Sliding Window Based Infrequent Weighted Itemset Mining

by

Abdullah Al - Matin

Exam Roll: Curzon Hall– 1085

Registration No: Ha - 895

Session: 2013–2014

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

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
UNIVERSITY OF DHAKA

November 2015

Declaration

I, hereby, declare that the work presented in this project is the outcome of the investigation performed by me under the supervision of Dr. Chowdhury Farhan Ahmed, Associate Professor, Department of Computer Science and engineering, University of Dhaka. I also declare that no part of this project has been or is being submitted elsewhere for the award of any degree or diploma.

Countersigned

.....

(Dr. Chowdhury Farhan Ahmed)

Supervisor

Signature

.....

(Abdullah Al - Matin)

Abstract

Now a days, frequent itemset mining is a popular approach. For finding frequent itemset, several approaches is already established. Infrequent items can be also very important for finding unexpected behavior like fraud detection, cost function minimization, equal task distribution for data center resource management etc. Weight may play a very important role in this scenario. In general approach, weighted itemset is treated all alike. But we have proposed an algorithm for mining infrequent weighted itemset by treating them differently based on their weight. Experimental results and graphs are added for the support of our proposed algorithms efficiency and effectiveness.

Acknowledgements

First of all, I am thankful and expressing my gratefulness to Almighty Allah who offers me His divine blessings, patient, mental and psychical strength to complete this project work. I am deeply indebted to my project supervisor Dr. Chowdhury Farhan Ahmed, Associate Professor, Department of Computer Science and Engineering, University of Dhaka. His scholarly guidance, important suggestions, endless patience, constant supervision, valuable criticism, and enormous amount of work for going through my drafts and correcting them, and generating courage from the beginning to the end of the research work has made the completion of the project possible.

I would like to express my deep gratitude to Md. Samiullah and Muhammed Tawfiqul Islam, lecturer, Dept of CSE DU for his support and help for my work. The discussions with him on various topics related my works have helped me to enrich my knowledge and conception regarding this work.

Last but not the least; I am highly grateful to my parents and family members for their support and constant encouragement, which have always been a source of great inspiration for me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Data mining is the process for finding interesting and useful patterns from a dataset. The very first approach was finding frequent items i.e. the items that are used frequently. It plays an important role for finding knowledge discovery technique such as association rule mining, classification, clustering, time-series mining, graph mining, web mining etc. A lots of research and algorithm have been proposed for finding frequent patterns using Apriori based algorithms[], FP tree based algorithms[] and others algorithms[]. In real life, frequent item set finding application is used in market basket analysis [1], medical image processing [4], biological data analysis [9] etc. Traditionally, all items in a transaction are treated equally. Though all items interest/influence is not same, they should be treated differently. For solving this problem, the notion of weighted item set mining has been introduced [18],[19],[20]. The weight of an item is the value that signifies the local interest within each transaction.

Several approaches have been proposed for IWIM (Infrequent Weighted Itemset Mining) [14], [15]. In those approaches, they consider all data from the very beginning of the database. Therefore these approaches cannot be applicable for a large-scale data set such as data streams [7]-[8],[17]-[12], where data set flows in the form of continuous stream. Data stream is a continuous, unbound and ordered sequence of items that comes in order of time. In this case, it is impossible to

Transaction IDs (tid)	CPU usage reading
1	(a,0) (b,100) (c,57) (d,71)
2	(a,0) (b,43) (c,29) (d,71)
3	(a,43) (b,0) (c,43) (d,43)
4	(a,100) (b,0) (c,43) (d,100)
5	(a,86) (b,71) (c,57) (d,71)
6	(a,57) (b,71) (c,0) (d,71)
7	(a,91) (b,32) (c,11) (d,0)
8	(a,17) (b,61) (c,0) (d,72)

TABLE 1.1: Example of a transaction data stream with weight

maintain all data from the dataset and cannot possible to process all data at a time.

Existing algorithm works on whole fixed dataset and generates infrequent item set. For data stream, it have handle data in an efficient manner. Older data will be lost and also new updated data have to process. So it should differentiate recently generated data from older information which may unimportant or obsolete in context of time.

