2013

Detection of Interesting Traffic Accident Patterns by Association Rule Mining

Harisha Donepudi
Louisiana State University and Agricultural and Mechanical College

DETECTION OF INTERESTING TRAFFIC ACCIDENT PATTERNS BY ASSOCIATION RULE MINING

A Thesis

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Master of Science in Systems Science

in

The Department of Electrical and Computer Engineering

by
Harisha Donepudi
B.E., Anna University, 2010
August 2013

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to my major professor Dr. Omer Soysal for his valuable teachings, continuous support and guidance that motivated me to pick my thesis topic in the field of data mining. He is a great mentor and guided me through every step of my thesis.

My sincere thanks to Dr. Jianhua Chen, co-chair of my thesis committee for her valuable teachings, encouragement, guidance and suggestions that helped during my research.

I would also like to thank Dr. Jian Zhang for accepting my request to be a part of the thesis committee.

Finally, I thank my parents and grandparents for supporting my education till now and encouraging me to do thesis. It was their motivation and trust which made my degree possible.

TABLE OF CONTENTS

LIST OF FIGURES

ABSTRACT

In recent years, the accident rate related to traffic is high. Analyzing the crash data and extracting useful information from it can help in taking respective measures to decrease this rate or prevent the crash from happening. Related research has been done in the past which involved proposing various measures and algorithms to obtain interesting crash patterns from the crash records. The main problem is that large numbers of patterns were produced and vast number of these patterns would be obvious or not interesting. A deeper analysis of the data is required in order to get the interesting patterns. In order to overcome this situation, we have proposed a new approach to detect the most associated sequential patterns in the crash data. We also make use of the technique, "Association Rule Mining" to mine interesting traffic accident patterns from the crash records. The main goal of this research is to detect the most associated sequential patterns (MASP) and mine patterns within the data sets generated by MASP using a modified FP-growth approach in regular association rule mining. We have designed and implemented data structures for efficient implementation of algorithms. The results extracted can be further queried for pattern analysis to get a deeper understanding. Efficient memory management is one of the main objectives during the implementation of the algorithms. Linked list based tree structures have been used for searching the patterns. The results obtained seemed to be very promising and the detected MASPs contained most of the attributes which gave a deeper insight into the crash data and the patterns were found to be very interesting. A prototype application is developed in C# .NET.

1 INTRODUCTION

Today, data plays a major role in all the aspects. Mining data has become one of the emerging fields. In the field of data mining, many techniques have been proposed to extract useful information from the available data. Analyzing the data and retrieving useful information from it is data mining. Association rule mining is one of the techniques in data mining which is widely used to find relations hidden within data.

1.1 Motivation

Crash patterns can be identified by association rule mining, but all crash patterns found are not interesting and useful. Most of the rules obtained are obvious. There are various measures suggested in the field of association rule mining to find interesting patterns according to their target problem and hence no measure is universally best across all application domains [9].

The items with high frequency in the data may hinder the generation of interesting association rules. Most of the rules generated would involve the high frequency attribute values in them. In order to avoid this, we have proposed an algorithm to find the most associated sequential patterns and their respective blocks/counter blocks to be mined with respect to the association strength threshold.

1.2 Problem Statement

Finding interesting rules is one of the main problems in the field of association rule mining. Apriori and FP-Growth algorithms are more often used to find frequent item sets from the data. Vast number of these patterns would be obvious or not interesting. A new approach "Detecting MASPs" is presented in this thesis to find the most associated sequential patterns and also generate data sets (blocks) along with it that contains the transactions to be mined to find interesting patterns. A prototype application is developed in C# .NET.

1.3 Objectives

The objectives of this thesis are efficient implementation of the MASP and several other data mining algorithms proposed by Dr. Soysal and applying the proposed approach to the traffic accident data for detection of interesting patterns.

1.4 Definitions

- **Data table, D**: A set of all records which is consisted of attribute-value pairs.
- **Attribute set**: $A = \{A_i\}$, where $i = 1, 2, .., |A|$
- **Set of classes of an attribute**: $V_i = \{V_{ij}\}$, where $i = 1, 2, .., |A|$ and $j = 1, 2, .., |V_i|$
- **Item:** Each unique attribute-value pair is called an item. As a remark, each element of a record is an instance of the item.
- **Itemset:** Any set of items. An itemset can be considered as a pattern.
- **Transaction:** An instance of the data table.
- **Most associated sequential pattern (MASP):** A sequential itemset $(I_1, I_2, .., I_{i-1}, I_i, .., I_K)$ where a child item I_i has the highest frequency given its parent $(I_1, I_2, .., I_{i-1})$ among the other items at the level i and satisfies the association strength threshold is named as MASP. That is the probability $P(I_i \mid I_1, I_2, .., I_{i-1})$ is the maximum over the transactions conditioned to the parent MASP. Sometimes $MASP_t$ is used to identify a specific MASP; for simplicity, we omit the subscript 't' otherwise needed. A MASP pattern is sequential because each item in the pattern depends on the sub-pattern before the item.
- **Association Strength Threshold:** Association strength threshold $\tau_A = \alpha\, \tau_C$ is used to decide whether an item to be included in a MASP; where the weight α adjust the strength of the threshold and τ_C is the minimum confidence. Notice that the range of the alpha value is $[1, 1/\tau_C]$.
- **Item-max:** An item I_k at a level k is the item-max I_{MAX} of the MASP(k-1).
- **Item-space:** The item space at a level k is the set of transactions retrieved from the data table by the query statement of the level.
- **Block:** A set of transactions which is obtained by the query of the MASP. A MASP query is composed of three parts: 1) select clause formed by from the all attributes in the predicate excluding the ones which have the equality form of "attribute=value" in predicate of the MASP, 2)

Data source which is the set of all transactions to be mined, and 3) the predicate which is composed with conjunctions of the items in the MASP. Hereafter, |B| denotes the number of transactions in a block B.

- **Counter-block:** A set of transactions which are obtained by the query whose predicate is formed with replacing the logical operator of the last item of the MASP with "inequality"; the select clause is formed after this replacement as in formation of its block.
- **Q(D):** Q(D) is defined as a query to obtain a subset of the data table.
- **FP-tree:** The classic frequent-pattern tree is a tree structure defined below.
  "1. It consists of one root labeled as "null", a set of item-prefix subtrees as the children of the root, and a frequent-item-header table.
  2. Each node in the item-prefix subtree consists of three fields: item-name, count, and node-link, where item-name registers which item this node represents, count registers the number of transactions represented by the portion of the path reaching this node, and node-link links to the next node in the FP-tree carrying the same item-name, or null if there is none.
  3. Each entry in the frequent-item-header table consists of two fields, (1) item-name and (2) head of node-link (a pointer pointing to the first node in the FP-tree carrying the item-name)" [1].
- **K-Patterns:** An itemset of length K is called K-patterns.
- **Frequent pattern:** A pattern is frequent if its count is greater than or equal to the minimum support.

Note that the notion of our most associated sequential pattern (MASP) is different from the traditional sequential patterns [2] in the association rule mining literature. The items in the well-known concepts of sequential patterns are associated with a time stamp and aims at finding the frequently occurring sequences to predict the future data or mining the periodical patterns. For example "80% customers who bought a phone also bought a case and then bought the screen guard with certain time gap." On contrast, the items in the MASP are in a sequential order and do not have time stamps. A sequential pattern is like $(I_1, I_2)(I_3)(I_4, I_5)$. In a normal pattern, the order of the items does not matter, but in our approach, the items in the pattern are called sequential due to the order of the items. For example (I_1, I_2) is different from (I_2, I_1).

2 BACKGROUND

"Association rules are if/then statements that help uncover relationships between seemingly unrelated data in a relational database or other information repository" [2]. "An association rule is an implication expression of the form $X \rightarrow Y$, where X and Y are disjoint itemsets. i.e., $X \cap Y = \Phi$. The strength of the association rule can be measured in terms of its support and confidence" [3]. "The support of an association rule is the percentage of groups that contain all of the items listed in that association rule" [5]. "Support determines how often a rule is applicable to a given data table" [3]. The confidence of the rule is defined as the ratio of the proportion of the transaction in the data set that contains all the items in the rule to the proportion of the transaction in the data set that contains all the items in the rule antecedent. "Confidence determines how frequently items in Y appear in transactions that contain X" [3]. Given a rule$X \rightarrow Y$, where X is the antecedent and Y is the consequent, and the data set of size N, the support and confidence are defined by the equations (1) and (2). The interestingness of the association rule is measured using lift. The lift is defined as the ratio of the confidence of the rule to the expected confidence of the rule. The expected confidence of the rule is defined as the product of the support of the antecedent and the consequent divided by the support of the antecedent [5]. The lift value can be calculated using equation (3).

$$\text{Support}(X \rightarrow Y) = \frac{Count(XUY)}{N} \quad \text{(eq 1)}$$

$$\text{Confidence}(X \rightarrow Y) = \frac{P(XUY)}{P(X)} \quad \text{(eq 2)}$$

$$\text{Lift}(X \rightarrow Y) = \frac{Confidence(X \rightarrow Y)}{P(Y)} \quad \text{(eq 3)}$$

3 PREVIOUS WORK

The algorithms to mine association rules generate large number of patterns and it is not feasible for an expert to go through all possible patterns to assess them. The support and confidence acts as a primary and secondary filter in rule mining [6]. One of the methods used to cope up with such an amount of output depends on using association rule interestingness measure. Finding interesting patterns has become one of the important criteria in the field of association rule mining. In [7], it is stated that many measures were proposed to determine the interestingness of association patterns but they provide conflicting information about the interestingness of a pattern. This paper suggests to consider several key properties in order to select the right measure for a given application domain and the measures with their properties are shown in Figure 1.

The interestingness measures may not produce all the interesting patterns. Some patterns produced by these measures may not be interesting to the user. Most of the objective interestingness measures can be transformed to subjective measures by replacing the average probability by the expected probability.

Symbol	Measure	Range	P1	P2	P3	O1	O2	O3	O3'	O4
ϕ	ϕ-coefficient	$-1\cdots0\cdots1$	Yes	Yes	Yes	Yes	No	Yes	Yes	No
λ	Goodman-Kruskal's	$0\cdots1$	Yes	No	No	Yes	No	No*	Yes	No
α	odds ratio	$0\cdots1\cdots\infty$	Yes*	Yes	Yes	Yes	Yes	Yes*	Yes	No
Q	Yule's Q	$-1\cdots0\cdots1$	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
Y	Yule's Y	$-1\cdots0\cdots1$	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No
κ	Cohen's	$-1\cdots0\cdots1$	Yes	Yes	Yes	Yes	No	No	Yes	No
M	Mutual Information	$0\cdots1$	Yes	Yes	Yes	No**	No	No*	Yes	No
J	J-Measure	$0\cdots1$	Yes	No	No	No**	No	No	No	No
G	Gini index	$0\cdots1$	Yes	No	No	No**	No	No*	Yes	No
s	Support	$0\cdots1$	No	Yes	No	Yes	No	No	No	No
c	Confidence	$0\cdots1$	No	Yes	No	No**	No	No	No	Yes
L	Laplace	$0\cdots1$	No	Yes	No	No**	No	No	No	No
V	Conviction	$0.5\cdots1\cdots\infty$	No	Yes	No	No**	No	No	Yes	No
I	Interest	$0\cdots1\cdots\infty$	Yes*	Yes	Yes	Yes	No	No	No	No
IS	Cosine	$0\cdots\sqrt{P(A,B)}\cdots1$	No	Yes	Yes	Yes	No	No	No	Yes
PS	Piatetsky-Shapiro's	$-0.25\cdots0\cdots0.25$	Yes	Yes	Yes	Yes	No	Yes	Yes	No
F	Certainty factor	$-1\cdots0\cdots1$	Yes	Yes	Yes	No**	No	No	Yes	No
AV	Added value	$-0.5\cdots0\cdots1$	Yes	Yes	Yes	No**	No	No	No	No
S	Collective strength	$0\cdots1\cdots\infty$	No	Yes	Yes	Yes	No	Yes*	Yes	No
ζ	Jaccard	$0\cdots1$	No	Yes	Yes	Yes	No	No	No	Yes
K	Klosgen's	$(\frac{2}{\sqrt{3}}-1)^{1/2}[2-\sqrt{3}-\frac{1}{\sqrt{3}}]\cdots0\cdots\frac{2}{3\sqrt{3}}$	Yes	Yes	Yes	No**	No	No	No	No

where: P1: $O(\mathbf{M})=0$ if $det(\mathbf{M})=0$, *i.e.*, whenever A and B are statistically independent.
P2: $O(\mathbf{M_2}) > O(\mathbf{M_1})$ if $\mathbf{M_2} = \mathbf{M_1} + [k\ -k;\ -k\ k]$.
P3: $O(\mathbf{M_2}) < O(\mathbf{M_1})$ if $\mathbf{M_2} = \mathbf{M_1} + [0\ k;\ 0\ -k]$ or $\mathbf{M_2} = \mathbf{M_1} + [0\ 0;\ k\ -k]$.
O1: Property 1: Symmetry under variable permutation.
O2: Property 2: Row and Column scaling invariance.
O3: Property 3: Antisymmetry under row or column permutation.
O3': Property 4: Inversion invariance.
O4: Property 5: Null invariance.
Yes*: Yes if measure is normalized.
No*: Symmetry under row or column permutation.
No**: No unless the measure is symmetrized by taking $\max(M(A,B), M(B,A))$.

Figure 1 Properties of interestingness measures [7]

Several factors like disjunct size, the imbalance of the class distribution, attribute costs, misclassification costs and asymmetry in classification rules are often neglected which influence the interestingness of the rule. A new criterion called attribute surprisingness is introduced and these factors are discussed in [8].

The objective measures indicate the support and degree of correlation of a pattern for the data considered. They automatically remove the obvious rules. "The objective measure is based only on the raw data and no knowledge of the user or application is required" [9]. The subjective measures take the knowledge of the user who uses the data into account. The semantic measures are special type of subjective measures that considers the semantics and explanations of the patterns. [9].

A rule selection framework was proposed in "Structure-Based Rule Selection Framework for Association Rule Mining of Traffic Accident Data" which classifies, selects and filters the rules based on the rule structures. This framework consists of semantic rule classification and permutation analysis. The

semantic rule classification involves classifying the rules to candidate, strongly, abundant and weakly abundant while the permutation analysis filters the equivalent but less significant ones and the rules that cover the other rules are selected and the ones being covered are discarded [10].

In [11] and [12], the association rules generated from the crash data are filtered using the measures support, confidence and lift. The lift is considered as the parameter in order to obtain interesting rules. The rules with higher lift values were considered as interesting rules. We have used the same measures in obtaining the association rules from the MASP blocks.

