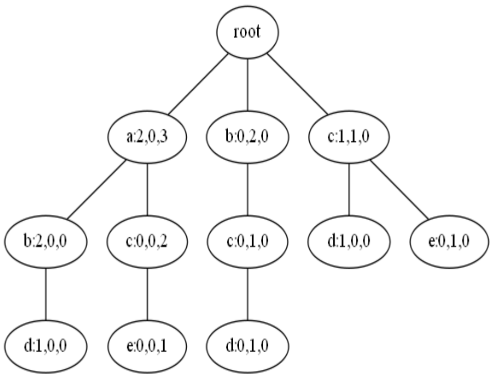


(B) Tree Construction.

Figure 2.12: DWFPM after after inserting first and second Batch.

Item	Weight	Frequency
a	0.45, 0.6, 0.5	2, 0, 3
b	0.9, 0.7, 0.3	2, 2, 0
c	0.2, 0.4, 0.7	1, 2, 2
d	0.3, 0.5, 0.4	2, 1, 0
e	0.5, 0.4, 0.45	0, 1, 1

(A) Header-Table.



(B) Tree Construction.

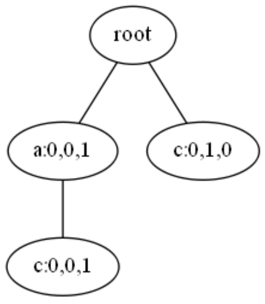
FIGURE 2.13: DWFPM after inserting first, second and third Batch.

2.2.3.3 Mining process of DWFPM

All branches of prefixing that item that item are taken with the frequency value of that item,when a prefix tree is created for the particular item. After that deletes all frequently itemized nodes, the conditional tree is constructed from the prefix tree. As weighted pattern does not follow downward closure property, DWFPM method does not prune item immediately by only looking its weighted support. It uses GMAXW to calculate maximum weighted support of items. The maximum probability of an item being frequent is reflected by this value. The item is pruned if the maximum weighted support is less than threshold otherwise it as candidate of frequent pattern. In this case the maximum weight calculated within every weight in all of the batches. For example, in table 2.4 , global maximum weight of item “b” has 0.9. this is referred as GMAXW. When they are doing the mining operation for a particular item then The local maximum weight is needed which is not always equal to GMAXW. Another example , in table 2.4 , item “e” does not occur with “b” and “d” . For this reason , the prefix tree of “e” only contains items “a” and “c”, in the mining operation. Here the local highest weight can sustain the downward closer property so GMAXW is not used. Among all the item “c”, “a” and “e”, the local maximum weight for “e” is 0.7. This local maximum weight is referred to as LMAXW. The probability of a pattern for becoming a candidate is reduced by using LMAXW instead of GMAXW. Apprehend the database showed in table 2.4, the tree is build for that database in figure 2.13(B), and the minimum threshold is 1.2. Here GMAXW is 0.9. frequency list of the dynamic weighted is <a:4.5, b:3.6, c: 4.5, d:2.7, e:1.8>, after multiplying the total frequency. As a result all items are candidate for a single element. In bottom up fashion the prefix and conditional trees are constructed and mined the frequent patterns of dynamic weighted. First of all the prefix tree of the bottom –most item “e” is generated by taking all of the prefixing item of branches “e” in figure 2.14 (B). To create a conditional tree, the nodes are removed from the prefix tree, which cannot be a candidate. For item “e” , LMAXW is 0.7. The frequency list of dynamic weighted <a:0.6, c:1.4 >is generated after mutinying the frequencies in the header table with LMAXW. Since item “a” has a low frequency of dynamic weighted with item “e”, it has to be removed to obtain the conditional tree of “e”. The conditional tree of item “e”is shown in figure 2.15. At this point the candidate patterns “ce” and “e” are created. The prefix tree of item “d” is generated in figure 2.16. LMAXW is 0.09 here, the dynamic weighted list is <: 0.9, b: 1.8, c:1.8>. The conditional tree of item “d” is generated by deleting item “a” from prefix tree figure 2.17(B). The conditional patterns “bd”, “cd”, and “d” are created. In figure 2.18(B) the prefix tree of pattern “dc” is shown. The dynamic weighted frequency list is <0.9>. So, no conditional tree is generated. In figure 2.19 the prefix tree of item “c” is created. Here LMAXW is 0.9 . The conditional tree of item “c” is generated by deleting item “b” from prefix tree figure 2.20(B). At this point the candidate patterns “ac” and “c” are created. In figure 2.21(B) the prefix tree of item “b” is generated. The dynamic weighted frequency list is <“a:1.8”>and LMAXW is 0.9. It is also the conditional tree for “b”. Patterns “ab” and “b” are created . The top most item “a” generates last conditional tree. The candidate must test all patterns, including the actual dynamic weighted frequency using Eq (3). Table 2.5 shows those calculations.

Weight	Frequency
a	0, 0, 1
c	0, 1, 1

(A) Header-Table.

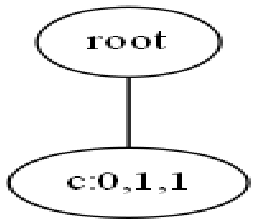


(B) Tree Construction.

FIGURE 2.14: DWFPM Prefix tree of item “e”.

Weight	Frequency
c	0, 1, 1

(A) Header-Table.

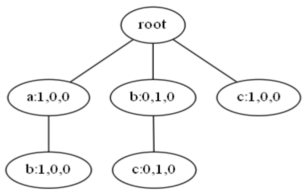


(B) Tree Construction.

FIGURE 2.15: DWFPM conditional tree of item “e”.

Weight	Frequency
a	1, 0, 0
b	1, 1, 0
c	1, 1, 0

(A) Header-Table.

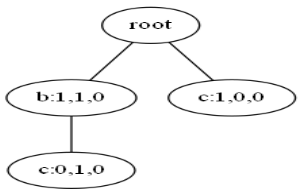


(B) Tree Construction.

Figure 2.16: DWFPM Prefix Tree of item “d”.

Weight	Frequency
b	1, 1, 0
c	1, 1, 0

(A) Header-Table.

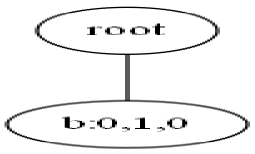


(B) Tree Construction.

Figure 2.17: DWFPM conditional Tree of item “d”.

Weight	Frequency
b	0, 1, 0

(A) Header-Table.

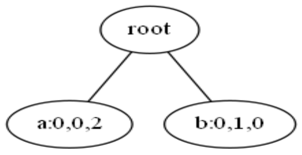


(B) Tree Construction.

Figure 2.18: DWFPM Prefix Tree of item “dc”.

Weight	Frequency
a	0, 0, 2
b	0, 1, 0

(A) Header-Table.



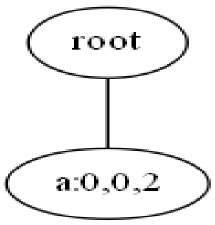
(B) Tree Construction.

Figure 2.19: DWFPM Prefix Tree of item “c”.