Consider as an example, the data set in Table 1.1 where there is a set of transactions from data stream. Here we have considered a small part of the data set for simplicity. It contains eight transactions. These transactions are identified by tids. Each transaction contains four distinct items with their weights. Its not necessary to be exact four items. Weight is defined by the corresponding degree of interest in the transaction i.e. item b contains weight 43 in tid 2 but in tid 3 is 0. Item weight is the value of CPU usage readings at a specific time. For data stream, this CPU usage reading is done at a fixed sampling rate. This CPU usage is important for data center resource management and application specific profiling. In tid4, weight of item a is 100. It means CPU a is working at high usage rate. While in tid 1, its weight is 0. It means temporarily a is idle. Knowledge extracting from this data set is important for domain expert or specific application expert to allocate system task break down for all CPU at a specific time. So that, the system output can be maximized.

1.1 Basic Concepts

The main approaches of data mining include:

Apriori: Apriori is a common approach for finding frequent items and association rule learning over transactional database. It was proposed by Agarwal and shrikant 1994[]. Apriori uses bottom-up approach. Subsets are extended one item at a time (also known as candidate generations) and groups of candidates are tested against the data. This procedure terminates when no further successful extensions are found. It uses breadth-first-search and a Hash tree structure to count candidate item sets efficiently. It generates candidate item sets of length K from item sets of length k-1. Infrequent sub patterns are pruned. Pruning is done based on minimum support (min_sup). Given a set of transactions (similar to database records in this context), where each transaction consists of items (or attributes), an association rule is an implication of the form $A \Rightarrow B$, where A and B are sets of items and $A \cap B = \emptyset$. The support of this rule is defined as the percentage of transactions that contain the set $A \cup B$, while its confidence is the percentage of these A transactions that also contain items in B. In association rule mining, all items with support higher than a specified minimum support are called frequent itemsets. One drawback of this approach is it requires multiple database scan. If number of items is large, it creates a lots of candidate patterns. It is also a limitation.

FP Growth (Frequent Pattern Growth): FP-Growth algorithm was proposed by Han[]. It is an efficient way for finding frequent items without generating candidate list. It uses divide-and-conquer approach. It uses a special data structure named frequent pattern tree, which retains the itemset association information. It decomposes the mining task into a set of smaller tasks for mining confined patterns in conditional databases, which dramatically reduces the search space. FP-growth method is efficient and scalable for mining both long and short frequent patterns, and is about an order of magnitude faster than the Apriori algorithm and also faster than some recently reported new frequent pattern mining methods.

Association Rule: Association rule is implied for discovering interesting relationships between variables in large databases. Following the original definition by Agrawal et al., [] The problem of association rule mining is defined as: Let $I = \{i_1, i_2, \ldots\ldots, i_n\}$ be a set of n binary attributes called items. Let $D = \{t_1, t_2, t_3, \ldots\ldots, t_n\}$ be a set of transactions called the database. Each transaction in D has a unique transaction ID and contains a subset of the items in I. A rule is defined as an implication of the form: $X \Rightarrow Y$ Where $X, Y \subseteq I$ and $X \cap Y = \emptyset$. Every rule is composed by two different set of items, also known as itemsets, X and Y, where X is called antecedent or left-hand-side (LHS) and Y consequent or right-hand-side (RHS).

Transaction IDs (tid)	milk	bread	butter	beer	diapers
1	1	1	0	0	0
2	0	0	1	0	0
3	0	0	0	1	1
4	1	1	1	0	0
5	0	1	0	0	0

TABLE 1.2: Sample transaction database

To illustrate the concepts, we use a small example from the supermarket domain in Table 1.2. The set of items is $I = \{$milk, bread, butter, beer, diapers$\}$ and in the table is shown a small database containing the items, where, in each entry, the value 1 means the presence of the item in the corresponding transaction, and the value 0 represent the absence of an item in a that transaction. An example rule for the supermarket could be $\{butter, bread\} \Rightarrow milk$ meaning that if butter and bread are bought, customers also buy milk.

Classification: Classification is the problem of identifying to which of a set of categories a new data belongs. It is done on the basis of a training set of data whose class label is known.

Cluster Analysis: Cluster analysis deals with data for which the class labels are

not known. Here, data is clustered or grouped based on the principle of maximizing the intraclass similarity and minimizing the interclass similarity [**?**].

Outlier Analysis: A data set may contain some data which behaves differently than the rest of the data. This type of data is known as outlier. Sometimes this outliers provides us with useful information about the dataset. Analysis of these outlier data is known as outlier analysis.