The traffic safety has become one of the highest priorities of the government. Black spots and black zones in terms of crash data and location characteristics are determined in [6]. Patterns at high frequency accident locations are obtained and compared with the patterns at low frequency accident locations. "The strength of this approach lies within the identification of relevant variables that make a strong contribution towards a better understanding of accident circumstances and the discerning of descriptive accident patterns from more discriminating accident circumstances to profile black spots and black zones" [6]. The interestingness measure used is $I = \frac{S_h - S_l}{\max\{S_l, S_h\}}$ where S_h and S_l is the support of rules in high frequency accident locations and low frequency accident locations respectively [6].

An association rule that contains more than one dimension or predicate is known as multidimensional association rule. In [13], the multidimensional association rules were found in the crash records which could mine the conditional factors of the traffic accidents.

Piatetsky-Shapiro's properties aim at specifying what a good measure is. The three key properties of Piatetsky-Shapiro for a Rule $X \rightarrow Y$ and Measure M are as follows [7].

- P1: M = 0 if X and Y are statistically independent;
- P2: M monotonically increases with P(X, Y) when P(X) and P(Y) remain the same;
- P3: M monotonically decreases with P(X) (or P(Y)) when the rest of the parameters (P(X, Y) and P(Y) or P(X)) remain unchanged.

The transactions that do not contain item X (antecedent) and item Y (consequent) has no influence on the result. This is called null invariance property and the null invariant measures are in Figure 2 [14].

Measure	Definition	Exponent
$AllConf(a,b)$	$\min\{P(a\|b), P(b\|a)\}$	$k \rightarrow -\infty$
$Coherence(a,b)$	$(P(a\|b)^{-1} + P(b\|a)^{-1} - 1)^{-1}$	$k = -1$
$Cosine(a,b)$	$\sqrt{P(a\|b)P(b\|a)}$	$k \rightarrow 0$
$Kulc(a,b)$	$(P(a\|b) + P(b\|a))/2$	$k = 1$
$MaxConf(a,b)$	$\max\{P(a\|b), P(b\|a)\}$	$k \rightarrow +\infty$

Figure 2 Null invariant measures [14]

Some methods were proposed in selecting appropriate interestingness measures like multi-criteria decision approach towards measure selection [15], the property matching [7], match expert ranking with the measure ranking [7]. In multi-criteria decision approach each measure is analyzed with respect to the properties and evaluated. The intuition of property matching approach is there is no measure that is consistently better than others in all application domains. This is because different measures have different intrinsic properties, some of which may be desirable for certain applications but not for others. Thus in order to find the right measure, we need to match the desired properties of an application against the properties of the existing measures. In matching expert ranking with the measure ranking t is practically not possible for an expert to rank all the tables manually, so a smaller set of contingency tables are given to the experts for ranking and use this information to determine the most appropriate measure.

4 METHOD

The main objective of the proposed method is to obtain MASPs and conduct the pattern mining over the block of each MASP. In construction of a MASP, the item-space is searched to find the most frequent item I_{MAX}. The I_{MAX} is added to the MASP under construction if it satisfies the two conditions as 1) minimum support and 2) association strength threshold. The formation of the MASP is finalized when no item is found in the item-space of the last level.

The detected MASPs are further utilized to obtain sub-sets of the data table. These sub tables are further mined to generate association rules. We have used modified version of the FP-Tree proposed in [1] to generate frequent patterns and the K-Pattern tree proposed by Dr. Soysal to generate rules. Figure 3 shows the overall framework of association rule mining by MASPs. The framework is composed of four main steps as

- Data preprocessing
- Detecting MASPs
- Generating patterns
    - Generating FP-Tree
    - Generating K-Pattern tree
- Finding association rules

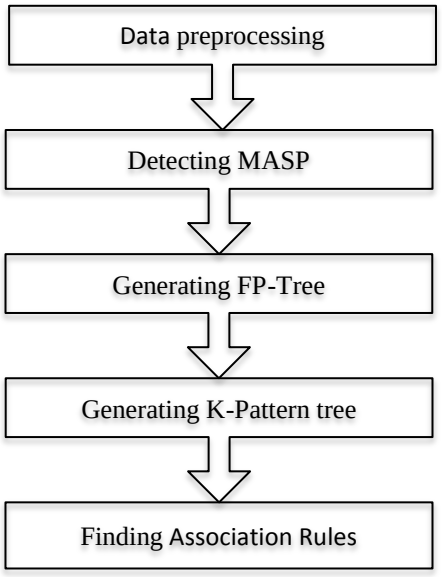


Figure 3 MASP Framework

4.1 Data Preprocessing

Association rule mining uses discrete data; hence the input data must be preprocessed so that the continuous data is discretized. After the discretization, the records are encoded to speed up the search process and to reduce memory allocation cost.

In our application, the user will be able to get the data for the selected attributes from the source data table and will be able to discretize the numeric data. By default, the bin type is set to default. The user can either give the total number of bins, or give the bins directly.

Example: Let us consider the values of an attribute A={62.5, 76.6, 72.4, 15.3}, Bins={0, 10, 20, 30, 40, 50, 60, 70, 80, 90}. When a value lies between *x* and *y* then the value is replaced with *x*. Therefore the continuous data is discretized as A_dis= {60, 70, 70, 10}.

Once the bins are set, the raw data is converted into Bin data. The Bin data is encoded by giving unique id to each discrete value starting from 1; Figure 4 illustrates the discretization process.

A1	A2	A3
a	c	62.5
a	d	76.5
b	d	72.5
c	e	15.3

Raw Data

A1	A2	A3
a	c	60
a	d	70
b	d	70
c	e	10

Bin Data

A1	A2	A3
1	4	7
1	5	8
2	5	8
3	6	9

Encoded Data

Figure 4 Converting raw data to bin data and bin data to encoded data

4.2 Detecting MASPs

In the second step, all of the most associated sequential patterns are detected. A frequent item is decided to be most associated if it passes the association strength threshold whose range is [0, 1]. This threshold is equal to or greater than the minimum confidence. In sequel, we explain MASP detection process.

Let $MASP_1 = \{\}$ and the occurrence of the item V_{11} is higher than the others. If the occurrence of V_{11} is greater than or equal to the association strength threshold, then $I_{MAX} = V_{11}$ is added to the MASP resulting in $MASP_1 = (V_{11})$. As seen in Figure 5, each I_{MAX} divides its parent's block into two children data set as the *block* and the *counter-block*. Similarly $\overline{V}_{11}$ is checked whether it should be added to the MASP or not.

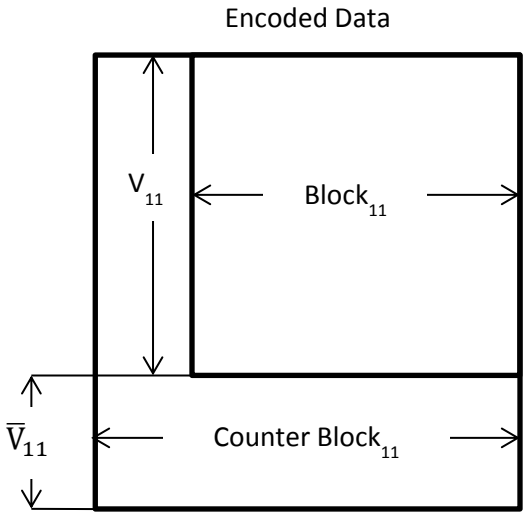


Figure 5 Blocks of $MASP_1$ at the level 1, where the $MASP_1$=(V11)

The formation of the $Block_{11}$ is a resultant of a query. This query consists of select clause, where clause and the data source of encoded data. The select clause includes the attributes other than the attributes of the items of the form "attribute = value" in the MASP's where-clause. The where clause includes conjunction of the items in MASP. Let's assume that the occurrence of V_{41} in the block $Block_{11}$ satisfies the MASP's thresholds, so V_{41} is added to the $MASP_1$. Similarly the block and counter blocks for V_{41} are formed as seen in Figure 6 $MASP_1 = (V_{11}, V_{41})$.

If the occurrence of $\overline{V}_{41}$ in the block $Block_{11}$ satisfies the MASP's thresholds, $\overline{V}_{41}$is added to a new $MASP_2 = (V_{11}, \overline{V}_{41})$.

This process continues until there is no data left or until a set of records (block) with no item satisfies the MASP's thresholds is found. As a remark, a block is obtained from the encoded data table using the most associated sequential pattern.

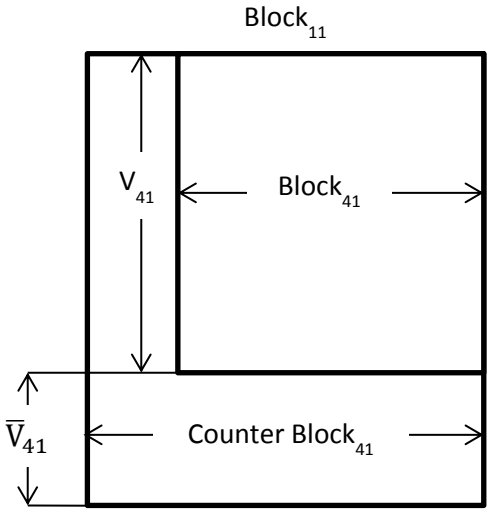


Figure 6 MASP blocks at the level k=2, where the $MASP_1 = (V_{11}, V_{41})$

The final blocks obtained from each MASP are individually mined to generate association rules. Once the final blocks are generated, each block is processed to find K-Patterns. In order to generate patterns, we need to first form the FP- tree. "A large database is compressed into a condensed, smaller data structure, FP-tree which avoid the costly, repeated database scans" [1]. In our FP- tree construction we do not use the frequent item header table. We have proposed a different approach to generate K-patterns from the FP-tree which is discussed in the algorithms section of this paper.

From the K- frequent patterns, the possible combinations of antecedent and consequent are formed and checked if they satisfy the minimum support and confidence given by the user in order to generate association rules. The algorithm proposed is discussed below.

Algorithm Detecting MASPs:

Inputs:

- Data table D to be mined, the attribute set A, and set of classes of an attribute V_i as defined in section 1.4 Definitions.
- The association strength threshold, τ.
- The threshold S_{min} to decide whether an item is frequent.

Output:

The set of MASPs.

Steps:

A) Initialize
   1. P = {Q(D)} //set of all candidate block queries to be processed
   2. M = {} //Set of all block queries to be mined

B) Repeat until P = {}:
   1. CanB = P(1) //Candidate-block
   2. P = P \ P(1) //First in first out
   3. Within CanB , obtain distributions S_i={*Count*$_1$, *Count*$_2$, ..} of each A_i and the set S={S_1, S_2, .., $S_{|A|}$}, i = 1, 2, .., $|V_i|$, in CanB.
   4. Obtain the set $S_{maxA} = \{\max\{S_i\} \geq S_{min}\}$ from S
   5. If $S_{maxA} \neq \{\}$, //that is CanB does not have any frequent itemset
      5.1. Find $S_{max}=\{\max\{S_{maxA}\}\}$ and obtain the most associated item I_{max}.
      5.2. If Attribute_Name(last item of CanB) = Attribute_Name(I_{max})
         5.2.a. Remove the last item of the CanB
      5.3. If $S_{max} \geq S_O$ (= τ_O |CanB|) //association strength threshold condition
         5.3.a. Form block B_{max} of the item I_{max}

5.3.b. If $|B_{max}| > 0$
   i. Add B_{max} to P
5.3.c. Else
   i. Add B_{max} to M

5.3.d. Form the counter-block CrB of I_{max}
5.3.e. If $|CrB| \geq S_{min}$
   i. If $\tau_O > 0.5$ Or $S_{max} > |CanB| (1 - \tau_O)$ //Under-representing condition as a short-cut
      1. Add CrB to M
   ii. Else
      1. Add CrB to P
5.3.f. Else
   i. Indicate that the block will not be mined
   ii. Add CrB to M

5.4. Else
5.4.a. Add CanB to M

6. Else
6.1. Check if the Candidate block is a block or a counter block
6.1.a. If CanB's last item is of the form (Attribute ≠ Value)
   i. Indicate that the block will not be mined
   ii. Add CanB to M
6.1.b. Else
   i. Add Parent block of the candidate block to M

4.2.1 Detail Explanation of the Algorithm Detecting MASP

The algorithm is discussed with an illustration. Let us consider a data table D as represented in Figure 7. Minimum support = 20% ≡ 2 records, and association strength threshold = 30% ≡ 3 records. Initialize the set of candidate blocks P= {D} and the set of MASPs M = {}. Since P is not empty, the candidate block, CanB = P(1) = D, P= P \ P(1) ={}. According to step B.3, the distribution of each item in the candidate block is obtained as shown in
Figure 8.

A1	A2	A3	A4	A5
V11	V21	V31	V41	V51
V11	V21	V32	V41	V53
V11	V22	V31	V42	V52
V11	V21	V32	V42	V52
V11	V22	V32	V43	V52
V11	V22	V32	V43	V52
V12	V22	V31	V44	V51
V12	V22	V31	V44	V51
V12	V23	V31	V45	V51
V12	V23	V32	V46	V51

Figure 7 Data table

Item	Frequency
A1=V11	6
A1=V12	4
A2=V21	3
A2=V22	5
A2=V23	2
A3=V31	5
A3=V32	5
A4=V41	2
A4=V42	2
A4=V43	2
A4=V44	2
A4=V45	1
A4=V46	1
A5=V51	5
A5=V52	4
A5=V53	1

Figure 8 Item distribution of the candidate block

S_{maxA} is obtained as in step B.4, S_{maxA} = { N(V11)= 6, N(V12) = 4, N(V21) = 3, N(V22) = 5, N(V23) = 2, N(V31) = 5, N(V32) = 5, N(V41) = 2, N(V42) = 2, N(V43) = 2, N(V44) = 2, N(V51) = 5, N(V52) = 4}, where N(X) gives the number of transactions having X. Since $S_{maxA} \neq \{\}$, S_{maxA} ={max{S_{maxA}}} ={ N(V11) =6}. I_{max} = V11. Step B.5.2 indicates whether the last item of the CanB MASP's attribute is same as the V11's attribute name, i.e. A1. Since MASP is currently empty, we can proceed to step B.5.3 that involves checking if S_{max} is greater than or equal to association strength threshold, S_0. S_0= 0.3×|CanB|= 0.3×10=3 records. From this it is seen that $S_{max} > S_0$. According to step B.5.3.a form a block B_{max} from item I_{max}. The first step to form a block is adding I_{max} to the predicate part (MASP) of the query and the second step is adding all attributes of the candidate block, CanB except the attribute of the item, A1 to the select clause. So the query of Bmax obtained is "SELECT A2, A3, A4, A5 FROM encoded data table WHERE A1=V11". Thus, the block B_{max} is the set of transactions formed by querying the encoded data table with the above query. Since $|B_{max}| >0$, we add the query of B_{max} to P as shown in Figure 9.