Several tree based algorithms have been proposed for mining uncertain database. Carson Kai-sang Leung propose DS Tree: Data Stream Tree to mine database for data streams [2]. In that paper authors only implement sliding window method for capture data streams. But the problem is that his DS Tree only capture certain items and able to generate all the frequent patterns not interesting one by giving effective and significant importance to items. They did not apply weight on the items of the database. Moreover in DS tree mining procedures algorithms can generate many patterns which may not interesting to the users depending on the applications. But we our proposed approach can handle data streams and generate more interesting patterns by applying the weights

Weight	Frequency
a	0, 0, 2

(A) Header-Table.

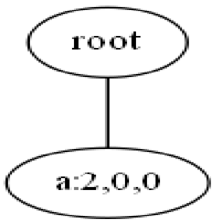


(B) Tree Construction.

Figure 2.20: DWFPM Conditional Tree of item "c".

Weight	Frequency
a	2, 0, 0

(A) Header-Table.



(B) Tree Construction.

Figure 2.21: DWFPM Prefix and Conditional Tree of item "b".

No.	Candidate Patterns	DWsupport Calculation	Result
1	ce:0,1,1	$(((0.4{+}0.4)/2)\times1) + (((0.7 + 0.45)/2) \times 1)$ $= 0.4{+}0.575 = 0.975$	Prunned
2	e:0,1,1	$(0.4 \times1) + (0.45 \times 1) = 0.85$	Prunned
3	cd:1,1,0	$(((0.2{+}0.3)/2)\times1) + (((0.4 + 0.5)/2) \times 1)$ $= 0.25{+}0.45 = 0.7$	Prunned
4	bd:1,1,0	$(((0.9{+}0.3)/2)\times1) + (((0.7 + 0.5)/2) \times 1)$ $= 0.6{+}0.6 = 1.2$	Pass
5	d:2,1,0	$(0.3 \times2) + (0.5 \times 1) = 0.1.1$	Prunned
6	ac:0,0,2	$((0.5 + 0.7)/2) \times2 = 1.2$	Pass
7	c:1,2,2	$0.2\times1 + 0.4 \times 2 + 0.7 \times 2 = 2.4$	Pass
8	ab:2,0,0	$((0.9{+}0.45)/2)\times2 = 1.35$	Pass
9	b:2,2,0	$0.9\times2 + 0.7 \times 2 = 3.2$	Pass
10	a:2,0,3	$0.45\times2 + 0.5 \times 3 = 2.4$	pass

Table 2.5: DW support calculations of the candidate patterns.

of items. Chowdhury Farhan Ahmed propose a novel tree mining which is able to capture all the data streams and generate interesting patterns [1]. He apply weights on the items of this datasets. But this prefix tree e was not enough compact. In our proposed work we want to improve his work by compacting the generated prefix tree for saving memory space.

2.3 Related Works and differences with our Approaches

Related Works and differences with our Approaches Several tree based algorithms have been proposed for mining uncertain database. Carson Kai-sang Leung propose DS Tree: Data Stream Tree to mine database for data streams[15]. In that paper authors only implement sliding window method for capture data streams. But the problem is that his DS Tree only capture certain items and able to generate all the frequent patterns not interesting one by giving effective and significant importance to items. They did not apply weight on the items of the database. Moreover in DS tree mining procedures algorithms can generate many patterns which may not interesting to the users depending on the applications. But we our proposed approach can handle data streams and generate more interesting patterns by applying the weights of items. Chowdhury Farhan Ahmed propose a novel tree mining which is able to capture the data streams and generate interesting patterns[39]. He apply weights on the items of this datasets. But this prefix tree e was not enough compact. In our proposed work we want to improve his work by compacting the generated prefix tree for saving memory space.

Chapter 3

Proposed Approach

Data mining is one of the most demanding sector in computer science which discover hidden patterns from large data sets. For finding frequent patterns from different types of data sets, there are many researches, methodologies, tools, procedures have been developed. In this thesis is also accomplish such a novel procedure that helps to explore new branches of data mining sector. In this chapter we explore our proposed word in great details.

3.1 Overview

In our thesis work we basically league tree three important and most demanding topics of data mining those re uncertain data, data streams and weighted frequent patterns that builds new methodology focusing recent demand. Our target is simple but useful. Our proposed work helps to perfectly capture data streams where data is uncertain and introduces a mining techniques that ensure only find interesting patterns depending on user needs. There are many example where data are uncertain like medical database and student data sets are growing faster in size as a result cannot afford to store whole the data stream and mine them. It is unnecessary to generate all the frequent patterns of that huge data stream which may also be huge in size and can be generate without considering user needs. Generating only those patterns that actually has influence on application is more efficient and less time consuming. Several researches proposed many approach to find patterns from databases and also from data streams. But we want to improve our previous work by reducing consuming time and memory size which is appropriate for large data sets.

3.2 Our goals

In this thesis book we propose a new novel approach of capturing uncertain data streams and finding interesting patterns. We named our approach Compact Pattern Tree for Static Weights (CPTSW)-growth approach. In later sections we will explain our CPTSW-growth approach with proper example.

3.3 preliminaries

a data stream could have infinite number of transactions. A sets of transactions are contained by a batch of transactions. Table 2.3 shows an example of transaction data stream divided into four batches with equal length. A fixed number of non-overlapping batches can compose a Window . in our example data stream , we assume that one window contains tree batches of transactions. Here, batch1,batch 2 and batch3 is contains by Window 1 . Similarly batch2, batch3 and batch 4 is contains by Window 2.

By multiplying its support in W with its weight, the weighted support of a pattern P can be calculated over a Window W. Therefore, pattern P is weighted frequent pattern in W if its weighted support is greater than or equal to the minimum threshold. For example, “ab” is a weighted frequent pattern in window2 if the minimum weighted threshold is 2.0. 1,2 and 1 respectively its frequency in batch2 , batch3 , batch4. So the total frequency is 4 in Window2 . Its weighted support in Window2 is 4* 0.55 =2.2, which is greater that minimum support 2.

Batch	TID	Transactions
first	T_1	{a,b,c,d,g,h}
	T_2	{a,e,f}
second	T_3	{b,e,f,g,h}
	T_4	{a,d,d,g}
third	T_5	{a,b,d,e}
	T_6	{a,b,c,d,g}
fourth	T_7	{a,b,g}
	T_8	{a,c}

(A) Transaction Stream

Items	Weight
a	0.6
b	0.5
c	0.2
d	0.35
e	0.5
f	0.3
g	0.4
h	0.38

(B) Weight

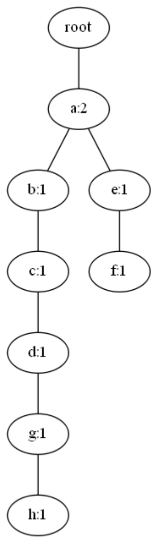
FIGURE 3.1: Example of a transaction data stream with weight table.