1.2 Infrequent Itemset Mining

Frequent itemset mining is a very common approach for finding frequent items. In our approach, we are trying to find infrequent items. Those are the infrequent items, whose value is below threshold. For a given max_support value, items value is below the max_support. Infrequent item is important for resource management, application profiling, and fraud detection. It can be used for finding any unexpected or abnormal behaviour of a system.

1.3 Motivation

Mining for infrequent itemset is recently gained attention of research community. Main focus for finding infrequent itemset whose frequency value is below `max_support`. Not only frequency, weight can be also used for finding infrequent itemset mining.

Lets consider a scenario. A banking system where lots of transactions is done every day. As a bank officer, one may want to know if everything okay or not. In general cases, all transactions are okay but in fraud cases or unexpected cases, it may vary. Its difficult for a bank officer to browse all data of the transactions to find any abnormal cases every day. Its pretty hard for him.

1.4 Objective

We have studied that the existing approaches are limited to some context. As a solution, we have to develop an algorithm that can overcome the limitations. The objectives of our work are listed below:

- To carry out an extensive literature view regarding the infrequent itemset mining approach.
- Propose a new efficient algorithm for finding infrequent items.
- Propose an algorithm that can perform in broader area.

1.5 Thesis Contribution

The contributions our research work are listed below:

- We have used sliding window based approach. This reduces the processing time & memory space.
- We can get the actual contribution of an item in the dataset. Previous approach uses equivalence weighting function for data items.
- Its easier to discard frequent items from the candidate infrequent pattern list.
- Extra cost for checking infrequent patterns validation (i.e. is the item actually infrequent or not) is not required.

1.6 Thesis Outline

The five chapters labeled as Introduction, Related Work, Proposed Approach, Performance Evaluation and Conclusion form the shape of the book. Every chapter comprises of the following topics:

Chapter 1 contains some introductory concepts such as data mining and various data mining approaches, infrequent itemset mining and also the motivation and contribution of this work.

Chapter 2 describes various terms related to previous work in the field of infrequent itemset mining and also the limitations of the existing approaches.

Chapter 3 describes the proposed score and algorithm. It also contains the details of our work and application of the proposed algorithm.

Chapter 4 contains the implementation results and performance evaluation results of the proposed algorithm.

Chapter 5 draws the conclusion of the thesis by summarizing the findings in the thesis and describing scope of possible extensions to this work.

The last part of the book is Bibliography which places all the reference cited in the work. The appendix works as additional information that helps the reader if necessary.

Chapter 2

Previous Works

In this chapter, we discuss some basic terminologies and background knowledge in infrequent itemset mining, sliding window based approach and Share-frequent approach. We also discuss some recent works in the field of infrequent itemset mining. Here, we highlight their working procedure and analyze the algorithms.

2.1 Related Works

Frequent itemset mining is commonly used and established process for finding frequent items [1]. In general approaches, all items in a dataset is treated equally. But when an item is having weight, it cant be treated equally with other items. If we do so, we will lose more information from the dataset. For differentiating the itemsets, [20] authors focus on finding more informative association rules. This rule is named Weighted association rules (WAR). In this approach, weight denotes the significance of an item. Based on weight, items are processed for mining. Mining process is also done by keeping weight in mind. Weight can be pre-assigned or not. Several approaches have been already established [19], [20]. Based on weight, infrequent itemset mining has also gained interest. Those itemsets are infrequent that frequency is less than min_support. Their frequency is also quite little. In [5], minimal infrequent itemsets are mined. In this approach, they have taken a

weighted dataset. Gives every item an equivalence weight for simplicity. Then they have created a weighted tree. This weighted is sent for mining. In mining process, they have applied FP-Growth like algorithm for finding infrequent itemsets. In their approach, tree construction is conducting more time. It also creates the tree more branches. Mining process shows infrequent items but they cannot provide the actual contribution of an item in the dataset. This approach works on a fixed dataset and cannot be applied for stream data.

Sliding window is an approach for processing stream data. In stream data, data is a continuous channel and arrives within a fixed time range. So, it requires to process those data within a very small span of time. In [3], an efficient approach is proposed for finding frequent items using sliding window mechanism. It introduces the idea of window size and batch size. Each batch contains a numbers of transactions. Window size is the number of batches. Window slides to next window new batches come. As previous data are no longer present, so it deletes the previous data and its related weight from table. It constructs the tree for a window and perform mining process. When a window is completed, it slides to next.