A2	A3	A4	A5
V21	V31	V41	V51
V21	V32	V41	V53
V22	V31	V42	V52
V21	V32	V42	V52
V22	V32	V43	V52
V22	V32	V43	V52

B_{max}

	QUERY
P(1)	Select A2, A3, A4, A5 from encoded data table where A1=V11

Figure 9 B_{max} and P

Now, we have to form the counter block, see step B.5.3.d in the algorithm. The first step to form the counter block is adding I_{max} to the predicate (MASP) of the query with inequality, i.e. A1≠ V11 and the second step is adding all the attributes of the candidate block CanB. Thus, the counter block CrB is the set of transactions formed by querying the encoded data table with the query "Select A1, A2, A3, A4, A5 from encoded data table where A1≠V11". According to Step B.5.3.e |CrB |=4 is greater than minimum support = 2, and τ_O =0.3 which is less than 0.5, so we add the counter block CrB to P as shown in Figure 10.

A1	A2	A3	A4	A5
V12	V22	V31	V44	V51
V12	V22	V31	V44	V51
V12	V23	V31	V45	V51
V12	V23	V32	V46	V51

CrB

	QUERY
P(1)	Select A2, A3, A4, A5 from encoded data table where A1=V11
P(2)	Select A1, A2, A3, A4, A5 from encoded data table where A1≠V11

Figure 10 Counter block and P

According to step B, repeat the logic until P is empty. CanB = P(1) = Select A2, A3, A4, A5 from encoded data table where A1=V11. P=P \ P(1). Refer Figure 11, to find the updated candidate block, CanB and P.

A2	A3	A4	A5
V21	V31	V41	V51
V21	V32	V41	V53
V22	V31	V42	V52
V21	V32	V42	V52
V22	V32	V43	V52
V22	V32	V43	V52

CanB

	QUERY
P(1)	Select A1,A2, A3, A4, A5 from encoded data table where A1≠V11

Figure 11 Candidate block and P

Within the CanB it is seen that S_{max} = 4. I_{max} can be either V32 or V52. Let us consider I_{max} as A3= V32. Since $S_{max} > S_0$, we form the block. Since the $|B_{max}| >0$, we add B_{max} to P. The counter block of I_{max} is obtained. Since $|CrB| = S_{min}$ and $\tau_O <0.5$ we add counter block into P as shown in Figure 12.

A2	A4	A5
V21	V41	V53
V21	V42	V52
V22	V43	V52
V22	V43	V52

B_{max}

Figure 12 Block, counter block and P

	QUERY
P(1)	Select A1,A2, A3, A4, A5 from encoded data table where A1≠V11
P(2)	Select A2, A4, A5 from encoded data table where A1=V11, A3=V32

A2	A3	A4	A5
V21	V31	V41	V51
V22	V31	V42	V52

CrB

	QUERY
P(1)	Select A1,A2, A3, A4, A5 from encoded data table where A1≠V11
P(2)	Select A2, A4, A5 from encoded data table where A1=V11, A3=V32
P(3)	Select A2, A3, A4, A5 from encoded data table where A1=V11, A3≠V32

Figure 12 continued

P is not empty, so CanB = P(1) = Select A1,A2, A3, A4, A5 from encoded data table where A1≠V11. P = P \ P(1), refer to Figure 13.

A1	A2	A3	A4	A5
V12	V22	V31	V44	V51
V12	V22	V31	V44	V51
V12	V23	V31	V45	V51
V12	V23	V32	V46	V51

CanB

	QUERY
P(1)	Select A2, A4, A5 from encoded data table where A1=V11, A3=V32
P(2)	Select A2, A3, A4, A5 from encoded data table where A1=V11, A3≠V32

Figure 13 Candidate block and P

S_{max} =4, Imax is A1=V12. It is seen that $S_{max} > S_0$. According to step B.5.2 the attribute of last item of MASP is same as the attribute of I_{max}, so we have to remove the last item of MASP and then form B_{max}, refer Figure 14.

A2	A3	A4	A5
V22	V31	V44	V51
V22	V31	V44	V51
V23	V31	V45	V51
V23	V32	V46	V51

B_{max}

Figure 14 Block, P and M

	QUERY
P(1)	Select A2, A4, A5 from encoded data table where A1=V11, A3=V32
P(2)	Select A2, A3, A4, A5 from encoded data table where A1=V11, A3≠V32
P(3)	Select A2, A3, A4, A5 from encoded data table where A1=V12

	QUERY
M(1)	Don't Mine - Select A1, A2, A3, A4, A5 from encoded data table where A1≠V11, A1≠V12

Figure 14 continued

Counter Block is formed according to step B.5.3.d and it is seen that there are no rows in counter block. So we add the Counter to M indicating that the candidate block will not be mined. Since P is not empty, CanB = P(1) = Select A2, A4, A5 from encoded data table where A1=V11, A3=V32. P=P \ P(1). Within the CanB it is seen that S_{max} = 3. I_{max} is A5=V52. Since $S_{max} > S_0$, we form the block. Since |Bmax| >0, add $|B_{max}|$ to P. Form the counter-block CrB of Imax. Since | CrB| < S_{min}, indicate that the counter block will not be mined and add CrB to M, see Figure 15.

A2	A4	A5
V21	V41	V53
V21	V42	V52
V22	V43	V52
V22	V43	V52

CanB

	QUERY
P(1)	Select A2, A3, A4, A5 from encoded data table where A1=V11, A3≠V32
P(2)	Select A2, A3, A4, A5 from encoded data table where A1=V12

A2	A4
V21	V42
V22	V43
V22	V43

B_{max}

Figure 15 Candidate block, block , counter block, P and M

	QUERY
P(1)	Select A2, A3, A4, A5 from encoded data table where A1=V11, A3≠V32
P(2)	Select A2, A3, A4, A5 from encoded data table where A1=V12
P(3)	Select A2, A4 from encoded data table where A1=V11, A3=V32, A5 = V52

A2	A4	A5
V21	V41	V53

CrB

	QUERY
M(1)	Don't Mine - Select A1, A2, A3, A4, A5 from encoded data table where A1≠V11, A1≠V12
M(2)	Don't Mine-Select A2, A4, A5 from encoded data table where A1=V11, A3=V32, A5≠V52

Figure 15 continued

Since P is not empty. CanB = P(1) and P = P \ P(1). The CanB items are found to fall below association strength threshold, so add CanB to M according to step B.5.4, refer Figure 16.

A2	A3	A4	A5
V21	V31	V41	V51
V22	V31	V42	V52

CanB

	QUERY
P(1)	Select A2, A3, A4, A5 from encoded data table where A1=V12
P(2)	Select A2, A4 from encoded data table where A1=V11, A3=V32, A5 = V52

	QUERY
M(1)	Don't Mine - Select A1, A2, A3, A4, A5 from encoded data table where A1≠V11, A1≠V12
M(2)	Don't Mine-Select A2, A4, A5 from encoded data table where A1=V11, A3=V32, A5≠V52
M(3)	Select A2, A3, A4, A5 from encoded data table where A1=V11, A3≠V32

Figure 16 Candidate block, P and M

P is not empty, so CanB = P(1) and P = P \ P(1). Since $|B_{max}| > 0$, add $|B_{max}|$ to P. Within the CanB it is seen that $S_{max} = 4$. I_{max} is A5=V51. Since $S_{max} > S0$, we form the block. Since $|B_{max}| > 0$, add $|B_{max}|$ to P. The counter block has no rows, so indicate that the CrB will not be mined and add CrB to M, see Figure 17.

A2	A3	A4	A5
V22	V31	V44	V51
V22	V31	V44	V51
V23	V31	V45	V51
V23	V32	V46	V51

CanB

	QUERY
P(1)	Select A2, A4 from encoded data table where A1=V11, A3=V32, A5 = V52

A2	A3	A4
V22	V31	V44
V22	V31	V44
V23	V31	V45
V23	V32	V46

B_{max}

	QUERY
P(1)	Select A2, A4 from encoded data table where A1=V11, A3=V32, A5 = V52
P(2)	Select A2, A3, A4 from encoded data table where A1=V12, A5=V51

	QUERY
M(1)	Don't Mine - Select A1, A2, A3, A4, A5 from encoded data table where A1≠V11, A1≠V12
M(2)	Don't Mine-Select A2, A4, A5 from encoded data table where A1=V11, A3=V32, A5≠V52
M(3)	Select A2, A3, A4, A5 from encoded data table where A1=V11, A3≠V32
M(4)	Don't Mine-Select A2, A3, A4, A5 from encoded data table where A1=V12, A5≠V51

Figure 17 Candidate block, block, P and M

A2	A4
V21	V42
V22	V43
V22	V43

CanB

Figure 18 Candidate block, P and M

	QUERY
P(1)	Select A2, A3, A4 from encoded data table where A1=V12, A5=V51

	QUERY
M(1)	Don't Mine - Select A1, A2, A3, A4, A5 from encoded data table where A1≠V11, A1≠V12
M(2)	Don't Mine-Select A2, A4, A5 from encoded data table where A1=V11, A3=V32, A5≠V52
M(3)	Select A2, A3, A4, A5 from encoded data table where A1=V11, A3≠V32
M(4)	Don't Mine-Select A2, A3, A4, A5 from encoded data table where A1=V12, A5≠V51
M(5)	Select A2, A4 from encoded data table where A1=V11, A3=V32, A5 = V52

Figure 18 continued

P is not empty, so form the candidate block, CanB = P(1) and P = P \ P(1). The S_{max} =2 is obtained from the CanB in Figure 18. Since $S_{max} < S_0$, according to step 5.4, add CanB to M, see Figure 18.

It is seen that the P in Figure 18 is not empty, so CanB = P(1). P=P/P(1)={}. S_{max} =3, I_{max} is A3=V31. It is seen that S_{max} = S0, so form the block. Since |Bmax| >0, add |Bmax| to P. Form the counter block. Since |CrB| < S_{min}, indicate that the counter block will not be mined and add it to M, see Figure 19.

A2	A3	A4
V22	V31	V44
V22	V31	V44
V23	V31	V45
V23	V32	V46

CanB

A2	A4
V22	V44
V22	V44
V23	V45

B_{max}

	QUERY
P(1)	Select A2, A4 from encoded data table where A1=V12, A5=V51, A3=31

Figure 19 Candidate block, block, counter block, P and M

A2	A3	A4
V23	V32	V46

CrB

	QUERY
M(1)	Don't Mine - Select A1, A2, A3, A4, A5 from encoded data table where A1≠V11, A1≠V12
M(2)	Don't Mine-Select A2, A4, A5 from encoded data table where A1=V11, A3=V32, A5≠V52
M(3)	Select A2, A3, A4, A5 from encoded data table where A1=V11, A3≠V32
M(4)	Don't Mine-Select A2, A3, A4, A5 from encoded data table where A1=V12, A5≠V51
M(5)	Select A2, A4 from encoded data table where A1=V11, A3=V32, A5 = V52
M(6)	Don't Mine-Select A2, A3, A4 from encoded data table where A1=V12, A5=V51, A3≠31

Figure 19 continued

P is not empty, therefore form the candidate block, CanB = P(1) and P = P \ P(1). P is {}. S_{max}=2. Since S_{max} <S0, according to step B.5.4, add CanB to M, refer Figure 20.

A2	A4
V22	V44
V22	V44
V23	V45

CanB

	QUERY
M(1)	Don't Mine - Select A1, A2, A3, A4, A5 from encoded data table where A1≠V11, A1≠V12
M(2)	Don't Mine-Select A2, A4, A5 from encoded data table where A1=V11, A3=V32, A5≠V52
M(3)	Select A2, A3, A4, A5 from encoded data table where A1=V11, A3≠V32
M(4)	Don't Mine-Select A2, A3, A4, A5 from encoded data table where A1=V12, A5≠V51
M(5)	Select A2, A4 from encoded data table where A1=V11, A3=V32, A5 = V52
M(6)	Don't Mine-Select A2, A3, A4 from encoded data table where A1=V12, A5=V51, A3≠31
M(7)	Select A2, A4 from encoded data table where A1=V12, A5=V51, A3=31

Figure 20 Candidate block and M

Since P is {}, the final blocks to be mined and MASP is obtained. Each block in M is mined to generate association rules. Thus for each block FP-Tree is formed, frequent patterns are mined and association rules are generated.

4.3 Generating Frequent Pattern-Tree

There are various approaches in the field of association rule mining to find K-patterns. One of it is finding k-patterns from the FP-tree [1]. We have implemented the FP-tree without the header table. The header table consists of the items sorted and their frequency head that allows us to traverse the FP-tree to find patterns by following the link of each frequent item. This approach occupies memory and consumes more time to find the following link in the FP-tree. In order to avoid this, we directly traverse through each node of the FP-tree and form 2-tree with the items in it saved along with the linked list of address location which are further used to generate K-patterns, K>2. This address location contains the address of the node in the FP-tree which helps directly accessing the node in the FP-tree. Since we used a linked list, a continuous memory block is not required. The steps involved in the construction of FP-tree are discussed in the algorithm.

Algorithm FP-Tree Generation:
Input: Count array C(I1) (or support value S(I1)) of I1, data table D = {T(*z*)}. Ordered frequent 1-itemset I1 can be needed if the transaction items does not have a fix order; e.g., T(z) = (a,b,c) or T(z) = (b,a,c). In our implementation, we assume T(z) of D come in a fixed order.
Output: FP-Tree
Steps:

A. Initialize:

1. Create a root node of the tree RooT

B. Constructing branches:

1. Repeat until no transaction left //Traversing transactions T(z) of D
   - 1.1. Set *transaction cursor tc* to the next transaction
   - 1.2. Get the item counts
   - 1.3. Sort the items of T(*z*) //Item index for T(z, *j*=null)
   - 1.4. Set *current root cursor crc* as RooT
   - 1.5. Repeat until no item left // Traversing items of transaction T(*z*)
     - 1.5.1. Search CNs of the root pointed by *crc* whether item = *crc*(CN).item
     - 1.5.2. If SearchChilds(*) returns a CN
       - 1.5.2.1. Then increase CN.count by 1,
     - 1.5.3. Else
       - 1.5.3.1. Add a new child (CN) to the node pointed by *crc*
       - 1.5.3.2. Set CN.item = item and set CN.count = 1
     - 1.5.4. Point *crc* to CN

4.3.1 Detail Explanation of the Algorithm Generating FP-tree

The algorithm is illustrated with an example. Let us consider a block with set of transactions given in Figure 21 and minimum support as 3 records. The items that satisfy the minimum support are represented in Figure 22. According to step A, initialize the root as shown in Figure 23. Step B deals with the construction of branches. The following process is repeated until no transaction is left.

According to step B.1.1, set the transaction cursor tc to first transaction. The count of each is obtained from the Item-Value and frequency table. The order of the items sorted in the transaction is 1, 2, 4, 6, 8. Set the current cursor crc as root according to B.1.4 as shown in Figure 24.