3.4 Tree construction

To build transactions having items with static weights, we construct a compact pattern tree for static weights (CPTSW). Our CPTSW builds a compact frequency descending tree with a single database scan. At first transaction of the first batch are inserted into CPTFW tree according to lexicographic order. After inserting a batch of transactions the CPTSW tree is dynamically restructured by frequency descending order. In summary CPTSW tree can be constructed in two phases: Insert phase: Scan all the transaction of a batch. Transactions are inserted into the tree according to the current item order and update the Item list (I-list) frequency count of the respective items . Restructuring phase: rearrange the I-list according to the frequency descending order of the items. Then rearrange the tree according to newly arrange I-list. The tree construction is starts with the insertion phase. The first insertion phase is begin with first transaction of the first batch according to lexicographic order. After inserting all the transaction of first batch the tree will be restructured. The tree will be restructured with path adjusting method. Recursive swapping of the adjacent node in the path is applied until it achieves the new sorted structure. Thus bubble sort method is used for swapping two nodes. In FP-tree the frequency count of a node cannot be greater than the frequency count of its parent node. For maintaining that property, Path adjusting method inserts a new node of the same name as a sibling of the parent node in the tree when the parent node needs to be exchanged with any child node which has smaller frequency count value. After swapping, two siblings contains same items, then they should be merged. Insertion and restructuring are conducted substitute until all transactions of all batches are inserted and restructured the tree in frequency descending order in batch by batch fashion. The pseudo-Code for our proposed work is shown in Algorithm 1.

Consider the example database of fig 3.1. At first the items of the first batch are inserted according to the lexicographic order Fig 3.2 (A). They are already in frequency descending order so no need to restructure this tree. Fig 3.2 (B) shows the tree after inserting the transaction of the second batch based on the item order which is achieved by inserting the first batch. Then the items are sorted in frequency descending order based on their current frequency which includes their frequency count of both the batches. Fig 3.2 (C) shows the tree after restructuring. It can be easily discovered which transactions have occurred in which batch as frequency information for each batch is kept separately in each node in the tree. The fig 3.2 (D) shows the tree structure after inserting the transactions of the third batch. Fig 3.2(G) is our final CPTSW tree structure which is achieved by inserting and restructuring all the transaction of all the batches. Our propose work CPTSW-growth tree structure has the following properties:

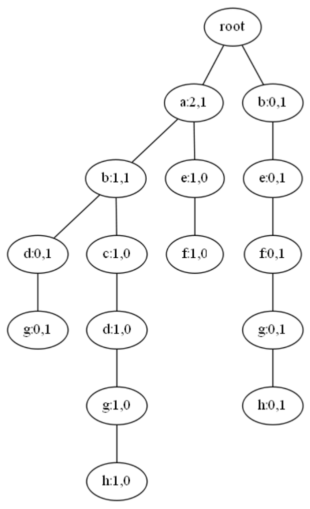
Items	Weight	I-list
a	0.6	2
b	0.5	1
c	0.2	1
d	0.35	1
e	0.5	1
f	0.3	1
g	0.4	1
h	0.38	1



-

(A) After inserting 1st batch.

Items	Weight	I-list
a	0.6	2,1
b	0.5	1,2
c	0.2	1,0
d	0.35	1,1
e	0.5	1,1
f	0.3	1,1
g	0.4	1,2
h	0.38	1,1



-

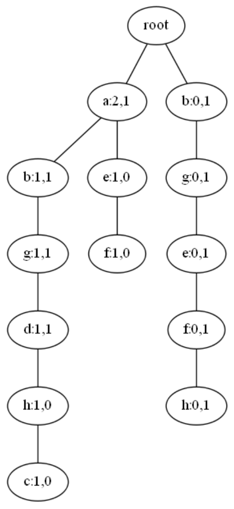
(B) After inserting 2nd batch.

– **Property 1:** The items in the tree are sorted according to the frequency descending order.

– **Property 2:** The tree structure can be constructed in single database scan.

– **Property 3:** the total frequency count of any node in the tree is greater than or equal to the sum of total frequency counts of the nodes children.

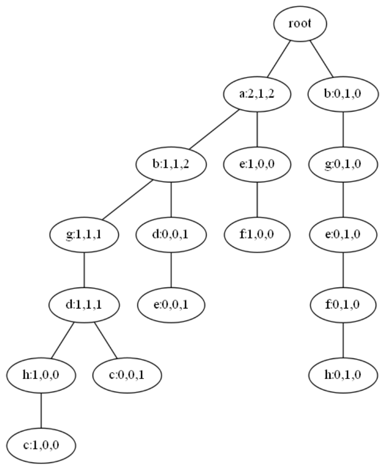
Items	Weight	I-list
a	0.6	2,1
b	0.5	1,2
g	0.4	1,2
d	0.35	1,1
e	0.5	1,1
f	0.3	1,1
h	0.38	1,1
c	0.2	1,0



-

(c) After restructuring 2nd batch.

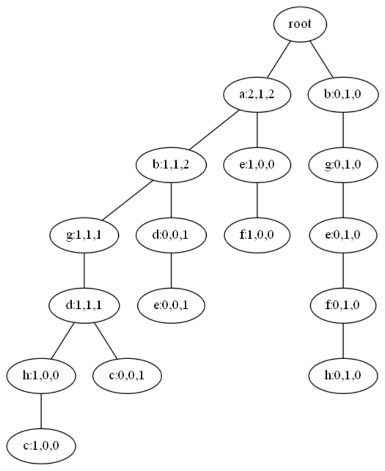
Items	Weight	I-list
a	0.6	2,1,2
b	0.5	1,2,2
g	0.4	1,2,1
d	0.35	1,1,2
e	0.5	1,1,1
f	0.3	1,1,0
h	0.38	1,1,0
c	0.2	1,0,1



-

(d) After inserting 3rd batch.

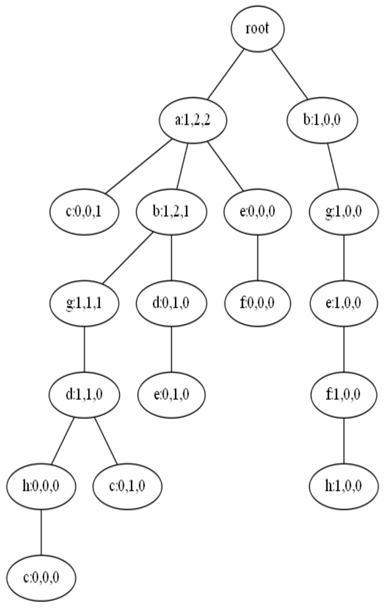
Items	Weight	I-list
a	0.6	2,1,2
b	0.5	1,2,2
g	0.4	1,2,1
d	0.35	1,1,2
e	0.5	1,1,1
f	0.3	1,1,0
h	0.38	1,1,0
c	0.2	1,0,1



-

(e) After restructuring 3rd batch.

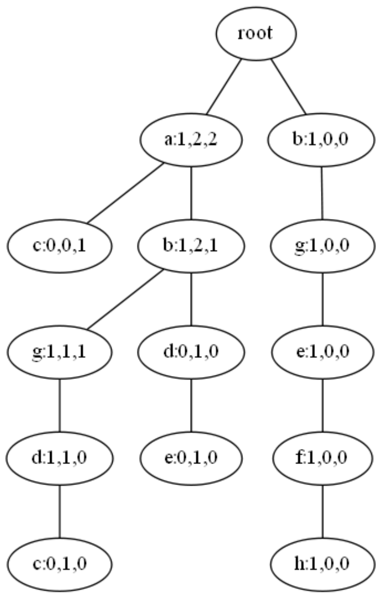
Items	Weight	I-list
a	0.6	1,2,2
b	0.5	2,2,1
g	0.4	2,1,1
d	0.35	1,2,0
e	0.5	1,1,0
f	0.3	1,0,0
h	0.38	1,0,0
c	0.2	0,1,1



-

(F) After inserting 4th batch.