Let, $I = \{i_1, i_2, \ldots, i_m\}$ be a set of items and T be a transaction database Transactional dataset $D = \{T_1, T_2, T_3, \ldots, T_n\}$ where each transaction Ti D is a subset of I. The support of an item is the number of transaction containing the item in the transactional database. To find frequent items, items should satisfy the minimum support in the transaction database. The downward closure property (Agarwal et al., 1993; Agarwal and Srikant, 1994) is used prune the infrequent items. This property ensures that if an item is infrequent then its supper set must be infrequent. In our approach, we are targeting these infrequent items. Apriori (Agarwal et al., 1993; Agarwal and Srikant, 1994) algorithm is widely used for mining frequent items and very useful in association rule mining. But candidate

generation and test, several database scan are the side effect of appriori algorithm. FP-growth (Han et al., 2004) algorithm solves the problem of candidate generation and test. It requires only two database scan.

To discover useful knowledge about numerical attributes associated with items in a transaction, Carter et al., 1997 first introduced the share-confidence model. SIP, CAC and IAN (Barber and Hamilton, 2000, 2001, 2003) have been proposed but they may not discover all shared-frequent patterns. Several approaches have been also proposed to resolve the discovery of all frequent items and that also maintain the downward closure property. In [], an effective approach is described,

The existing weighted infrequent pattern mining methods consider all transactions of a database from the very beginning and requires extra database scan. Hence, they are not appropriate for data stream. They cant find recent interesting information from the dataset. Therefore, we have proposed sliding window based novel algorithm for single-pass weighted infrequent pattern mining to extract the recent change of knowledge in a data stream adaptively. We have also applied the ShrFP-Tree approach to find the actual contribution of a pattern.

Chapter 3

Proposed Approach

In this chapter we present our proposed approach for finding infrequent itemset mining over data stream. We discuss about the definition, process and algorithms of our proposed approach.

3.1 Problem Definition

This paper address the problem of infrequent weighted itemset mining from a transaction dataset. Let $I = \{i_1, i_2, \ldots, i_m\}$ a set of data items. Transactional dataset $T = \{t_1, t_2, t_3, \ldots, t_n\}$ is a set of transaction where each transaction is a set of items in I and denoted by transaction ID (tid).

An Item set is a set of data items i.e. in k item sets there is k items. Support for an item is the count of that item in the dataset. Based on support count, an item may be frequent or infrequent. For identify an item as frequent or infrequent, a threshold E is used. This threshold can be set based on user interest. In this paper, we are targeting infrequent items. So, we need to discard items that are above the threshold.

In regular basis, for frequent item set mining, items and transactions are consider in the same way. But [5] according to this paper, we also treating each item differently. Here, each item is paired with its weight i.e (I,k), I is an item contain in the

transaction set T, K is the weight of the item that signifies the interest/intensity in the transaction.

As we are working on data stream, we need to process data in an efficient manner. To find infrequent itemset mining from a data stream, we cant perform multiple scan from a data stream. Once the streams flow through we lose them. To find recent important knowledge from a data stream, Single-pass and sliding window based mechanism [2] is required.

We have adopted similar definitions presented in the previous works (Carter er al., 1997; Barbar and Hamilton, 2000, 2001, 2003; Li et al., 2005a,b). Let $I = \{i_1, i_2, \ldots\ldots, i_m\}$ be a set of items and D be a transaction database $\{T_1, T_2, \ldots\ldots, T_n\}$ where each transaction Ti D is a subset of I.

Definition 1: The measure value mv(ip, Tq), represents the weight of an item ip in transaction Tq. For example, in Table 1.1, mv(a,T3) = 43.

Definition 2: The transaction measure value of a transaction Tq denoted as tmv(Tq) means the total measure value of a transaction Tq and it is defined by,

$$tmv(Tq) = \sum_{ip \in Tq} mv(ip, Tq) \tag{3.1}$$

For example, tmv(T4) = mv(a, T4) + mv(c, T4)+ mv(d, T4) = 100 +43+100 = 243 in Table 1.1.

Definition 3: The total measure value Tmv(DB) represents the total measure value in DB. It is defined by,

$$Tmv(DB) = \sum_{Tq \in DB} \sum_{ip \in Tq} mv(ip, Tq) \tag{3.2}$$

For example, Tmv(DB) = 1140 in Table 1.1 for window 2.