A1	A2	A3	A4	A5	A6	A7	A8
1	2		4		6		8
1	2	3	4	5	6	7	
1				5		7	
	2			5			8
1	2	3	4		6		8

Figure 21 Data table

Item Value	Frequency
1	4
2	4
4	3
5	3
6	3
8	3

Figure 22 Item-Value and frequency table

.

Figure 23 Root initialization

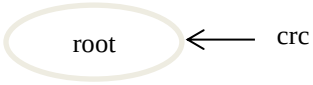


Figure 24 Point cursor to the root

For each item in the transaction, search the children nodes, CN of the root pointed by crc whether item is same as the current node item. Since the current node, crc is root and there are no children currently, we add a new child CN to the node pointed by crc. The child node's item is set to the item 1 and its count is set to 1. The child node is represented as "child node item : child node count", that is, 1:1 and Point crc to CN, refer to step B.1.5.4 as seen in Figure 25.

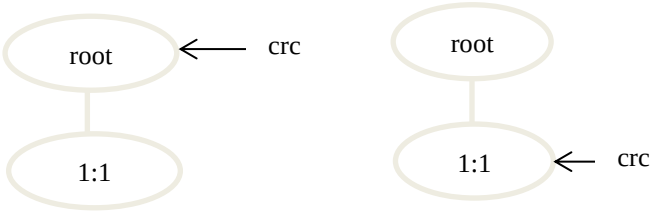


Figure 25 Point cursor to the current node

The same process is repeated for all the items and this process is illustrated stepwise in Figure 26.

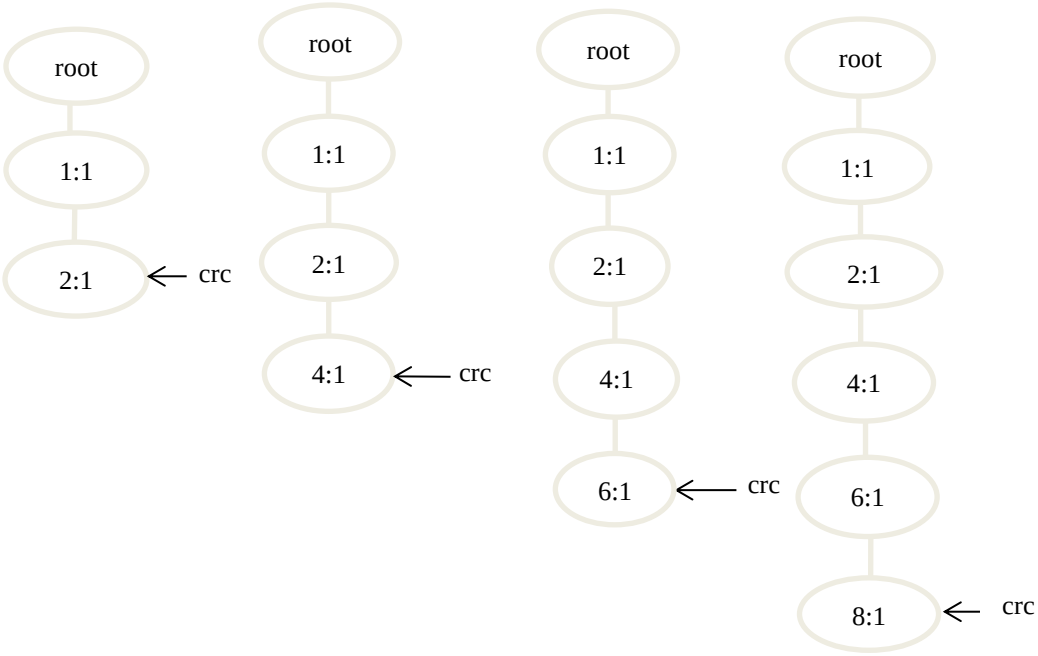


Figure 26 Stepwise illustration of adding nodes

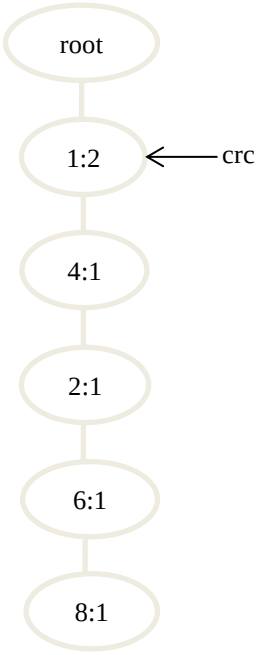


Figure 27 Adding nodes for next transaction

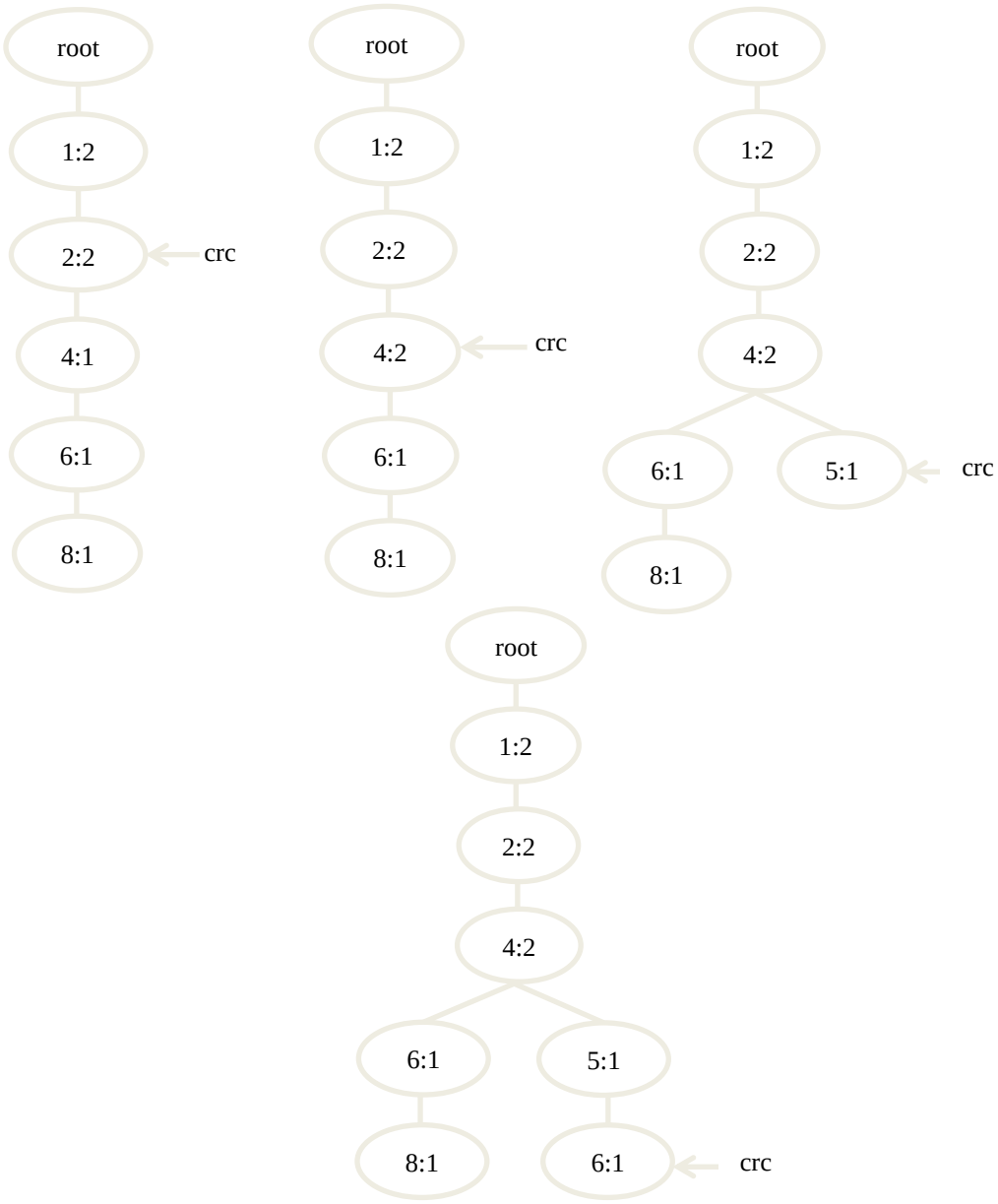


Figure 28 Illustration for adding nodes

All the items in the first transaction are processed so we go to the next transaction, refer to the step B.1.1. The count of each item is retrieved from the Item-Value and frequency table and the items are sorted according to the count as 1, 2, 4, 5, and 6. The crc is set to root and for every item in the transaction search the children nodes, CN of the root pointed by crc whether item = crc(CN) item. The item is 1 and it is the CN of the crc. Therefore increase its count by 1 and point the crc to CN as seen in Figure 27. Similarly the other items are processed which are illustrated in Figure 28. This process continues until no transaction is left and the FP-Tree is obtained as in Figure 29.

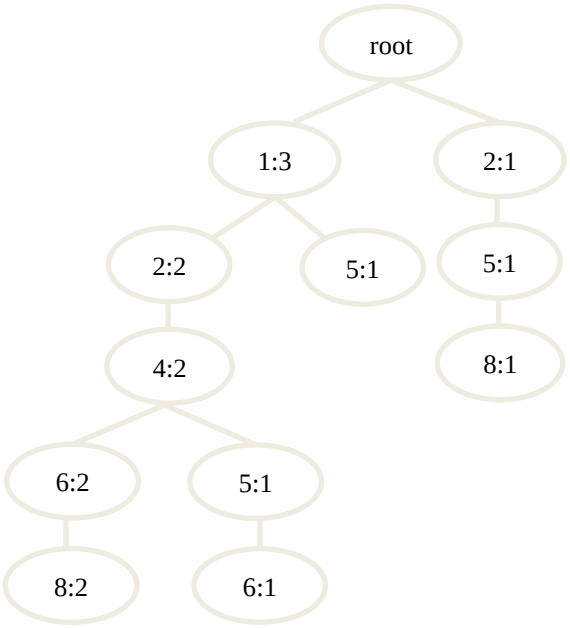


Figure 29 FP-tree obtained

From this FP-Tree obtained, the Pattern trees are generated.

4.4 Generate K-Pattern Tree

The K-patterns are generally obtained from the FP-tree. Since our FP-tree does not have the header table, we have proposed a new approach to find the K-patterns. The patterns are stored in a tree like structure.

Algorithm to generate K-pattern trees:
Input: FP-tree.
Output: k-Pattern Tree.
Steps:

A. Initialize:

2. Create the 2-pattern Tree's root root2P.
3. Add all children of the FP-Tree's root rootFP to the List

B. Constructing 2-Patterns Tree:

1. Repeat until no item left in the List
   - 1.1. Read the first item in the List as the *current node of the FP-tree* (cnFP),
   - 1.2. Delete the first item from the List.
   - 1.3. If cnFP is not a child of root2P, then
     - 1.3.1. Add it as a child to the root2P.
   - 1.4. Set the parent of cnFP as the *current ancestor node* (caFP)
   - 1.5. Repeat until caFP reaches rootFP
     - 1.5.1. Find root2P's child (as the parent2P) whose name matches the name of caFP
     - 1.5.2. Add cnFP as a child to the parent2P
     - 1.5.3. Set the parent of caFP as the new caFP

1.6. Add child (or children) of cnFP to the List at the position where cnFP is deleted.

2. Delete the parent node that has no children.

C. Constructing k-Patterns Tree (k>2):

1. Create the k-pattern Tree's root rootKP.
2. Create an empty List.
3. Repeat for all children pK_1 of the (k-1)-Pattern Tree
   - 3.1. Repeat for all children cK_1 of pK_1,
     - 3.1.1. If cK_1 satisfies the min support, then
       - 3.1.1.1. Add (pK_1.name, cK_1.name) as pK to the rootKP
       - 3.1.1.2. Repeat for all items nK_1 in the list of cK_1
         - 3.1.1.2.1. Add all children of the cK_1 of the FP-tree to the List
         - 3.1.1.2.2. Repeat until no item left in the List
           - 3.1.1.2.2.1. Read the first item in the List
           - 3.1.1.2.2.2. If the item is a child of pK
             - 3.1.1.2.2.2.1. Add the item to this child's list as a new node
           - 3.1.1.2.2.3. Else
             - 3.1.1.2.2.3.1. Add the item as a new node to the pK
           - 3.1.1.2.2.4. Add child (or children) of the node in FP-tree to the List
           - 3.1.1.2.2.5. Delete the first item from the List
       - 3.1.1.3. if pK has no child, then delete pK.

4.4.1 Detail Explanation of the Algorithm Generate K-Pattern Tree

The frequent patterns are generated in the form of tree structure. A sample FP-Tree is shown Figure 30. Each child with its parent is a pattern. The algorithm is illustrated with an example. Let us generate the patterns from the FP-Tree given in Figure 31.

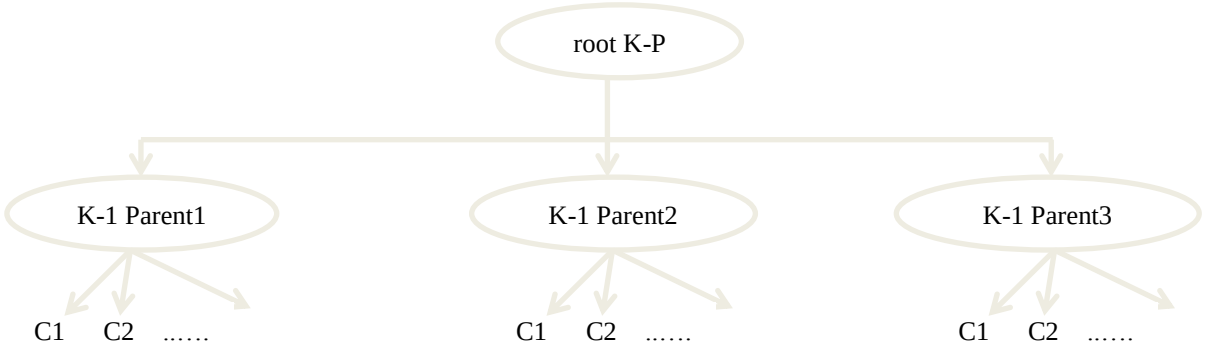


Figure 30 Basic structure of K-pattern tree

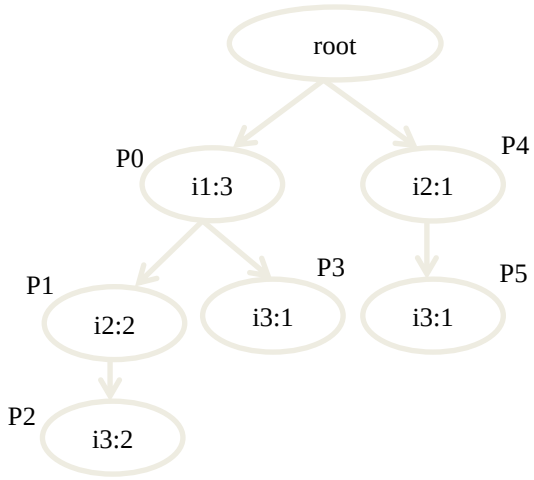


Figure 31 FP-tree with child positions

Initialize the root of the 2P-tree and add all the children of the FP-Tree root to the List. The list contains the items in the form address : item : frequency, refer to Figure 32.