Items	Weight	I-list
a	0.6	1,2,2
b	0.5	2,2,1
g	0.4	2,1,1
d	0.35	1,2,0
e	0.5	1,1,0
c	0.2	0,1,1
f	0.3	1,0,0
h	0.38	1,0,0



-

(G) After restructuring 4nd batch.

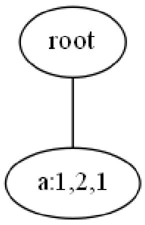
FIGURE 3.-2: Procedure of Tree Construction and Restructuring.

3.5 Mining process

In this section, we describe the mining process of our proposed CPTSW-growth technique. The weighted frequency of an item set does not have the downward closer property and to utilize this property global maximum weight have to use and this is the main challenges in this area. GMAXW denoted global maximum weight which means the maximum weight of all items in the global database. In our example fig 3.2 (G) the “a” has the global maximum weight 0.6 for Window1 and Window2. The local maximum weight, denoted by LMAXW, is needed when we are doing the mining operation for a particular item. Our tree is in weight descending order, so we get advantage in the top-down mining operation. The local maximum weight LMAXW is needed while doing the mining operation for a particular item, it is not always equal to GMAXW.

As our CPTSW tree is sorted according to frequency descending order, LMAXW could be anywhere for a particular item. We start our mining operation from the top-most item of the CPTSW tree structure. So for this case, LMAXW is the weight of the first item for sure. After that for the second item, compare its weight with the previous LMAXW and consider the larger one as the current LMAXW. By moving in this way, LMAXW calculation for each time can be saved. We consider the database presented in table 3.1, the tree constructed for that particular database in fig 3.2(G) and the minimum threshold is 1.3. Here GMAXW is 0.6 . After multiplying GMAXW with the total frequency of each item, we get a: 0.6* 5= 3.0, b: 0.6 * 5= 2.5 , g: 0.6 * 4=2.4 , d :0.6*3=1.8 , e: 0.6*2 =1.2, c: 2*0.6 =1.2 ,f:1*0.6 =.6 , h= 0.6*1= 0.6 . so “a” , “b” , “g” ,”d” items are single element candidates. We start our mining process with the top-most item “a” CPTSW tree. For “a” LMAXW is 0.6 , frequency of “a” is 1+2+2=5 . By multiplying the frequency of “a” with LMAXW of “a” we get 0.6*5= 3.0 , which is greater than minimum support threshold 1.3. So, single element pattern “a” is generated. After that, we consider item “b “as it is the second top-most item in fig 3.2(G) . So prefix tree of “b” is created by taking all branches prefixing item “b” as shown in fig 3.3(A) . The nodes that cannot be candidate patterns must be deleted from the prefix tree for creating conditional tree. For item “g” , LMAXW is 0.4 . After multiplying the frequency of the item “g” in the header table shown in fig 3.1 , We get 4*0.4 =1.6 . As the value is greater than the minimum support threshold value, that is 1.3, so no node should be deleted from the prefix tree. From the fig 3.3(B) the frequency of “b” is 4 , LMAXW of “b” is 5 . we get bg= 0.5 *4 =2.0 which is greter than minimum threshold . From fig 3.3(C) frequency of “a” is 3 and LMAXW is 0.6, so we get abg: 3* 0.6 = 1.8 , which is greater than 1.3. So, the candidate patterns “abg” and “bg” are generated at this point.

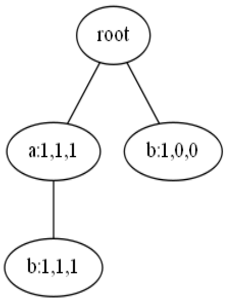
Items	I-list
a	1,2,1



-

(A) Conditional and prefix tree for 'b'

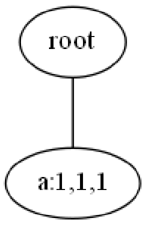
Items	I-list
a	1,1,1
b	2,1,1



-

(B) Conditional and prefix tree for 'g'

Items	I-list
a	1,1,1



-

(C) Conditional and prefix tree for 'bg'

Figure 3.-1: Mining operation

From fig 3.1we get frequency of "d" is 1+2+0=3 and LMAXW of "d" is 0.35 , so d: 3* 0.35= 1.05 which is less than 1.3 that is our minimum threshold. So " d" is not candidate item. Thus all the items in I-list of fig 3.1 is conducted for finding out all the candidate patterns of our example database. The pseudo code of the mining procedure is illustrated in Algorithm 2 and operations are shown in fig 4 .

3.6 Analysis of CPTSW-growth algorithm

working principle of CPTSW-growth approach with proper example was discussed in great details in the previous sections. now details description of the algorithm are explored in this section.

Algorithm 1 shows that how to construct our proposed tree. Tree is constructed by batch by batch. After getting full batch of transactoins then this batch of information will be inserted into the tree. But Every time getting new batch of transactions then deleted will be the oldest batch of the tree. Every item of the oldest batch and their number of frequency will be deleted from the HeaderTalbe. New inserted batch and their frequency count of each items are updating to the headertable. then applying path adjusting method function. This function check total number of frequency of every item of the tree. If any count of parent item is less than their children. Then these two node will be exchanged and all children will be deleted from children node and these children will be assigned in the parent parent node. Now This parent will be children of that children node. So now children in parent and parent is children. This procedure will be continued until all node not will be decreasing order. When swap of the two nodes then parent of the frequency count of parent node is deleted from the frequency count of children node. If frequency count of parent node is not equal to zero then this node should be exist otherwise is deleted.

Then checking every item of the tree branch by branch. If a parent node has same children then between two children will be merged.e Now rest of the children of this children items will be the same parent node.

Another important part of mining procedure. This procedure will be bottom up approach. At first we take the GMAXW as the maximum weight from the all items. Then test the every item is candidate or not. If this item is candidate then check we take this item and finding the prefix tree with the recursive algorithm. Then we take the LMAXW as the maximum value of this current pattern. The next item of the prefix tree is testing, this pattern is candidate or not. A pattern is passed when we calculate the minimum support of this candidate pattern. If minimum weighted support is greater than or equal the minimum weighted support then this pattern will be passed otherwise will be prunned.

Algorithm 1 Tree restructured method

```
procedure TREE_RESTRUCTURE(root_node, weighted_table, HeaderTable)
    Tree_traverse(root_node, HeaderTablee)
    function TREE_TRAVERSE(root, HeaderTable)
        if $len(root.children()) \geq 1$ then
            for each $child \in root.children$ do
                if $HeaderTable[child.name] > HeaderTable[root.name]$ then
                    Exchange_node(root, child)                  ▷ call exchange function
                    Tree_traverse(root.parent, HeaderTable)     ▷ recursive call
                else
                    Tree_traverse(child, HeaderTable)
    function EXCHANEG_NODE($X, Y$)
        $Z \leftarrow new\_node$
        $Z.name = Y.name$
        $Z.parent = X.parent$
        $X.delete\_all\_children()$
        $Z.children \leftarrow X$
        if $len(Y.children) \neq 0$ then
            $Z.children \leftarrow Y.children$
            $Y.delete\_all\_children()$
        $Z.window = Y.window$
        $X.window -= Y.window$
        Delete(Y)
        if $X.count == 0$ then
            Delete(X)
    Merge_node(root_node)                   ▷ now call Marge_node() function
```