Definition 4: The itemset measure value of an itemset X in transaction Tq, imv(X, Tq) is defined by,

$$imv(X, Tq) = \sum_{ip \in X} mv(ip, Tq) \tag{3.3}$$

where X = $\{i_1, i_2, \ldots\ldots, i_k\}$ is a k-itemset,

Definition 5: The local measure value of an itemset X is defined by,

$$lmv(X) = \sum_{Tq \in DBX} \sum_{ip \in X} mv(ip, Tq) \tag{3.4}$$

where DBX is the set of transactions contain itemset X. For example, lmv(bc) = imv(bc, T5) +imv(bc, T5) = 71 + 57 + 32 + 11= 171 in Table 1.1 for window 2.

Definition 6: The share value of an itemset X, denoted as SH(X), is the ratio of the local measure value of X to the total measure value in DB. So, SH(X) is defined by,

$$SH(X) = \frac{lmv(X)}{Tmv(DB)} \tag{3.5}$$

For example, SH(bc) = 171/1140 = 0.15, in Table 1.1 in window 2.

Definition 8: Given a maximum share threshold, $maxShare$, an itemset is share-infrequent if $SH(X) \geq maxShare$. If maxShare is 0.25 (we can also express it as 25%), in the example database, bc is a share-infrequent itemset, as SH(bc) = 0.15.

3.2 Solution Methodology

In this section, we describe our proposed approach for mining infrequent weighted itemset. Our approach is divided in two portions. In first portion, we construct

the tree with weighted inputs. In second portion, we pass the tree for mining process.

Construction process of our tree structure is to store the stream data using a single pass. We use ShrFP-Tree (Share-frequent pattern tree) for share-infrequent pattern mining. The header table is maintained to keep an item order in our tree structure. Each entry of a transaction in a header table explicitly maintains item-id and weight information for each item. Here, weight is the *tmv* (transaction measure value) values of items. To facilitate the tree traversals adjacent links are also maintained (not shown in figure for simplicity) in our tree structure.

Consider the example data stream at Table 1.1 . At first, ShrFP-Tree captures the *tmv* values of all items and keep in header table as weight. After that, we scan each database transaction one by one and then insert in the tree. In our example, first transaction T1 contains four items one of which is 0. If any items weight is 0 in the transaction, we will not insert that item in the tree. So, now we have three transactions b,c,d and insert them in the tree.

Item	Weight
b	371
c	371
d	371

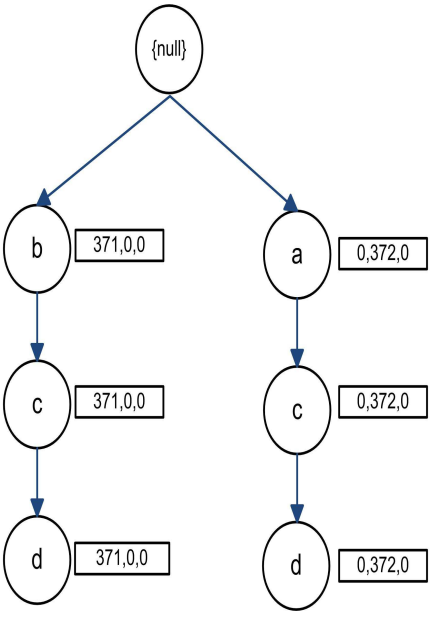

FIGURE 3.1: Insertion of Batch 1

Figure 3.1 shows the tree and header table after inserting batch 1. For batch 2

insertion in the tree, we calculate tmv for transactions. Then check in any item contains 0. If any item having 0, then omit it from the transaction. After that insert the transaction in the tree.

Item	Weight
a	372
b	371
c	743
d	743

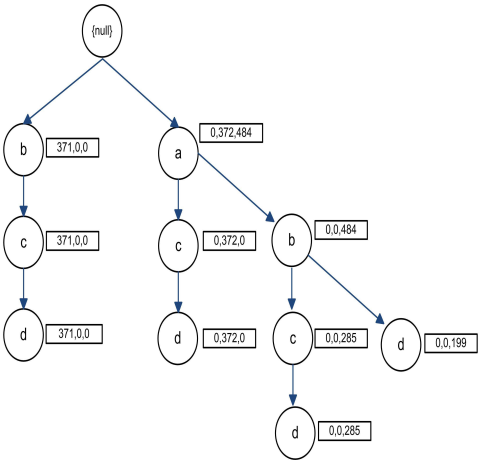

FIGURE 3.2: Insertion of Batch 2

Figure 3.2 shows the tree and header table after inserting batch 2. In the same way, batch 3 also inserted in the tree. The final tree after inserting window 1 in the tree is shown in Figure 3.3.