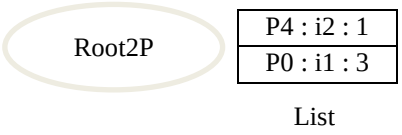


Figure 32 2P-tree and List

4.4.1.1 **Construction of 2P-tree, refer to section D in the algorithm**

Repeat until no item is left in the list. Read the first item of the list from bottom as the current node of the FP-tree cnFP. Set cnFP = P0 : i1 : 3. Delete the first item from the List. If cnFP is not a child of root2P, then add it as a child to the root2P, refer to Figure 33. Add the child of cnFP to the list at the position where cnFP is deleted, refer Figure 34.

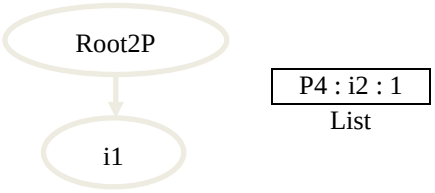


Figure 33 2P-tree and List

P4 : i2 : 1
P3 : i3 : 1
P1 : i2 : 2

List

Figure 34 List

According to step B.1, we have to repeat the process until there are no items left in the List. Read the first item of the List from the bottom as cnFP and delete it from the list. cnFP = P1 : i2 : 2. Since cnFP is not the child of root2P, add it as a child to the root2P, as in Figure 35.

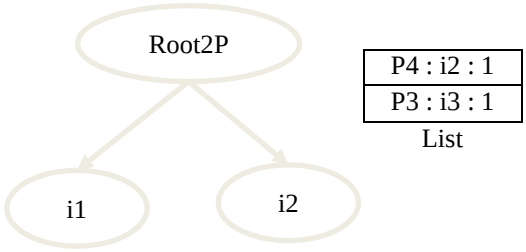


Figure 35 2P-tree and List

Step B.1.4, indicates to set caFP as P0: i3: 3. Step B.1.5, indicates to repeat the following until caFP reaches root. Find root2P's child (as the parent2P) whose name matches the name of caFP , add cnFP as a child to the parent2P and set the parent of caFP as the new caFP. Now caFP = root2P. So exit the loop at step B.1.5 and add the child of cnFP to the list, refer Figure 36. Since the list is not empty, the same procedure is repeated which is shown in Figure 37.

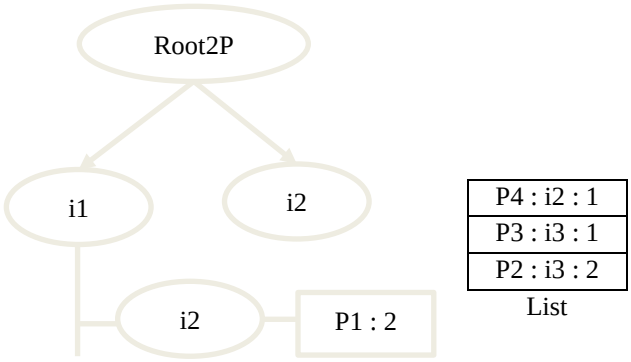
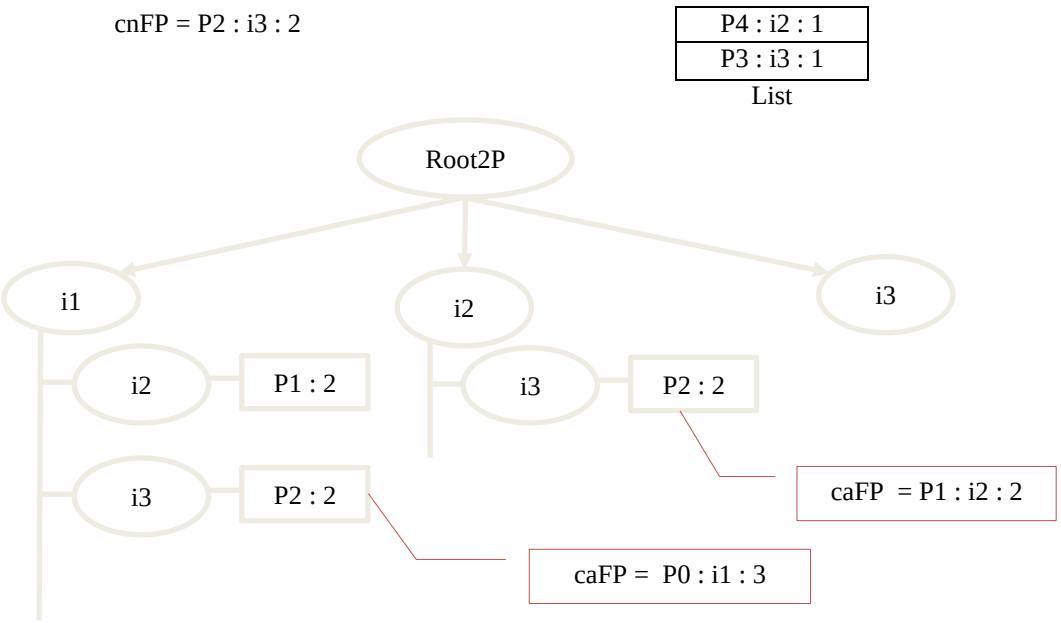
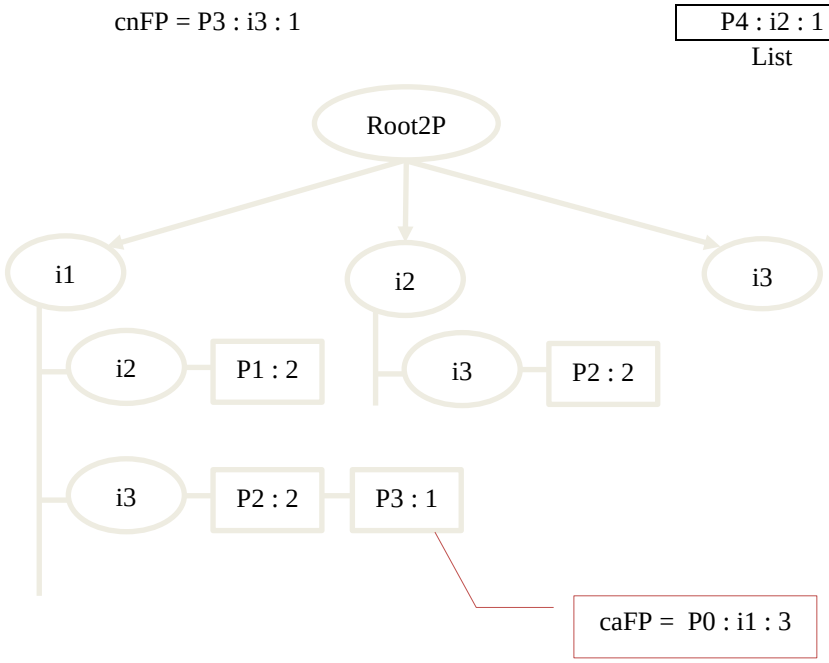


Figure 36 2P-tree and List



(a)

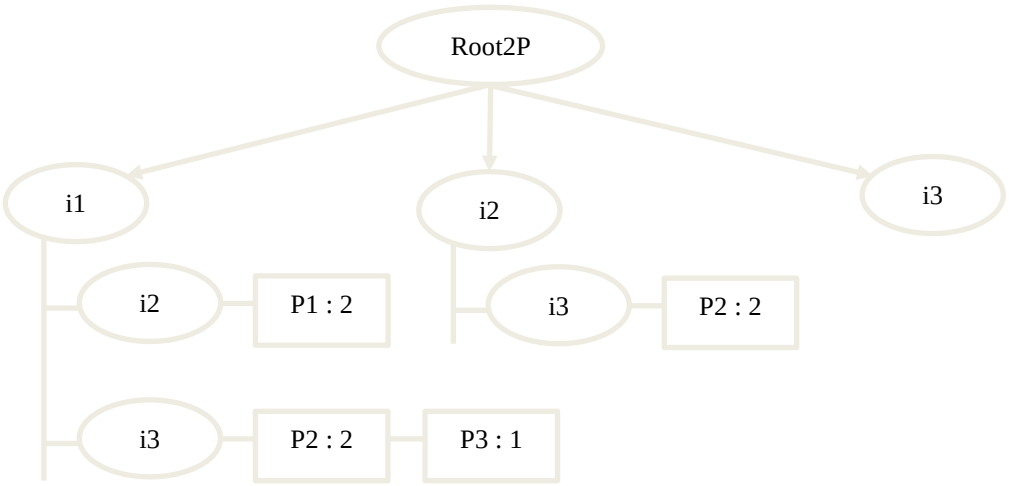


caFP = root

(b)

Figure 37 Stepwise construction of 2P-tree

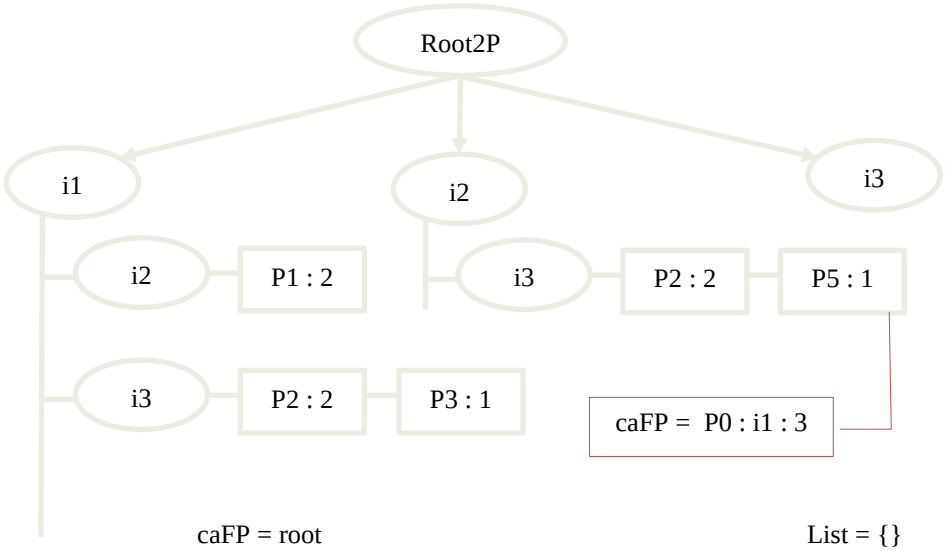
cnFP = P4 : i2 : 1 List = {}
caFP = root



P5 : i3 : 1
List

(c)

cnFP = P4 : i2 : 1 List = {}



caFP = root List = {}

(d)

Figure 37 continued

Step 2 in the algorithm indicates to delete the parent node that has no children. Since i3 has no children we can remove it. The final 2P-tree is Figure 38.

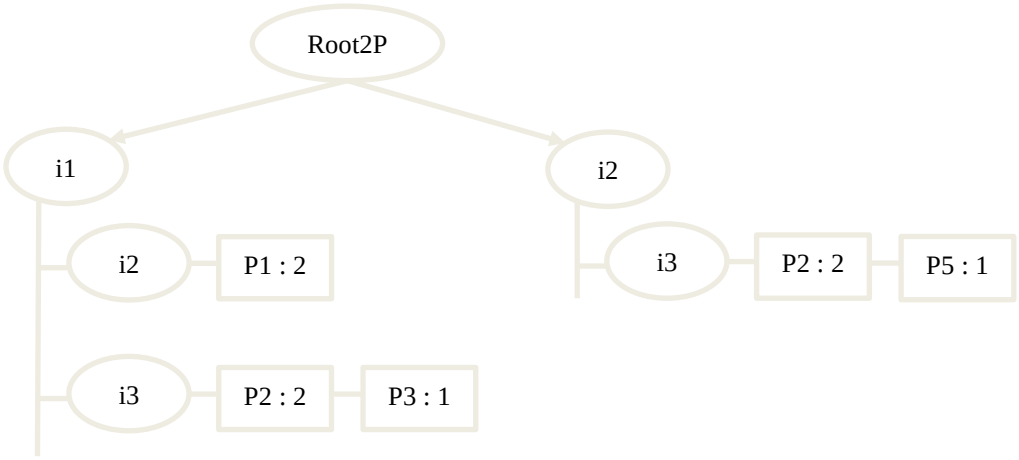
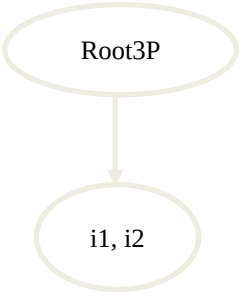


Figure 38 Final 2P-tree

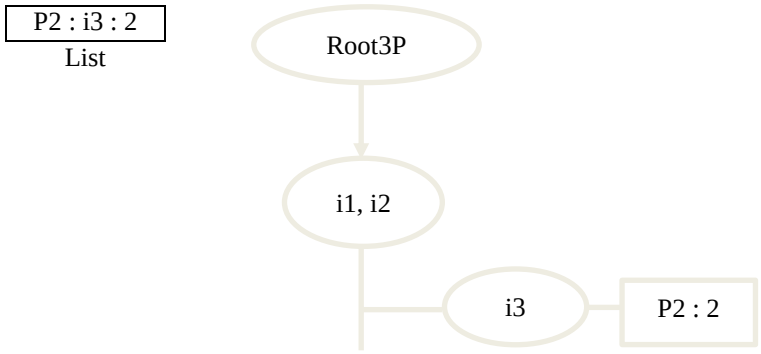
4.4.1.2 **Construction of 3P-tree, refer to section E in the algorithm**

Step C explains the construction of K-pattern tree where K>2. The procedure involves creation of K+1 pattern tree using K-pattern tree. A Root3P is created and an empty list is created. Let us consider the minimum support as 2. According to step C.3 for each child of the parent, where parent is the child of Root2P in K-pattern tree , that is 2-pattern tree, if the child satisfies the minimum support add (parent name, child name) as parent pK to the Root 3P. It is seen that i2 satisfies the minimum support. Therefore a new parent node is added to Root3P as shown in Figure 39 (a). For each position of the child in the FP, tree, add all children of the child of the FP-tree to the List and repeat the following until no item is left in the list according to step C.3.1.1.2.2. The position P1 has only one child, P2 : i3: 2 and the position P4 has one child, P5 : i3 : 1. For P1, the item i3 is added into the list and read the first item in the list. Check if the item is a child of the current parent. Since it is not, add a new child i3 with its address as shown in Since P2 has no children, no further items are added into the list. Remove the first item. For P4, the item i3 is added into the list and read the first item in the list. Check if the item is a child of the current parent. Since it is, just increase its count and add its position as shown in Figure 39 (b). Since P2 has no children, no further items are added into the list. Remove the first item. For P4, the item i3 is added into the list and read the first item in the list. Check if the item is a child of the current parent. Since it is, just increase its count and add its position as shown in Figure 39 (c).