Algorithm 2 Merge siblings

```
procedure MERGE_NODE(root)
    for each $X \in root.children$ do
        for each $Y \in root.children$ do
            if $X \neq Y$ & $X.name == Y.name$ then
                $X.window = X.window + Y.window$
                $X.children = X.children + Y.children$
                $Y \rightarrow False$
                Delete(Y)
    for each $X \in root.children$ do
        if $X = True$ then
            Merge_node(X)                   ▷ recursive call
```

Algorithm 3 Mining Procedure

```
1: input: weightedTable, HeaderTable, minSupport, batchSize, windowSize and root
   of the constructed tree
2: CH = new conditional header table
3: rPTree = new root of the prefix tree
4: procedure Minig_tree(root, prefixItemset, candidateItemset, HeaderTable)
5:     $X \leftarrow HeaderTable.keys$
6:     $X \leftarrow descendingsort(X, X \leftarrow HeaderTable.counts)$
7:     $GMAXW \leftarrow max(weightedTable)$
8:     for each $v \in X$ do
9:         if $freq(v) * GMAXW >= \delta$ then
10:            $newItemset = prefixItemset$
11:            $newItemset.add(v)$
12:            $prefixItemset.append(newItemset)$
13:            Call testCandidate($newItemset$)           ▷ to check prunned or passed!
14:            $conditional\_pattern\_base$ ←findPrefixPath($v$)
15:            $CH, rPTree \leftarrow prefixTree(conditional\_pattern\_base)$ ▷ Call prefixTree
16:            Call Minig_tree($rPTree, newItemset, prefixItemset, CH$)    ▷ recursive
```

Chapter 4

Experimental Result

To evaluate the performance of our proposed Method we use several datasets which are commonly used by data-mining scientists to determine the performance of their techniques. In this chapter proper description of different datasets followed by the performance of CPTSW-growth approach will be given. We also compare the performance of our approach to the WFPMDS-growth approach to our works in order to clarify the advantages of CPTSW-growth as our target is to improve WFPMDS-growth approach. We present here comparison in three criteria:

1. run time optimization
2. candidate pattern generation reduction
3. memory requirement efficiency

4.1 Datasets

Data mining is all about discovering knowledge from raw data that is extracting exactly those useful information that real matters to different types of applications and transforming data or datasets to evaluate the performance of our works and indicate the usefulness of our proposed method in order to finding interesting patters from data. In this work several real life for example chess, Mushroom datasets and synthesis data set likes T1014D100K are used to measure the performance of CPTSW-growth. Details of each datasets are discussed in the following sections.

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

Table 4.1: Mushroom Dataset.

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.

Table 4.1 shows the basic characteristics of mushroom data which reflects that data in Mushroom data sets is multivariate that is analysis is analysis is based on more than two variables per observation. Attribute is categorical in nature implies each variable has more than one category but there is no intrinsic ordering to the categories. Description of attributes are given bellow.

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

- Cap-shape :conical=c ,bell=b, convex =x, knobbed=k ,smooth=s, sunken=s,flat =f,
- Cap-surface : fibrous =f ,grooves =g, ,scaly=y,
- Cap-colour: gray =g , green =r, buff =b , brown=n, cinnamon=c,pink =p, purple =u , red =e , white=w
- Brises : no=f,bruises=t
- Odor: spicy=s musty =m,anise=I,almond=a, foul=f , creosote =c, fishy=y, none=n , pungent =p
- Gill-attachment : notched=n,descending =d, attached=a, free=f
- Gill-spacing : close =c, crowded =w , distant =d

- Gill-size: yellow=y, narrow=n, broad=b,
- Gill-color: brown=n ,black =k, buff=b , gray =g,chocolate=h , green =r , orange =o, purple=u, pink =p,red=r , white=w
- Stalk-root :club =c, bulbous =b , cup=u, equal=e, missing=?, rhizomorphs=z, rhizomorphs=z ,rooted=r, Stak-surface-above-ring : scaly=y ,silky=k,fibrous=f, smooth=s
- Stalk-surface-below-ring: silky=k ,smooth=s,fibrous=f ,scaly=y
- Stalk-color-above-ring: buff=b,pink=p,brown=n, cinnamon=c, orange=o, red=e,white=w ,gray =g,
- Stalk-color-below –ring: pink =p,brown =n , buff =b,pink=p,brown=n,gray=g , , red=e, white=w, cinnamon=c, orange=o
- Veil-type: universal=u,partial=p
- Veil-color: orange =o,yellow=y , brown=n, white =w
- Ring-number : one=o ,none=n,two=t
- Ring-type : evanescent =e, cobwebby=c, flaring=f, large=l,sheathing =s , pendant =p, zone=z,none=n,
- Spore-printcolor:black=k, buff=b, brown=n, chocolate=h, green=r, orange=o,white=w,purple=u
- Population : clusterd=c, abundant=a, numerous=n, scattered=s, several=v,solitary=y ,scattered=s,
- Habital: leave=l, meadows=m,,woods=d grasses, paths=p, waste=w, urban=u,

Missing attributes value 2480 of than denoted by"?" . class description: edible 4208 that is 51 % and poisonous 3916 that is 49%

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

Table 4.2: Chess Dataset.

4.1.2 Chess

Chees is a real life dataset built by domain theories to produce the lawful moves of chess games is one of the widely used dataset by data scientists to evaluate the performance of their methods. There are several domain theory of chess game such as Employs a geometric presentation for status ,with every square designed by X,Y coordinate and square calculated by vectors. Propagates lawful moves by 1st generating pseudo moves then extracting those that outcome of the moving player who is checked . Some general information of chess dataset is given in Table 4.2.

4.1.3 General Characteristics of Dataset

Some well-known datasets like T1014D100k , Kosarak etc. are also used to test CPTSW-growth algorithm . T1014D100k is produced using the creator of IBM Almaden Quest research group is sparse in nature.original datasets are lebeled and unweighted. Some important characteristics of datasets that used in our methods to be tested is given in Table 4.3.

Data Set	No. of Trans.	No. of distinct value(D)	Avg Trans length(A)	Dense/Sparse Characteristic ratio R= (A/D)*100
Mushroom	8124	119	23	19.327
Chess	3196	75	37	49.33
Kosarak	990002	41270	8.1	.0196
T101D100K	100000	870	10.1	1.16

Table 4.3: Characteristics of Dataset.

4.2 Environment setup

all the analysis is done with same environment setup and software as we can realise the real differences between the compared algorithm. all the software are stopped at the time of mining so that we can get better performance and get the actual run-time of our algorithm.

Environment setup for performance analysis is given in table 4.4.