Item	Weight
a	856
b	855
c	1028
d	1227

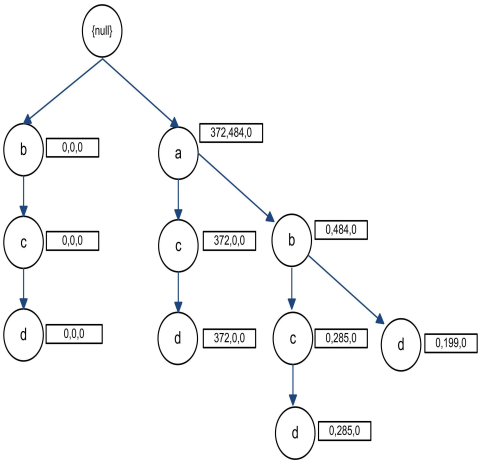

FIGURE 3.3: Insertion of Batch 3

As its a data stream, the stream moves to batch 4. It needs to delete previous information of batch 1. Because batch 1 is not belongs to window 2 and therefore

the information of batch 1 becomes garbage in the window 2 and it will no longer be used. We delete information of batch 1 from the tree. We also update the header table. After deletion of batch 1, the header table and the tree is shown in Figure 3.5. Some nodes which do not have any information of batch 2 and batch 3. We can delete them from the tree. In case of other nodes, the weight are shifted one position left to remove the weight information of batch 1.

Item	Weight
a	856
b	484
c	657
d	856

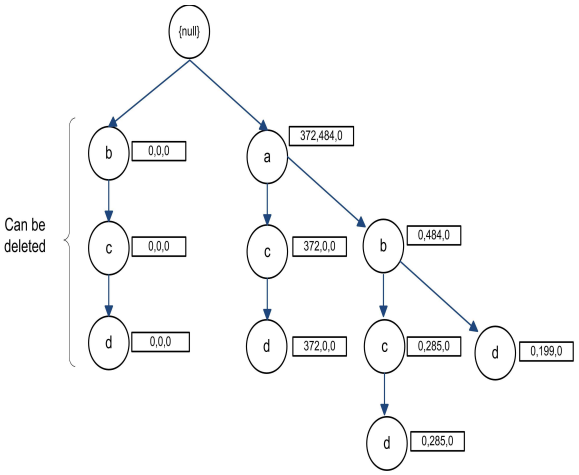

FIGURE 3.4: Deletion of Batch 1

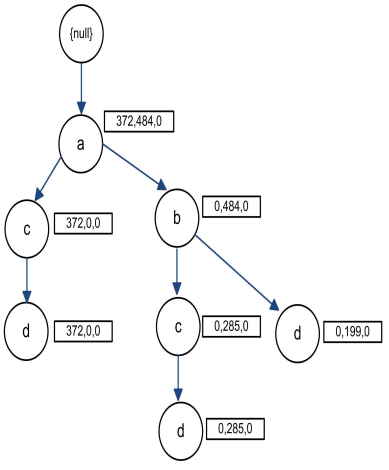

FIGURE 3.5: After deletion of Batch 1

Now insert new batch i.e. batch 4 in the tree. As a result, the three weight information of each node represents batch 2, batch 3 and batch 4. Figure 3.6 shows

the tree after insertion of batch 4.