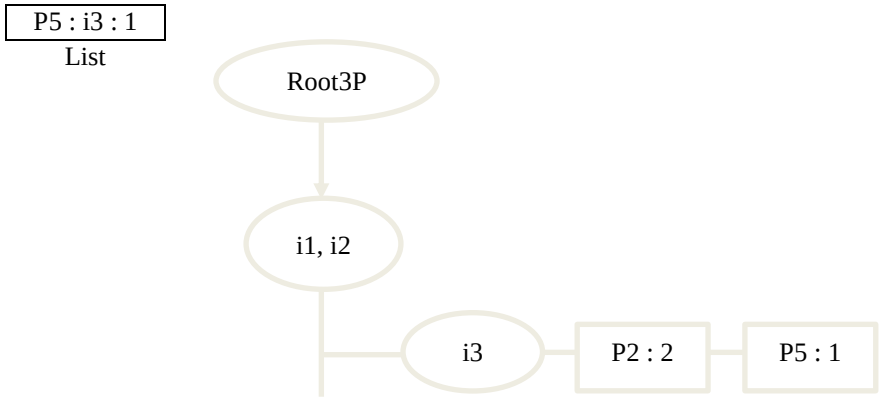


(a)

Figure 39 Construction of 3P-tree



(b)



(c)

Figure 39 continued

Since P2 has no children, no further items are added into the list. Remove the first item. The next child is i3 and it satisfies the minimum support according to step C. 3.1.1. It is seen that there is no children futher, so the parent will also have no children. According to step C..1.1.3, delete the current pk, that is i1,i3. The loop is executed for all the children, so we move to the next parent of 2P-tree, which is i2.

According to step 3.1.1 it is seen that the child i3 satisfies the minimum support, therefore, we add a new parent i2,i3. The position of i3 is P2 and P5, and it is seen that they have no further children. So, the current parent node will also not have any children. According to step C.3.1.1.3, delete the current pk, that is i2,i3. The loop E3 ends as there are no further parent nodes in 2P- tree. The final 3P-tree is shown in Figure 40.

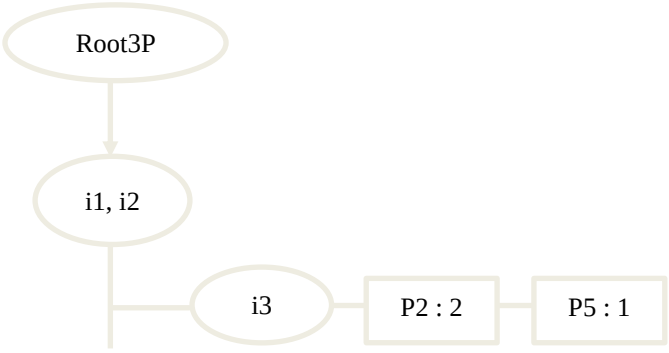


Figure 40 3P-tree

The 4P-tree is formed from the 2P-tree, similar to how the 3P-tree is formed from the 2P-tree. From the Figure 40, it is seen that there can be only one parent in the 4P-tree, they cannot have children as position P2 and P5 in the FP tree has no children. So there are no 4 patterns for this FP-tree and the maximum size that a pattern can have is 3.

4.5 Forming rules from K-Pattern tree

The association rules are produced from the K-patterns, by forming various combinations of the items as antecedent and consequent from the patterns. This section explains how a rule antecedent and consequent is formed.

Algorithm to generate rules:
Aim: Find rules from K-Pattern tree
Input: K-Pattern Tree Collection
Output: Rules
Steps:

1. Initialize
    1.1. K = 2
    1.2. Rule-List-Collection = {}
    1.3. Final-Rule-List={}
2. Repeat until no tree is left in the pattern tree collection
    2.1. Set collection cursor CC to K-pattern tree
    2.2. Repeat until no pattern is left in the K-pattern tree
        2.2.1. Rule-List = {}
        2.2.2. Read the pattern from the tree
        2.2.3. Add each item in the pattern into the look-up table and item array
        2.2.4. Generate base rules from the pattern and add them into the Rule-List using AddRule-I
        2.2.5. Repeat for j= 2,3, .., K-1
            2.2.5.1. tempRuleList = Rule-List
            2.2.5.2. Rule List = {}
            2.2.5.3. Repeat for each Rule in the tempRuleList
                2.2.5.3.1. key = Last item in the antecedent of the Rule
                2.2.5.3.2. index = Dictionary[key]
                2.2.5.3.3. Repeat t= index+1, index+2, .., K-1
                    2.2.5.3.3.1. Add rules to the Rule-List using AddRule-II
            2.2.5.4. If |Rule-List| = 0
                2.2.5.4.1. Break

AddRule-I: Generate Rules and add them into the Rule List

1. Repeat for i = 0,1, .., K-1
   1.1. Initialize a new rule
   1.2. Add Array[i] into Antecedent of new Rule
   1.3. Find Consequent (see details below)
   1.4. Calculate confidence of the rule
   1.5. If confidence of the rule is greater than or equal to min_conf
      1.5.1. Add the rule into Final-Rule-List
2. Add the rule into Rule-List

AddRule-II: Generate Rules and add them into the Rule List

1. Antecedent = Antecedent of the current rule + Array[t]
2. Find Consequent (see details below)
3. Calculate confidence of the rule
4. If confidence of the rule is greater than or equal to min_conf
   4.1. Add the rule into Final-Rule-List
5. Add the rule into Rule-List

Steps to find consequent:

1. Start=0
2. Repeat for each Antecedent Item in antecedent
   2.1. Repeat until *i* is less than the index of the item in Dictionary, where *i* = Start, Start +1, ..
      2.1.1. Add item of index i from array to the consequent
   2.2. Start= Antecedent item index +1
3. Repeat for j is less than K, where j = Start, Start +1, Start +2….
   3.1. Add item of index j from array to the consequent.

4.5.1 Detail Explanation of the Algorithm

The pattern tree collection consists of K-pattern trees. Since the pattern tree starts from 2 patters we have initialized K = 2. The Final-Rule-List is a collection of rules that satisfy the minimum confidence. For each pattern in the pattern trees of the pattern tree collection the rules are generated. Let us consider a 3–pattern = {I1, I2, I3}, where K=3.

Initialize Rule-List = {} and read the pattern from the tree. The items are added into the look up table and array as shown in Figure 41.

I1	I2	I3

Array

Key	Index Value
I1	0
I2	1
I3	2

Look up table

Figure 41 Array and look up table

Step 2.2.4 indicates to generate base rules from the pattern and add them into the Rule-List. The function AddRule-I generate rules from a K-pattern = $\{I_1, I_2, .., I_i, .., I_K\}$. We have index value i = 0,1,2. A new rule is initialized and array[i=0] is added to the antecedent, that is antecedent = I1. Then, find consequent items for the given antecedent by adding items other than the item present in antecedent into the consequent. Initially the start is 0 and i assigns the value from start Dictionary[I1]-1. Since Dictionary[I1]-1 < 0, the loop is not executed. Now the start is set to Dictionary[i1] +1which is equal to 1. The variable j

assigns the value from1 to K-1, that is j=1,2. Therefore we get array[j] = i2, when j=1 and array[j] = i3, when j=2 which are added into the consequent list.

The rule is I1 -> I2, I3. The confidence of the rule is found and if the confidence is greater than the minimum confidence the rule is added into the Rule-List. Similarly the other base rules are generated as I2->I1, I3 when i = 1 and I3->I1, I2 when i =2. Step 2.2.5 indicates for j=2 , tempRuleList = Rule-List and Rule List = {}. For each rule in the tempRuleList, Key is the last item in the antecedent of the rule, I1 in rule I1->I2, I3. The index assigns the value of Dictionary[key] which is 0, and for the variable t from 1 to 2 add rules to Rule-List using AddRule-II. For t=1, the antecedent items are the antecedent of the current rule and Array[t] which is i1 and i2. The consequent will be found as i3. The rule generated is I1,I2->I3, will be added to the final rule list if it satisfies the minimum confidence and rule list. Similarly for t=2, antecedent item are antecedent of the current rule and Array[t] which is i1 and i3. The consequent will be found as i2. The rule generated is I1,I3->I2, will be added to the final rule list if it satisfies the minimum confidence and rule list. The same process is done for the other rules; the final rules generated from this pattern are below.

I1 -> I2, I3
I2 -> I1, I3
I3 -> I1, I2
I1, I2 -> I3
I1, I3 -> I2
I2, I3 -> I1

4.6 Data Structure

In this section, we are going to discuss the data structure of the MASP, FP-tree, K-Patterns and association rules implemented in our application.

The comparison of different data structures [16] available in C# is shown in Figure 42. The Dictionary, SortedList, List, HashSet, Stack and Queue occupy contiguous memory. Since our approach deals with huge data, it is necessary to efficiently manage the memory and contiguous storage leads to out of memory. Among the non-contiguous storage collections we have selected LinkedList since sorting is not required in our approach.

The MASP query is composed of two parts, they are “select attributes” and “where clause”. The “select attributes” is a linked list of attribute names and the “where clause” consists of the MASP items. We have implemented the MASP using the code snippet in Figure 43. Figure 44 represents the data structure for MASP.

The FP-tree is a directed tree whose nodes may have zero or more children. We implemented the node of FP-tree using the link-list as seen in Figure 46 and we have implanted the FP-tree using the code snippet in Figure 45.

Figure 46 illustrates the implementation of this node. As an example, the node N1, N1.1, and N1.1.2 have severity = fatal, primary contribution factor = violation and construction maintenance zone = true, respectively. Notice that, each node of the linked-list initiates another linked-list.

Collection	Ordering	Contiguous Storage?	Direct Access	Lookup Efficiency	Manipulate Efficiency	Notes
Dictionary	Unordered	Yes	Via Key	Key: O(1)	O(1)	Best for high performance lookups.
SortedDictionary	Sorted	No	Via Key	Key: O(log n)	O(log n)	Compromise of Dictionary speed and ordering, uses binary search tree.
SortedList	Sorted	Yes	Via Key	Key: O(log n)	O(n)	Very similar to SortedDictionary, except tree is implemented in an array, so has faster lookup on preloaded data, but slower loads.
List	User has precise control over element ordering	Yes	Via Index	Index: O(1) Value: O(n)	O(n)	Best for smaller lists where direct access required and no sorting.
LinkedList	User has precise control over element ordering	No	No	Value: O(n)	O(1)	Best for lists where inserting/deleting in middle is common and no direct access required.
HashSet	Unordered	Yes	Via Key	Key: O(1)	O(1)	Unique unordered collection, like a Dictionary except key and value are same object.
SortedSet	Sorted	No	Via Key	Key: O(log n)	O(log n)	Unique sorted collection, like SortedDictionary except key and value are same object.
Stack	LIFO	Yes	Only Top	Top: O(1)	O(1)*	Essentially same as List<T> except only process as LIFO
Queue	FIFO	Yes	Only Front	Front: O(1)	O(1)	Essentially same as List<T> except only process as FIFO

Figure 42 Comparison of different data structures [16]

```
public class SelectWhere
    {
        private bool _mine = true;

        private LinkedList<string> sel = new LinkedList<string>();
        private LinkedList<string> wh = new LinkedList<string>();
        public LinkedList<string> Select { get{return sel;} set{sel = value;} }
        public LinkedList<string> Where { get{ return wh;} set{wh = value;} }
        public bool Mine { get { return _mine; } set { _mine = value; } }
    }
```

Figure 43 Code snippet for MASP implementation

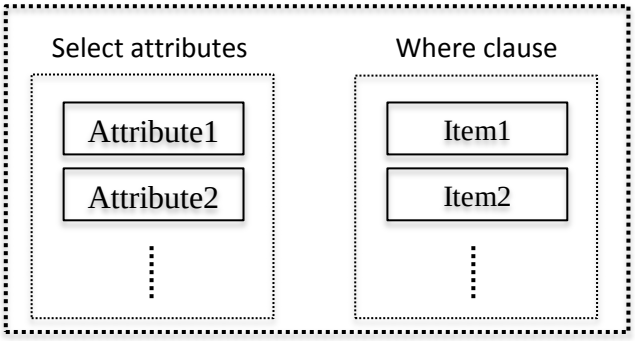


Figure 44 Data structure for MASP

```
public class Node
    {
        public string nodeName { get; set; }
        public int nodeCount { get; set; }
        public IntPtr parent { get; set; }

        LinkedList<Node> Nodes = new LinkedList<Node>();
    }
```