CPU	Intel Core i5 4th Generation (4590)
RAM	8GB DDR3
OS	Windows 10 64-bit
Hard disk	54rpm Western Digital Hardisk
Language	Python,version: 3.7.3
Exwcution softweare	Anaconda Notebook, version: 4.7.10
Graphics card	2GB AMD saphire graphics card

Table 4.4: Environment Setup

4.3 Experimental Result Analysis

All the Data that are described ,no weight associate with each item are picked up from FIMI repository, UCI machine learning repository. So first of all some sorts of preprocessing have to be performed on dataset before analysis the result like calculating probabilistic value for items, generating weight so that only application intended patterns can be mined. several techniques that deals unstable data, generate randomly probabilistic value for items. But we follow probabilistic distribution to generate uncertain value depending on the demand of application. exploring real life status patterns those frequency is average which is more wanted rather than highly frequent patterns or which patterns those are just cross the frequent limit. So we follow normal distribution to engage probability of data items by giving extreme probability to that items which frequency count is average and engage lower probability to less frequent data items.Following normal distribution experimental weight of the items also generate randomly . We perform analysis of our algorithm in terms of run-time and memory consumed by our proposed algorithm.
For performance evaluation we are considering three things:

- run-time analysis with different thresholds
- memory consumption
- patterns generation count

4.3.1 Effect of Run-time variation with different thresholds

Performing runtime analysis of our approch we conducted several experiment on data set T1014D100K , Chess,Kosarak , Mashroom. Thresholds or minimum support is the most important concept in frequent pattern mining which generate variation in the total

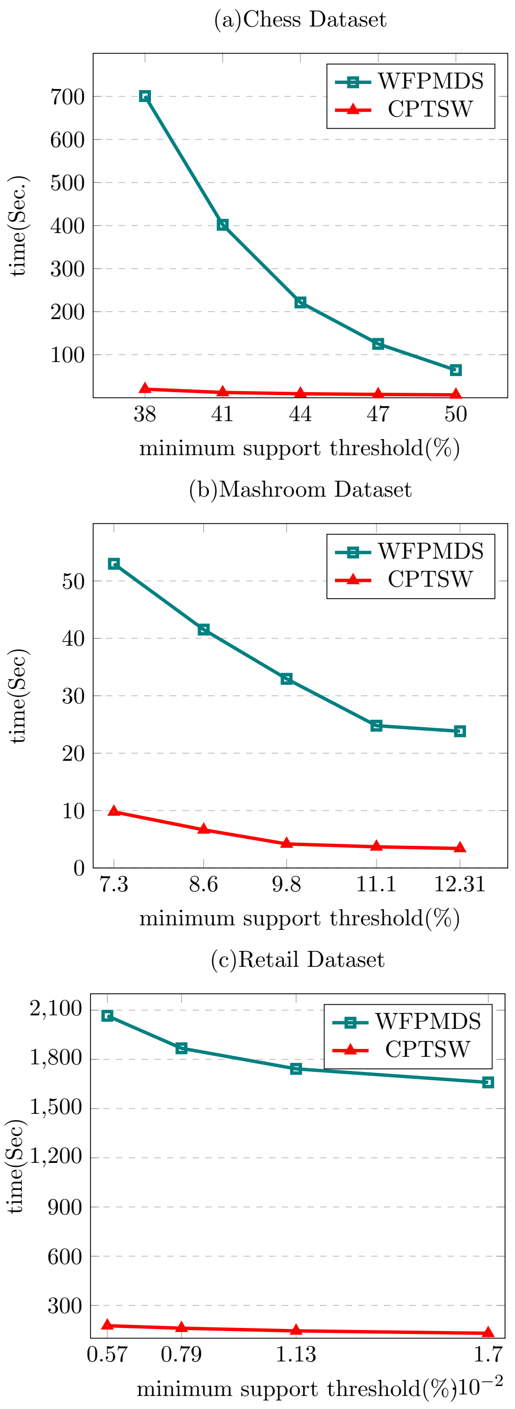


FIGURE 4.1: Runtime Analysis: CPTSW

count of candidate patterns. With the changes of minimum thresholds the runtime varies as the generated patterns varies. To show the effect of run-time with different thresholds we choose Chess , Mashroom and Retail datasets which are more dense in nature.

Chess dataset has 3196 transactions with 75 distinct items. Figure 4.1 (a) shows the effect of runtime with different thresholds. Performing this experiment we set thresholds 1200, 1300, 1400, 1500,1600 so that the runtime variation of different thresholds can be clearly shown. We record the run-time for entire dataset basically the time changes is observed without difficulty. X-axis of the graph represents the minimum support threshold and Y-axis represents the corresponding run-time. In real life altering of thresholds is meaningful as different applications are searching only that patterns which have real significance.We perform Similar type of analysis on Mashroom datasets and Retail datasets. In this time number of transaction are 8124 and respectively . Result of this analysis are shown in figure 4.1(b) and 4.1(c).

4.3.2 Effect of window size variation

Our proposed method is sliding window based which capture the most current data and execute mining to recent window so runtime and memory requirement on window size. so, we change number of batch in a window and number of transactions in a batch to evaluate the corresponding runtime.while varying window size we choose chess data and mushroom data to execute runtime analysis. Figure 4.2 shows the effect of runtime with respect to window size. The X-axis reflect the window size and Y-axis for runtime in seconds.Performing analysis we set window size 4,6,8,10,12 with chess data set, and 2,4,6,8,10 with mushroom dataset.Figire 4.2(a) and 4.2(b) respectively reflect the effect of window size on runtime. Batch size of chess data is 250 and threshold was 38%. Batch size of chess data is 700 and threshold was 7.3%.

Run-time Comparison with WFPMDS- growth algorithm: The existing algorithm WFPMDS-growth algorithm is compatible for mining weighted frequent patterns from data streams. But as their prefix tree are quite large the mining operation requires much time. On the other hand, our proposed method CPTSW-growth algorithm generate more compact prefix tree using path adjusting method, so mining operation needs less time than WFPMDS-growth algorithm. It is clear as our generated prefix tree is compacted so that it took less time