Item	Weight
a	1140
b	768
c	791
d	1006

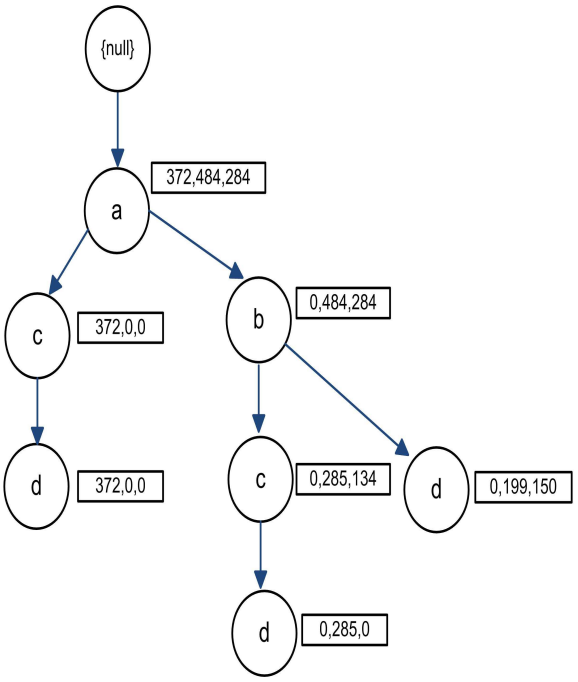

FIGURE 3.6: Insertion of Batch 4

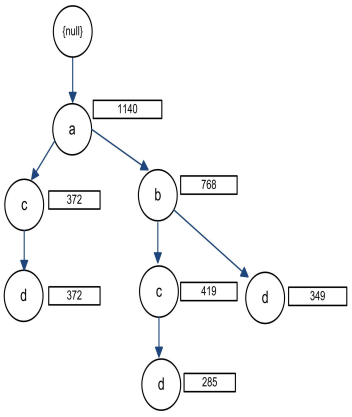

FIGURE 3.7: Complete tree after insertion of Batch 4

Property 1: The total value of *tmv* (transaction measure value) of any node in ShrFP-Tree is greater than or equal to the sum of total value of *tmv* values of its children.

3.3 Mining Process

ShrFP-Tree is a pattern growth mining process that mines all the candidate share-infrequent patterns. According to our property 1, pattern growth mining algorithm can be directly applicable to it by using *tmv* values.

Consider the database of Table 1.1 and maxShare = 0.20 in that database. The final ShrFP-Tree for the database we have considered in Table 1.1, is shown in Figure 3.7. At first, we have prepared a header table for all items. According to header table, least weighted item will be mined first. In our example, *b* is the first item for mining. A conditional tree is prepared for item *b* is shown in 3.8. For preparing conditional tree, we have taken all the branches prefixing the item *b*. Based on max_lmv value item *b* is not in candidate infrequent pattern list. But its candidate pattern can be in infrequent list. So, we have added them in the candidate infrequent pattern list. Then again, we have started for item *c*. Same process is applied for generating conditional tree for item *c* shown in 3.9. For item d, generation conditional tree is shown in 3.10 Then we have completed the process for generating candidate infrequent pattern list. Table 2 shows the calculation process for finding actual share-infrequent patterns from the candidate infrequent pattern list. The resultant share-infrequent patterns are $\{c\}$,$\{b,c\}$,$\{b,c,d\}$ for window 2.

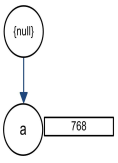

FIGURE 3.8: Conditional tree for item b

Item	Weight
a	791
b	419

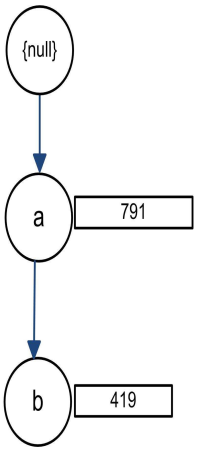

FIGURE 3.9: Conditional tree for item c

Item	Weight
a	1006
b	634
c	657

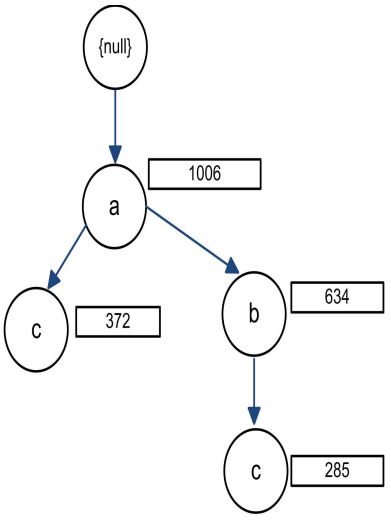

FIGURE 3.10: Conditional tree for item d

Patterns	lmv	SH	Shared Infrequent Patterns
b	235	0.206	No
c	154	0.135	Yes
d	357	0.313	No
ab	486	0.426	No
bc	171	0.152	Yes
ac	474	0.415	No
cd	357	0.313	No
bd	417	0.365	No
abc	384	0.336	No
acd	372	0.326	No
abd	577	0.506	No
bcd	199	0.174	Yes
$abcd$	285	0.251	No

TABLE 3.1: Calculation process of Share-Infrequent patterns

3.4 Proposed Algorithm

Algorithm will be placed here.