Figure 45 Code snippet for node implementation

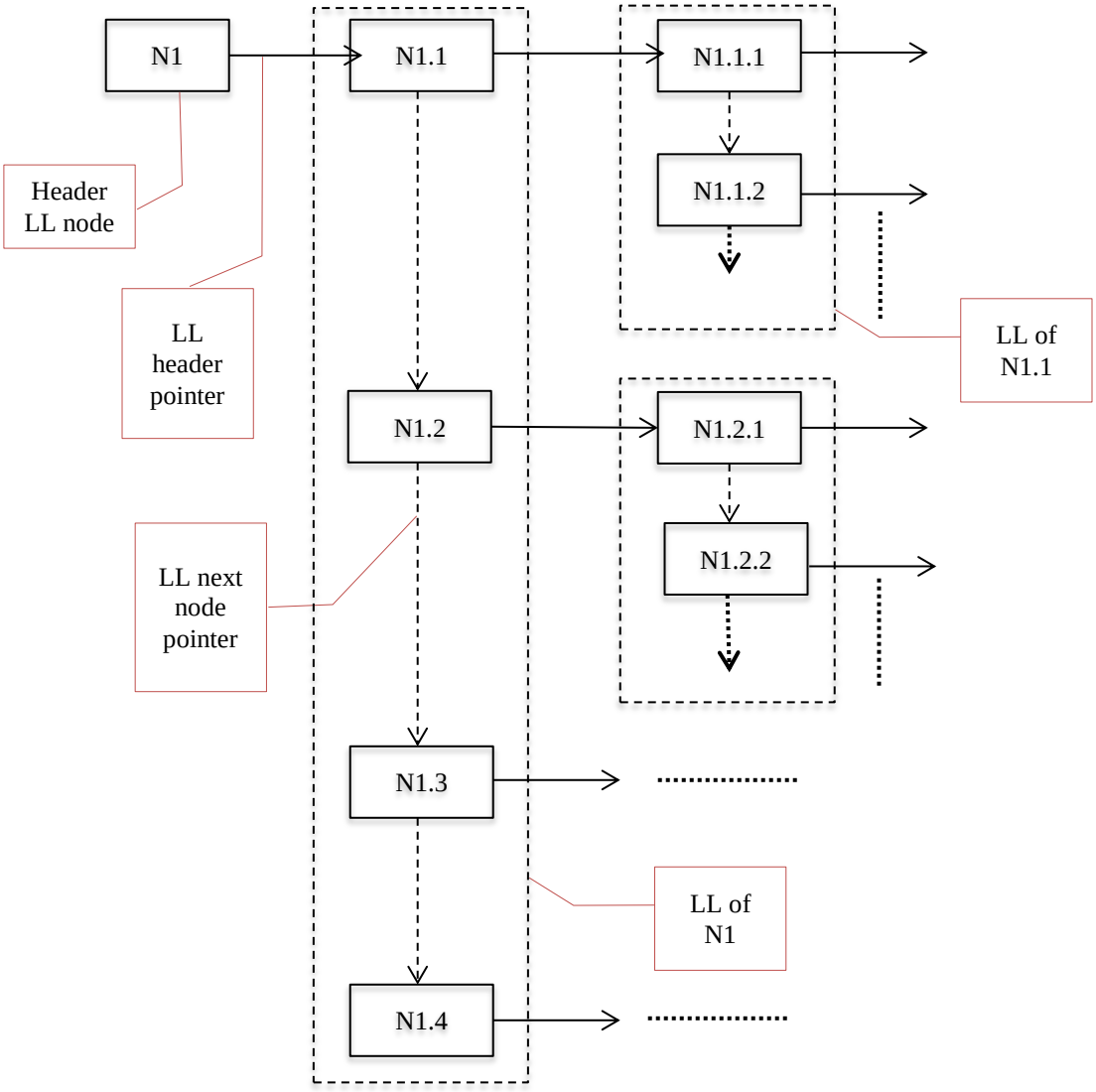


Figure 46 Linked-list implementation of a directed tree (LL: Linked-list)

The K-pattern tree also adapts a directed tree data structure in our implementation. The structure is composed of a root node with children of K-1 patterns (*pattern tree nodes*), which act as the prefix, refer to Figure 48. As an example, let the 3-pattern tree has the sequences of {FC, FA, ..} and the node "FA" has the children of {(A,2), (M, 3, 6), ..}. Then PTN1 = FC, PTN2 = FA, .. where PTN is the pattern tree node and the node CP1.1 will have a tree of the root "A" with a child node AD.1.1 as "2", a tree of the root "M" with two linked-list nodes "3" and "6".We have implemented the K-pattern tree using the code snippet in Figure 47. The CP 1.1 is the child position for the PTN1 with index 1. The AD 1.1.1 holds the addresses and count of the CP 1.1 in the FP-tree.

```
    public class PatternTreeNodes

    {

        public string prefixName { get; set; }

        public LinkedList<ChildPositions>
_childPositions  = new
LinkedList<ChildPositions>();

        public LinkedList<ChildPositions>
ChildPositions

        {

              get { return _childPositions; }

              set { _childPositions = value; }

        }
```

Figure 47 Code snippet for K-pattern tree implementation

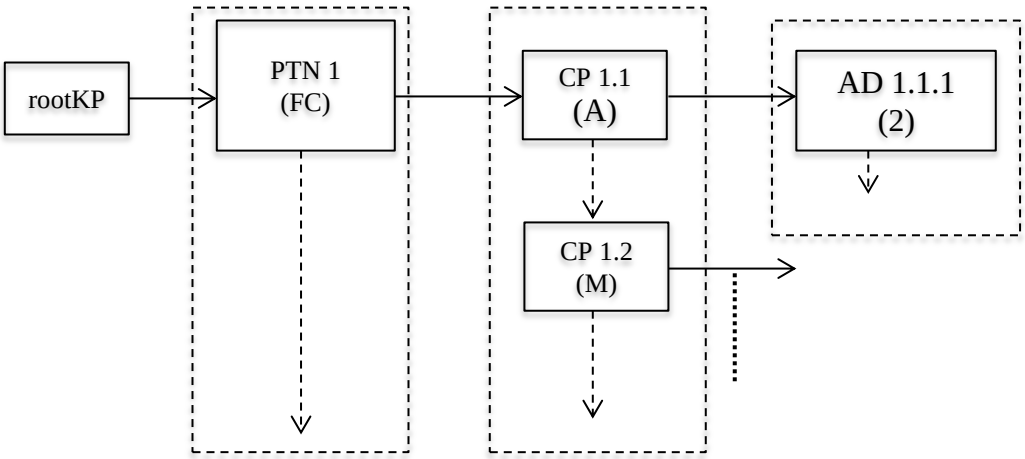


Figure 48 Data structure for K-pattern tree. Values in the parenthesis are given as an example.

The members of the association rule are antecedent, consequent, block ID, support and confidence. The antecedent and consequent are linked list of items. The data structure of an association rule is shown in Figure 49.

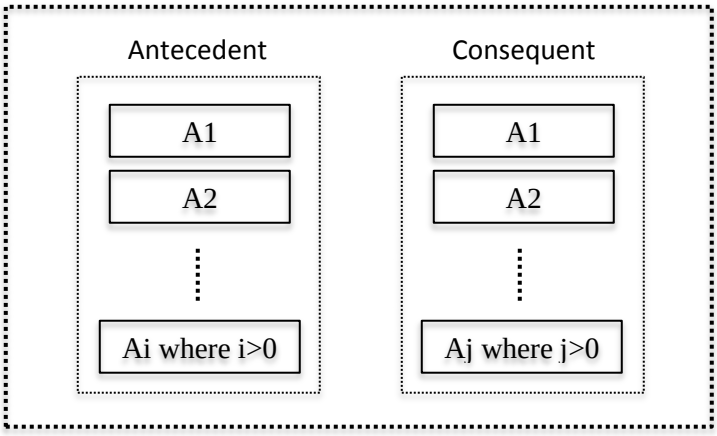


Figure 49 Association rule data structure

4.7 Complexity

The MASP algorithm generates candidate nodes where the number of distinct items I with their frequency is obtained and the I_{max} is found from I = $\{I_1, I_2, I_3, \ldots, I_n\}$ where $n \leq N$, total number of distinct items and the number of rows in a block $|B| \leq |D|$, D is the data table. The complexity in terms of the number of items and the number of candidate nodes is obtained as follows: Let N_D be the number of candidate nodes generated, $|B_A|$ denotes average number of rows in B and M_A denotes average number of attributes for the block B, and $|I_A|$ denotes average number of items in B. I_A is obtained in $O(|B_A| M_A)$ and I_{max} is found in $O(|I_A|)$. The overall complexity for all candidate nodes will be $O(N_D \max(|B_A| M_A, |I_A|)) = O(N_D |B_A| M_A)$.

4.8 Data Export Module

This module allows the user to export the following in various formats

- Bins: The application allows the user to set the bins for the numerical attributes. The discrete data for each attribute can be exported in text or xml format. The text format uses the StringBuilder object and saves in the .txt file. The attributes, bin type and bins are saved in a table format so that they can also be exported in xml format.
- Item dictionary: The item dictionary is dictionary with value as an item and it's key with numbers as unique id exported to an excel sheet.
- MASP: The MASP is exported in three formats. The table structure of MASP is exported in excel and xml format, whereas the query structure of MASP is exported in .txt format.
- Processed Data and Encoded Data: The processed data and encoded data are data tables which that can be exported in .csv, .txt and .xml formats.
- Session: The session is exported in .session format. StringBuilder is used to construct a session. The session exports the input parameters data source, selected attributes and bins. The "Session all" exports session with the item dictionary.
- Association rule: Exporting association rule involves exporting MASP in excel format, antecedent and consequent in table format, rule statistics in table format, item dictionary and item information such as minimum support, confidence and association strength threshold. The advantage of exporting in table format is allowing the user to query the output to obtain his required results.

4.9 User Interface

We have implemented our approach using C# .net. The user interface is shown in Figure 50. The data source is the place where we give the input data. The type of the data is given at database type and the location of the database can be browsed. Session can also be imported. Once the data source is given the attributes are listed in the attributes list box of the select attributes block. User can select the attributes and load the data for the selected attributes only. The numerical data can be processed to discretize it by setting the bins. The minimum support, confidence and alpha values are entered in the parameters block. Once the

data is processed and all the parameters required are entered, the user can mine the MASPs. Export Data helps the user to export the rules generated, MASP'S, bins, processed data, encoded data and session in various formats.

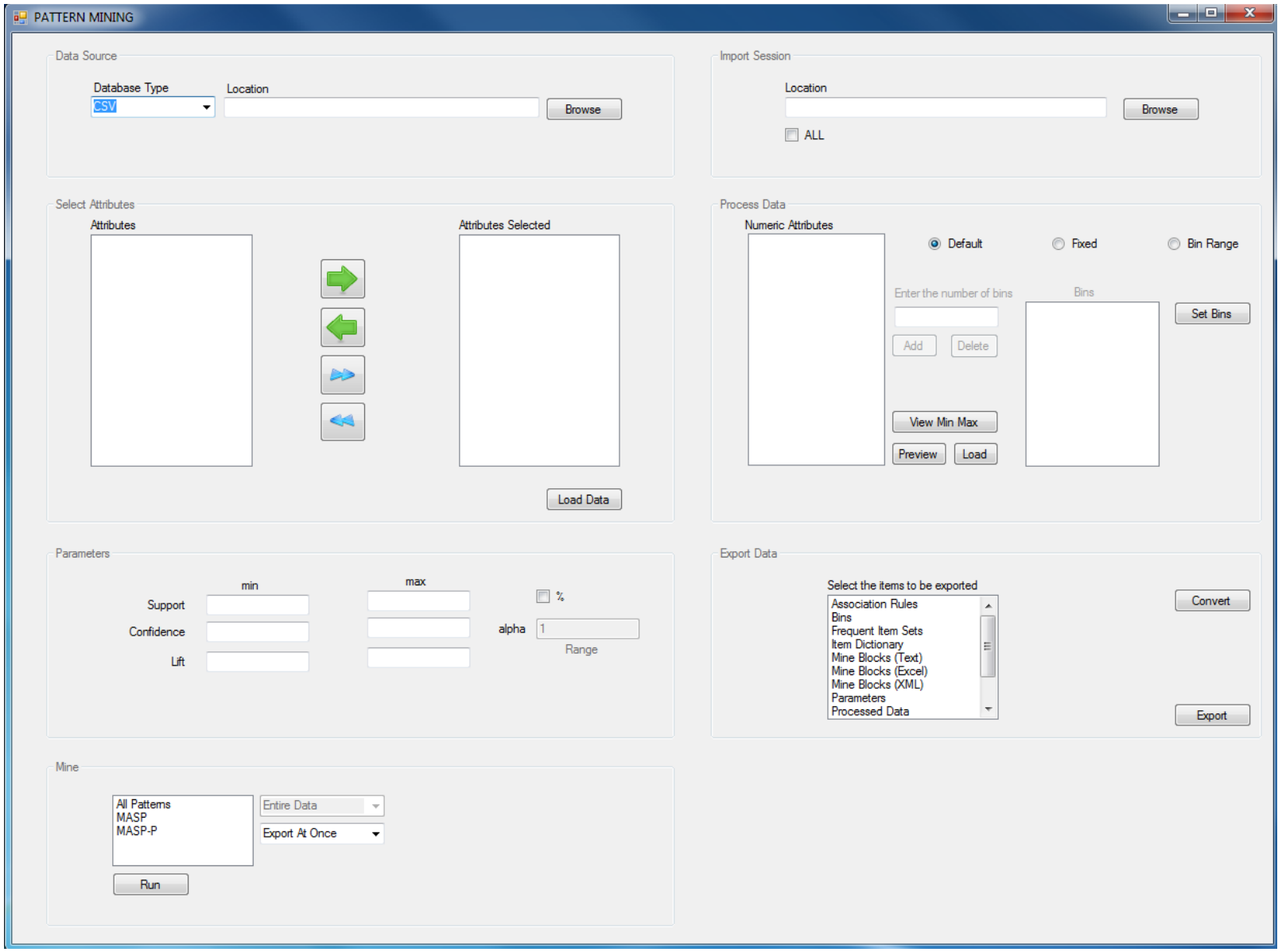


Figure 50 User interface

5 EXPERIMENTS AND RESULTS

5.1 Data

We have experiment our approach using the crash records. We have used and 7997 records with 10 attributes namely Weather, DriverAgeGroupDes, AlcoholPresent, ViolationsCode, Lighting, MannerCollision, PrimaryContributingFactor, RoadwayDeparture, DayOfWeek, TimeofDay. The "TimeofDay" attribute bin type is changed to bin range and its values are discretized to 8 distinct values. After setting the bins the raw data is discretized into the bin data. The bin data is encoded as described before.

5.2 Experimental Setup

We have conducted experiments for the above data with different input parameters. The first experiment is conducted with the minimum support as 10 records, alpha = 1 and minimum confidence as 15%. The second experiment for the same data is conducted with the minimum support as 10 records, alpha = 1 and minimum confidence as 80%.

5.3 Results and Discussion

The total numbers of MASPs detected in experiment 1 are 620, from which most of them were found to be interesting. Some of them are discussed in this section. MASP is also considered as a collection of rules and Figure 52 shows the rules from $MASP_{21}$ in Figure 51. The probability of adults involved in the crash when AlcoholPresent = alcohol no, RoadwayDeparture = 0, PrimaryContributingFactor = violations, Weather = clear, Lighting = daylight, MannerCollision = rear end is 14%, the reliability of the rule is 60% and lift is 1 in $MASP_{21}$, refer Figure 51. The same conditions added with the DriverAgeGroupDes = Adult, Violations = no violations, and TimeofDay= late afternoon, it is seen that the probability of crashes on Friday is 1%, with the confidence as 26% and lift as 1.4 in $MASP_{21}$ and in $MASP_{74}$ the probability of crashes on Friday is 1%, the confidence as 21% and lift as 1.4. From Figure 53, it is seen that the probability of crashes involving youth is 0.1% with confidence 20% and lift 2 when the situation is AlcoholPresent = alcohol no, RoadwayDeparture = 1, MannerCollision = non-collision with motor vehicle, PrimaryContributingFactor = violations, Violations = careless operation, Weather = clear, Lighting = dark - no street lights. $MASP_{388}$ in Figure 54, explains that the probability of crashes due to vehicle condition is 0.21% when AlcoholPresent = alcohol no, RoadwayDeparture = 1, MannerCollision = non-collision with motor vehicle, PrimaryContributingFactor ≠ violations, Weather = clear, Lighting = daylight and DriverAgeGroupDes = Adult. The second experiment detected one MASP with AlcoholPresent = Alcohol No and RoadwayDeparture = 0 with support 78%, confidence 81% and lift 1.