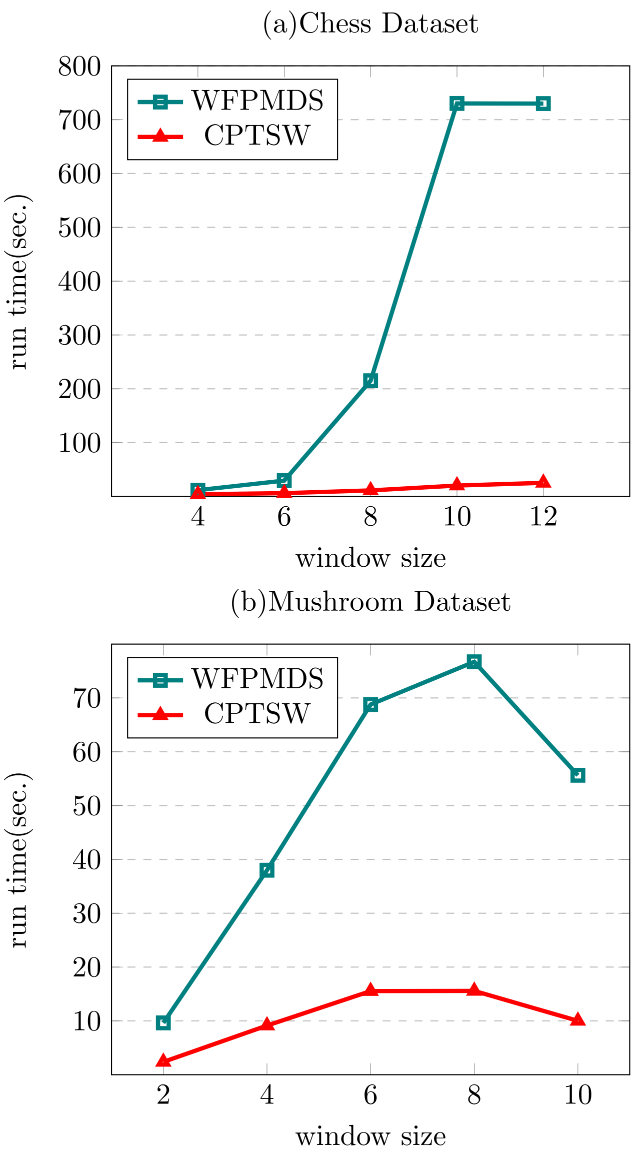


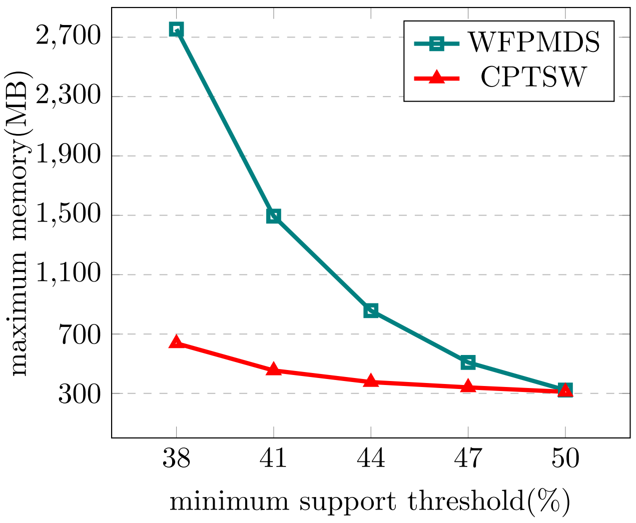
FIGURE 4.2: Window size analysis

4.3.3 Memory consumption analysis

The memory requirement for the prefix tree is low enough to use the gigabyte range memory available now a days is shown by research on prefix tree based frequent mining. If we can handle our prefix tree we can reduce our memory usage. Our proposed CPTSW tree can represent information of transaction in a very compressed structure because transactions have common items.By using more prefix sharing, our CPTDS tree structure can reduce memory space.

Another experiment has been conducted on CPTSW and WFRMDS methods by showing the effect of memory vs threshold. Chess dataset has 3196 transactions with 75 distinct items. Figure 4.2 describe the effect of thresholds in memory space. to perform this experiment we define thresholds 1200, 1300, 1400, 1500,1600 so that the memory consumption effect of different thresholds can be clearly represented.we record the memory for entire data set basically the change of time can be easily noticed. In the graph X-axis represents the threshold and Y-axis represent the corresponding memory . Variation of thresholds in real life is important as different applications are looking only those patterns that have real significance. We also performed similar type of analysis on Mashroom datasets. here , number of transaction are 8124 and respectively . Result of this analysis are shown in figure 4.2(b).

(a)Chess Dataset



(b)Mashroom Dataset

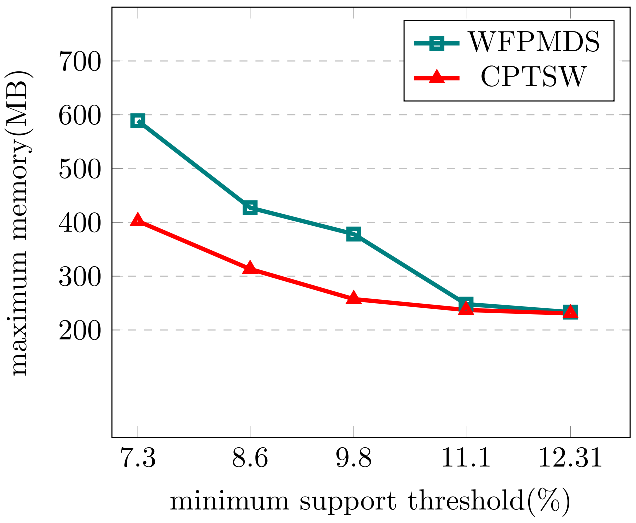


FIGURE 4.3: Memory Analysis

4.3.4 Runtime Distribution

Recall from 3.1 about our CPTSW tree construction process, CPTDSW requires several swapping operations to restructure the tree in frequency descending order after the insertion of the transaction of each batch. So there might arouse an issue that, CPTSW should requires more time for the tree construction which is true. But our prefix tree is compact , so our mining time is less than WFPMDS method. We get a significant gain in overall runtime due to the frequency descending compact structure of tree.

4.3.5 Scalability of CPTSW

our proposed algorithm can easily handle large number of transaction containing datasets that is shown by the experimental results. Hence the experimental result exhibit the scalability of our proposed algorithm to manage and mine large number of transaction and separate item. Our CPTSW algorithm outperforms the existing WFPMDS method by using proficient tree structure and pattern mining technique in terms of run-time and memory usage.

4.4 Conclusion

In our thesis work we basically propose a novel tree structure which efficiently successfully capture data stream and pattern mining algorithm which extract only significant patterns from most recent window. for tree construction and mining operations,it claims only single pass of stream data .This approach suitable to use in real time data processing to discover valuable recent knowledge. Our CPTSW saves time consumption and memory space by using an efficient tree structure and mining approach . Extensive performance shows our algorithm can handle a large number of items and transactions.

Chapter 5

Conclusions

In this thesis, we developed a sliding window based strategy of finding weighted patterns from data stream. Our proposed CPTSW-growth is a complete method that successfully capture the uncertain static weighted data and mine only significant patterns that depends on users interest. In our strategy which patterns are important to users can be defined by them and mine only those significant patterns. CPTSW-growth just doesn't generate all frequent patterns without considering anything but those patterns that have significant value to the application.

5.1 Research Summary

Throughout this book we organize our works like in Chapter 1 , by discussing current situation of data mining we introduced our research topic ,why uncertain data is required in today's world. Specially in section 1.1 we clearly specify the need of mining methods for uncertain data stream. Problem of the existing system is also discussed here. In order to solve the current problems we had to face several challenges. In section 1.2 those challenges are discussed in details. Section 1.4 represent our contribution on data mining research. To deal with the current problem that is finding weighted patterns from uncertain data stream we need to research several data mining methods, techniques, tools in order to establish our approach. Section 2.1 introduce important topics related to our research. In section 2.1.1 to 2.1.6 contains details description of those topics. In section 2.2 we described those concepts which influenced our work .Here we talked about those methods that basically inspired us to do our works like DS-Tree, WFPMDS-growth, DWFPM-growth in section 2.2.1, 2.2.2,2.2.3 . Total summery of current methodologies that deal with uncertain data and their limitations are discussed in 2.3 . 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 concept in section 3.3. Finally I n section 3.4 we describe our methods, mining techniques, analysis and so on in detains. Chapter 4 basically shows the experimental results ,analysis of result, several comparisons with existing method and observing different characteristics of data sets used to analysis our methods.

5.2 Future works

In our thesis works we basically focus on decreasing mining time and memory requirement by compacting prefix tree of WFPMDS-growth method. In future we will try to apply some other constrain to find more strong and interesting patterns which have more significance and more realistic to users.

Bibliography

[1] Rakesh Agrawal, Tomasz Imieliński, and Arun Swami. Mining association rules between sets of items in large databases. In *Acm sigmod record*, volume 22, pages 207–216. ACM, 1993.