Chapter 4

Performance Evaluation

In this chapter, we compare the performance of the proposed approach with an existing approach, IWIM Using FP Growth [**?**]. We apply both the proposed approach and the existing approach to real world datasets. We perform several experiments to prove the accuracy of the proposed approach.

4.1 Experimental Settings

All experiments of the proposed approach and existing IWIM Using FP Growth are conducted on a machine with 3.30 GHz Intel CORE i3 processor, 4 GB ram and Windows OS. Both the algorithms have been implemented using Java programming language.

4.2 Dataset Characteristics

We perform a performance study in our experiments using real world datasets. We have also experimented with synthetic dataset.

4.3 Performance Metrics

We have considered the following performance metrics as parameter to evaluate the performance of the proposed algorithm:

Accuracy: It is the degree of correctness of the result of a calculation. In the case of graph classification, accuracy means the precision with which the graphs are classified.

Runtime: It is the time a program takes to produce the output. The matrices are following:

- Tree construction time parameter
- Mining time parameter
- Total time (Tree construction, Mining, Frequent removal) parameter

Window Size Variation: Depending on window size, we have compared the result.

4.4 Performance Comparison

4.5 Summary

Chapter 5

Conclusion

5.1 Summary of Research

5.2 Scope of Future Work

Appendix A

Appendix Title Here

Write your Appendix content here.

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] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, and Young-Koo Lee. Shrfp-tree: an efficient tree structure for mining share-frequent patterns. In *Proceedings of the 7th Australasian Data Mining Conference-Volume 87*, pages 79–86. Australian Computer Society, Inc., 2008.

[3] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, and Young-Koo Lee. Efficient tree structures for high utility pattern mining in incremental databases. *Knowledge and Data Engineering, IEEE Transactions on*, 21(12):1708–1721, 2009.

[4] Maria-Luiza Antonie, Osmar R Zaiane, and Alexandru Coman. Application of data mining techniques for medical image classification. In *Proceedings of the Second International Workshop on Multimedia Data Mining, MDM/KDD'2001, August 26th, 2001, San Francisco, CA, USA*, pages 94–101, 2001.

[5] Luca Cagliero and Paolo Garza. Infrequent weighted itemset mining using frequent pattern growth. *Knowledge and Data Engineering, IEEE Transactions on*, 26(4):903–915, 2014.

[6] Joong Hyuk Chang and Won Suk Lee. A sliding window method for finding recently frequent itemsets over online data streams. *Journal of Information science and Engineering*, 20(4):753–762, 2004.

[7] Joong Hyuk Chang and Won Suk Lee. estwin: Online data stream mining of recent frequent itemsets by sliding window method. *Journal of Information Science*, 31(2):76–90, 2005.

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

[9] Gao Cong, Anthony KH Tung, Xin Xu, Feng Pan, and Jiong Yang. Farmer: Finding interesting rule groups in microarray datasets. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 143–154. ACM, 2004.

[10] David J Haglin and Anna M Manning. On minimal infrequent itemset mining. In *DMIN*, pages 141–147, 2007.

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

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

[13] 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 *SDM*, pages 68–79. SIAM, 2005.

[14] Anna M Manning and David J Haglin. A new algorithm for finding minimal sample uniques for use in statistical disclosure assessment. In *Data Mining, Fifth IEEE International Conference on*, pages 8–pp. IEEE, 2005.

[15] Anna M Manning, David J Haglin, and John A Keane. A recursive search algorithm for statistical disclosure assessment. *Data Mining and Knowledge Discovery*, 16(2):165–196, 2008.

[16] 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.

[17] 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.

[18] Ke Sun and Fengshan Bai. Mining weighted association rules without preassigned weights. *Knowledge and Data Engineering, IEEE Transactions on*, 20(4):489–495, 2008.

[19] 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.

[20] Wei Wang, Jiong Yang, and Philip S Yu. Efficient mining of weighted association rules (war). In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 270–274. ACM, 2000.