The association rules obtained from the MASPs blocks also seemed to be interesting. In experiment 1, 173764 rules were produced from blocks of the MASPs and some of them are in Figure 55.

Attribute	= or ≠	Value	Lift	Confidence	Support
AlcoholPresent	E	alcohol no			96%
RoadwayDeparture	E	0	1	81%	78%
PrimaryContributingFactor	E	violations	1	80%	62%
Weather	E	clear	1	77%	48%
Lighting	E	daylight	1.1	71%	34%
MannerCollision	E	rear end	1.3	71%	24%
DriverAgeGroupDes	E	adult	1	60%	14%
Violations	E	no violations	1.3	59%	8%
TimeofDay	E	late afternoon	1.4	35%	3%
DayOfWeek	E	friday	1.4	26%	1%

Figure 51 $MASP_{21}$

AlcoholPresent = alcohol no -> RoadwayDeparture = 0 Support = 78%, confidence = 81%, lift = 1

AlcoholPresent = alcohol no, RoadwayDeparture = 0 → PrimaryContributingFactor = violations Support = 62%, confidence = 80%, lift = 1

AlcoholPresent = alcohol no, RoadwayDeparture = 0, PrimaryContributingFactor = violations → Weather = clear Support = 48% confidence = 77%, lift = 1

AlcoholPresent = alcohol no, RoadwayDeparture = 0, PrimaryContributingFactor = violations, Weather = clear → Lighting = daylight Support = 34%, confidence = 71 %, lift = 1.1

AlcoholPresent = alcohol no, RoadwayDeparture = 0, PrimaryContributingFactor = violations, Weather = clear, Lighting = daylight → MannerCollision = rear end Support = 24%, confidence = 71%, lift = 1.3

AlcoholPresent = alcohol no, RoadwayDeparture = 0, PrimaryContributingFactor = violations, Weather = clear, Lighting = daylight, MannerCollision = rear end → DriverAgeGroupDes = adult Support = 14%, confidence = 60%, lift = 1

AlcoholPresent = alcohol no, RoadwayDeparture = 0, PrimaryContributingFactor = violations, Weather = clear, Lighting = daylight, MannerCollision = rear end, DriverAgeGroupDes = adult → Violations = no violations Support = 8%, confidence = 59%, lift = 1.3

AlcoholPresent = alcohol no, RoadwayDeparture = 0, PrimaryContributingFactor = violations, Weather = clear, Lighting = daylight, MannerCollision = rear end, DriverAgeGroupDes = adult, Violations = no violations → TimeofDay = late afternoon Support = 3%, confidence = 35%, lift = 1.4

AlcoholPresent = alcohol no, RoadwayDeparture = 0, PrimaryContributingFactor = violations, Weather = clear, Lighting = daylight, MannerCollision = rear end, DriverAgeGroupDes = adult, Violations = no violations, TimeofDay = late afternoon → DayOfWeek = Friday Support = 1%, confidence = 26%, lift = 1.4

Figure 52 Rules from $MASP_{21}$

Attribute	= or ≠	Value	Lift	Confidence	Support
AlcoholPresent	E	alcohol no			96%
RoadwayDeparture	E	1	0.9	19%	18%
MannerCollision	E	non-collision with motor vehicle	4.1	67%	12%
PrimaryContributingFactor	E	violations	0.9	67%	8%
Violations	E	careless operation	3.2	71%	6%
Weather	E	clear	0.9	64%	4%
Lighting	E	dark - no street lights	2.5	32%	1%
DriverAgeGroupDes	E	youth	2	20%	0.19

Figure 53 $MASP_{515}$

Attribute	= or ≠	Value	Lift	Confidence	Support
AlcoholPresent	E	alcohol no	1	96%	96%
RoadwayDeparture	E	0	1	81%	78%
MannerCollision	E	non-collision with motor vehicle	1	80%	62%
PrimaryContributingFactor	NE	violations	1	77%	48%
Weather	E	clear	1.1	71%	34%
Lighting		daylight	1.3	71%	24%
DriverAgeGroupDes	E	adult	1	60%	14%
PrimaryContributingFactor	E	vehicle conditions	12	27%	0.21%

Figure 54 $MASP_{388}$

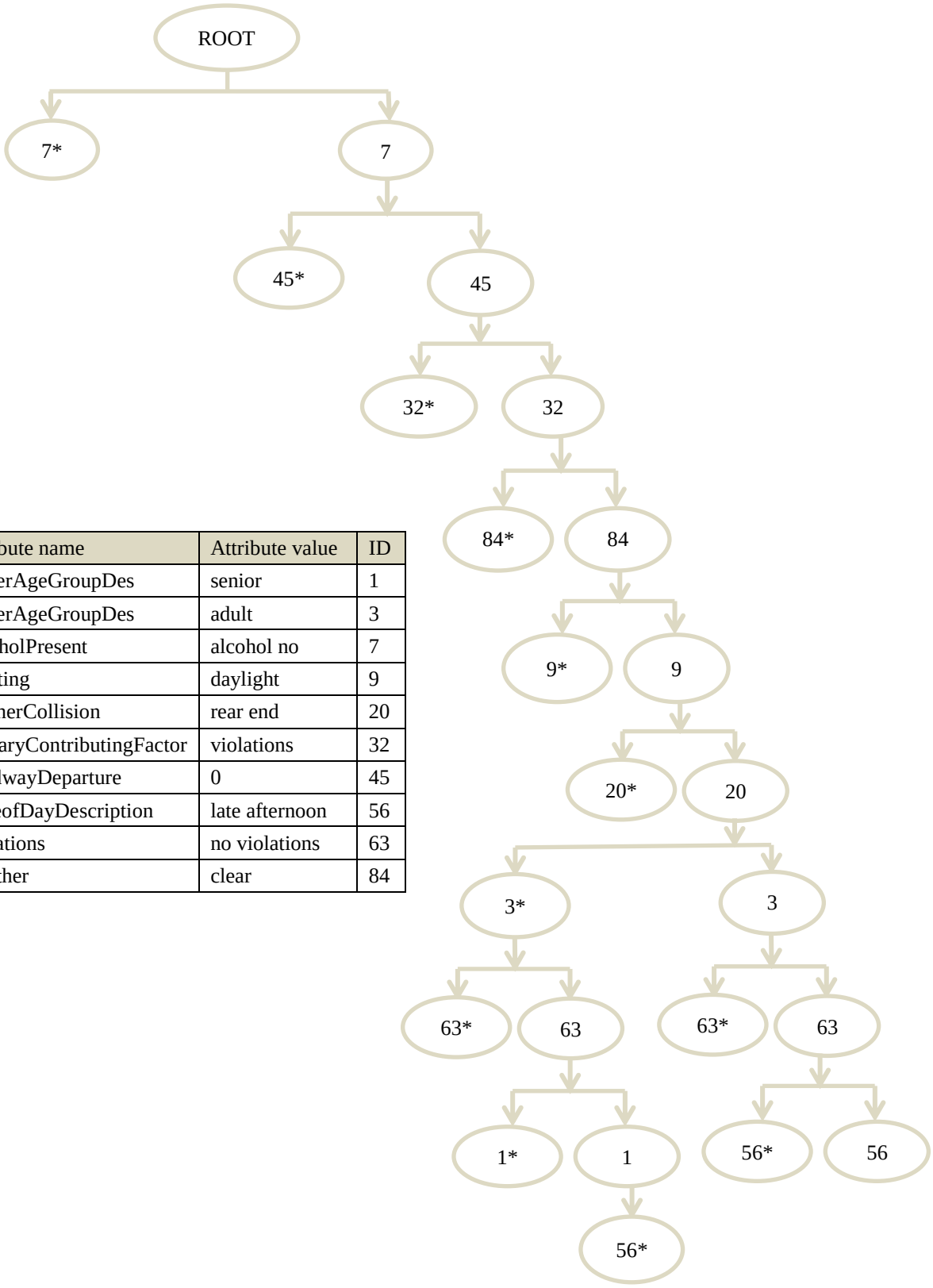
(AlcoholPresent = alcohol no, RoadwayDeparture = 0, PrimaryContributingFactor = violations, Weather = clear, Lighting = daylight, MannerCollision = rear end, DriverAgeGroupDes ≠ adult, Violations = careless operation), TimeofDay = late morning → DriverAgeGroupDes = young adult Support = 0.1%, confidence = 50%, lift= 5.1

(AlcoholPresent = alcohol no, RoadwayDeparture = 0, PrimaryContributingFactor = violations, Weather = clear, Lighting = daylight, MannerCollision = rear end, DriverAgeGroupDes ≠ adult, Violations = careless operation), DriverAgeGroupDes = young adult → TimeofDay = late morning Support = 0.1%, confidence = 80%, lift= 8.99

(AlcoholPresent = alcohol no, RoadwayDeparture = 1, PrimaryContributingFactor ≠ violations, Weather ≠ clear, Lighting ≠ daylight, Lightning ≠ dark-continuous street light), Weather = cloudy → DriverAgeGroupDes = adult, MannerCollision = rear end Support = 0.12%, confidence = 40%, lift= 1.53

Figure 55 Rules from MASPs blocks, in the form (MASP), antecedent → consequent

The MASPs detected for the same data with the support 0.3, confidence 30% and alpha 1 is given in Figure 56. Experiment 1 and experiment 2 were conducted using Weka with the support and confidence values generated 6885 rules with maximum number of items involved in a rule = 5 and 827 rules maximum number of items involved in a rule = 7 using Apriori algorithm, respectively. Most of the rules generated using weka was obvious and the FP-growth algorithm was out of memory for the data used by us, but our approach handled the same data and detected interesting MASPs and association rules from the blocks of MASPs. In our approach, the maximum number of items involved in a rule in experiment 2 is 10. The same experiments were conducted using R software, but it was noticed that R did not have user friendly interface. In weka and R, the results are exported in text format, but our application is user friendly and allows the user to export the results in text and table format. The intermediate results like the encoded data, bin data, bins set for each attribute, etc. can be exported in various formats. The table format allows the user to further query the results to get a deeper insight into the results obtained.



Attribute name	Attribute value	ID
DriverAgeGroupDes	senior	1
DriverAgeGroupDes	adult	3
AlcoholPresent	alcohol no	7
Lighting	daylight	9
MannerCollision	rear end	20
PrimaryContributingFactor	violations	32
RoadwayDeparture	0	45
TimeofDayDescription	late afternoon	56
Violations	no violations	63
Weather	clear	84

Figure 56 MASPs detected for support 0.3%, confidence 30% and alpha =1

6 CONCLUSION AND FUTURE WORK

In this thesis, we aim to introduce a new association rule detection method. The proposed approach searches for the most associated sequential patterns and generate rules from these patterns. Further, regular association rules can be generated from the sub-data space of the original data. The proposed approach can be very promising for analysis of *Big Data as well* since the MASP algorithm can detect very long interesting patterns efficiently. In the implementation of the data structures for association rule mining algorithms LinkedList class from the C# library is utilized for efficient memory allocation instead of array-like data structures which requires continuous memory blocks. In the future, parallel programing, multi-threading, using array-likes and LinkedList for storage and B-tree based search can be implemented to improve the speed and memory management. Since there is a large volume of data, it would be more useful to increase the efficiency in terms of speed and memory.

REFERENCES

[1] P. JIAN, Y. YIN, R. MAO and M. Heikki, "Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach*," Data Mining and Knowledge Discovery, vol. 8, no. 1, pp. 53-87, 2004.

[2] IBM, [Online]. Available: http://searchbusinessanalytics.techtarget.com/definition/association-rules-in-data-mining.

[3] P.-N. Tan, M. Steinbach and V. Kumar, "Association Analysis: Basic Concepts and Algorithms," in Introduction to Data Mining, 2006.

[4] IBM, [Online]. Available: http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.im.model.doc/c_lift_in_an_association_rule.html.

[5] IBM, [Online]. Available: http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.im.model.doc/c_lift_in_an_association_rule.html.

[6] K. Geurts, G. Wets, T. Brijs and K. Vanhoof, "Profiling High Frequency Accident Locations Using Association Rules," Transportation Research Record: Journal of the Transportation Research Board, vol. 1840, pp. 123-130, 2003.

[7] P.-N. Tan, V. Kumar and J. Srivastava, "Selecting the right interestingness measure for association patterns," in KDD '02 Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, New York, 2002.

[8] A. Freitas, "On rule interestingness measures," Knowledge-Based Systems, vol. 12, no. 5-6, pp. 309-315, 1999.

[9] L. Geng and H. J. Hamilton, "Interestingness measures for data mining: A survey," vol. 38, 2006.

[10] R. Marukatat, "Structure-Based Rule Selection Framework for Association Rule Mining of Traffic Accident Data," in Computational Intelligence and Security, Springer Berlin Heidelberg, 2006, pp. 231-239.

[11] A. Montella, "Identifying crash contributory factors at urban roundabouts and using association rules to explore their relationships to different crash types," Accident Analysis and Prevention, vol. 43, p. 1451–1463, 2011.

[12] A. Pande and A.-A. Mohamed, "Market basket analysis of crash data from large jurisdictions and its potential as a decision support tool," Safety Science, vol. 47, no. 1, pp. 145-154, 2009.

[13] H. Song-bai , Y.-j. Wang, Y.-k. Sun, W.-W. Gao, Q. Chen and Y.-Q. An , "The Research of Multidimensional Association Rule in Traffic Accidents," in Wireless Communications, Networking and Mobile Computing, 2008. WiCOM '08. 4th International Conference, 2008.

[14] T. Wu, Y. Chen and J. Han, "Association Mining in Large Databases: A Re-examination of Its Measures," in Knowledge Discovery in Databases: PKDD 2007, Springer, 2007, pp. 621-628.

[15] F. Guillet and H. J. Hamilton, "Choosing the Right Lens: Finding What is Interesting in Data Mining," in Quality Measures in Data Mining, Springer, 2007, pp. 3-24.

[16] [Online]. Available: http://geekswithblogs.net/BlackRabbitCoder/archive/2011/06/16/c.net-fundamentals-choosing-the-right-collection-class.aspx.

[17] J. Blanchard, F. Guillet, R. Gras and H. Briand, "CiteSeerX," 2005. [Online]. Available: http://citeseer.uark.edu:8080/citeseerx/viewdoc/similar?doi=10.1.1.60.6689&type=ab.

[18] R. Agrawal and S. Ramakrishnan, "Mining sequential patterns: Generalizations and performance improvements," in Advances in Database Technology, 1996, pp. 1-17.

VITA

Donepudi, Harisha was born in Vishakhapatnam, India to Mrs. Rajeswari and Mr. Srinivas Chakravarthy. After graduating from her high school with distinction, she went to study B.E in Electronics and communications. She is currently a Masters student in the Department of Electrical and Computer Engineering at Louisiana State University, where she has been a graduate student since Spring 2011.