[2] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[3] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data mining and knowledge discovery*, 8(1):53–87, 2004.

[4] Jiawei Han, Hong Cheng, Dong Xin, and Xifeng Yan. Frequent pattern mining: current status and future directions. *Data mining and knowledge discovery*, 15(1): 55–86, 2007.

[5] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, and Young-Koo Lee. Cp-tree: a tree structure for single-pass frequent pattern mining. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 1022–1027. Springer, 2008.

[6] Unil Yun and John 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*, pages 636–640. SIAM, 2005.

[7] Unil Yun and John J Leggett. Wlpminer: weighted frequent pattern mining with length-decreasing support constraints. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 555–567. Springer, 2005.

[8] Unil Yun. Efficient mining of weighted interesting patterns with a strong weight and/or support affinity. *Information Sciences*, 177(17):3477–3499, 2007.

[9] Chun Hing Cai, Ada Wai-Chee Fu, Chun Hung Cheng, and Wang Wai Kwong. Mining association rules with weighted items. In *Proceedings. IDEAS'98. International Database Engineering and Applications Symposium (Cat. No. 98EX156)*, pages 68–77. IEEE, 1998.

[10] Feng Tao, Fionn Murtagh, and Mohsen Farid. Weighted association rule mining using weighted support and significance framework. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 661–666. ACM, 2003.

[11] Unil Yun. Mining lossless closed frequent patterns with weight constraints. *Knowledge-Based Systems*, 20(1):86–97, 2007.

[12] Chedy Raïssi, Pascal Poncelet, and Maguelonne Teisseire. Towards a new approach for mining frequent itemsets on data stream. *Journal of Intelligent Information Systems*, 28(1):23–36, 2007.

[13] Ahmed Metwally, Divyakant Agrawal, and Amr El Abbadi. An integrated efficient solution for computing frequent and top-k elements in data streams. *ACM Transactions on Database Systems (TODS)*, 31(3):1095–1133, 2006.

[14] Nan Jiang and Le Gruenwald. Research issues in data stream association rule mining. *ACM Sigmod Record*, 35(1):14–19, 2006.

[15] Carson Kai-Sang Leung and Quamrul I Khan. Dstree: a tree structure for the mining of frequent sets from data streams. In *Sixth International Conference on Data Mining (ICDM'06)*, pages 928–932. IEEE, 2006.

[16] Yun Chi, Haixun Wang, Philip S Yu, and Richard R Muntz. Moment: Maintaining closed frequent itemsets over a stream sliding window. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 59–66. IEEE, 2004.

[17] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM, 2000.

[18] Chris Giannella, Jiawei Han, Jian Pei, Xifeng Yan, and Philip S Yu. Mining frequent patterns in data streams at multiple time granularities. *Next generation data mining*, 212:191–212, 2003.

[19] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26, 2005.

[20] Young-Hee Kim, Won-Young Kim, and Ung-Mo Kim. Mining frequent itemsets with normalized weight in continuous data streams. *Journal of information processing systems*, 6(1):79–90, 2010.

[21] Roberto J Bayardo Jr. Efficiently mining long patterns from databases. In *ACM Sigmod Record*, volume 27, pages 85–93. ACM, 1998.

[22] Francesco Bonchi and Claudio Lucchese. On closed constrained frequent pattern mining. In *Fourth IEEE International Conference on Data Mining (ICDM'04)*, pages 35–42. IEEE, 2004.

[23] Cristian Bucilă, Johannes Gehrke, Daniel Kifer, and Walker White. Dualminer: A dual-pruning algorithm for itemsets with constraints. *Data Mining and Knowledge Discovery*, 7(3):241–272, 2003.

[24] Laks VS Lakshmanan, Carson Kai-Sang Leung, and Raymond T Ng. Efficient dynamic mining of constrained frequent sets. *ACM Transactions on Database Systems (TODS)*, 28(4):337–389, 2003.

[25] Carson Kai-Sang Leung, Laks VS Lakshmanan, and Raymond T Ng. Exploiting succinct constraints using fp-trees. *ACM SIGKDD Explorations Newsletter*, 4(1): 40–49, 2002.

[26] Jong Soo Park, Ming-Syan Chen, and Philip S. Yu. Using a hash-based method with transaction trimming for mining association rules. *IEEE transactions on knowledge and data engineering*, 9(5):813–825, 1997.

[27] Jian Pei, Jiawei Han, and Laks VS Lakshmanan. Mining frequent itemsets with convertible constraints. In *Proceedings 17th International Conference on Data Engineering*, pages 433–442. IEEE, 2001.

[28] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, Young-Koo Lee, and Ho-Jin Choi. Single-pass incremental and interactive mining for weighted frequent patterns. *Expert Systems with Applications*, 39(9):7976–7994, 2012.

[29] Wei Wang, Jiong Yang, and Philip Yu. War: weighted association rules for item intensities. *Knowledge and Information Systems*, 6(2):203–229, 2004.

[30] William Cheung and Osmar R Zaiane. Incremental mining of frequent patterns without candidate generation or support constraint. In *Seventh International Database Engineering and Applications Symposium, 2003. Proceedings.*, pages 111–116. IEEE, 2003.

[31] Hao Huang, Xindong Wu, and Richard Relue. Association analysis with one scan of databases. In *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 629–632. IEEE, 2002.

[32] Carson Kai-Sang Leung, Quamrul I Khan, Zhan Li, and Tariqul Hoque. Cantree: a canonical-order tree for incremental frequent-pattern mining. *Knowledge and Information Systems*, 11(3):287–311, 2007.

[33] Jin Li, David Maier, Kristin Tufte, Vassilis Papadimos, and Peter A Tucker. No pane, no gain: efficient evaluation of sliding-window aggregates over data streams. *SIGMOD Record*, 34(1):39–44, 2005.

[34] Chih-Hsiang Lin, Ding-Ying Chiu, Yi-Hung Wu, and Arbee LP Chen. Mining frequent itemsets from data streams with a time-sensitive sliding window. In *Proceedings of the 2005 SIAM International Conference on Data Mining*, pages 68–79. SIAM, 2005.

[35] Md Badi-Uz-Zaman Shajib, Md Samiullah, Chowdhury Farhan Ahmed, Carson K Leung, and Adam GM Pazdor. An efficient approach for mining frequent patterns over uncertain data streams. In *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, pages 980–984. IEEE, 2016.

[36] Syed Khairuzzaman Tanbeer, Chowdhury Farhan Ahmed, Byeong-Soo Jeong, and Young-Koo Lee. Sliding window-based frequent pattern mining over data streams. *Information sciences*, 179(22):3843–3865, 2009.

[37] Ben Shneiderman. Batched searching of sequential and tree structured files. *ACM Transactions on Database Systems (TODS)*, 1(3):268–275, 1976.

[38] Darius S Culvenor. Extracting individual tree information. In *Remote Sensing of Forest Environments*, pages 255–277. Springer, 2003.

[39] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, and Byeong-Soo Jeong. Efficient mining of weighted frequent patterns over data streams. In *2009 11th IEEE International Conference on High Performance Computing and Communications*, pages 400–406. IEEE, 2009.

[40] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, and Young-Koo Lee. Handling dynamic weights in weighted frequent pattern mining. *IEICE TRANSACTIONS on Information and Systems*, 91(11):2578–2588, 2008.