

**AN N-LIST BASED ALGORITHM FOR MINING FREQUENT  
CLOSED PATTERN**

*Submitted by*

**M.PABEETHRA (13C62)**

**S.SANTHIYA (13C89)**

*in partial fulfillment for the award of the degree*

*of*

**BACHELOR OF ENGINEERING**

*in*

**COMPUTER SCIENCE AND ENGINEERING**

**THIAGARAJAR COLLEGE OF ENGINEERING,  
MADURAI – 15**

(A Government Aided ISO 9001: 2008 Certified Autonomous Institution Affiliated to Anna University)



**ANNA UNIVERSITY: CHENNAI 600 025**

*November 2016*

# THIAGARAJAR COLLEGE OF ENGINEERING MADURAI-15

(A Government Aided ISO 9001: 2008 Certified Autonomous Institution Affiliated to Anna University)



## BONAFIDE CERTIFICATE

Certified that this project report “**An N-List based Algorithm For Mining Frequent Closed Pattern**” is the bonafide work of “**M.PABEETHRA (13C62), S.SANTHIYA (13C89)**” who carried out the project work under my supervision during the Academic Year 2016-2017.

### SIGNATURE

Dr.S.Mercy Shalinie, M.E., Ph.D.,

### HEAD OF THE DEPARTMENT

COMPUTER SCIENCE AND ENGINEERING

THIAGARAJAR COLLEGE OF ENGG.

MADURAI – 625 015

### SIGNATURE

Mrs.B.Subbulakshmi, M.E,

### SUPERVISOR & ASSISTANT PROFESSOR

COMPUTER SCIENCE AND ENGINEERING

THIAGARAJAR COLLEGE OF ENGG.

MADURAI – 625 015

Submitted for the VIVA VOCE Examination held at Thiagarajar College of

Engineering on .....

**INTERNAL EXAMINER**

**EXTERNAL EXAMINER**

## **ACKNOWLEDGEMENT**

## ACKNOWLEDGEMENT

We wish to express our deep sense of gratitude to **Dr. V. Abhai kumar**, Principal of Thiagarajar College of Engineering for his support and encouragement throughout this project work.

We wish to express our sincere thanks to **Dr. S. Mercy Shalinie**, Head of the Department of Computer Science and Engineering for her support and ardent guidance.

We owe our special thanks and gratitude to our guide **Mrs. B. Subbulakshmi**, Assistant Professor, Department of Computer Science and Engineering for her guidance and support throughout our project.

We are also indebted to all the teaching and non-teaching staff members of our college for helping us directly and indirectly by all means throughout the course of our study and project work.

We extremely thank our parents, family members and friends for their moral support and encouragement for our project.

## **ABSTRACT**

## **ABSTRACT**

The mining of frequent itemset is the most significant issues addressed in data mining research which aims to identify the behaviors exhibited by most of the customers for transaction database. Frequent closed patterns (FCPs), a condensed representation of frequent patterns, have been proposed for the mining of (minimal) non-redundant association rules to improve performance in terms of memory usage and mining time. Recently, the N-list structure has been proven to be very efficient for mining frequent patterns. This study proposes an N-list-based algorithm for mining FCPs called NAFCP. Two theorems for fast determining FCPs based on the N-list structure are proposed. The N-list structure provides a much more compact representation compared to previously proposed vertical structures, reducing the memory usage and mining time required for mining FCPs. The experimental results show that NAFCP outperforms previous algorithms in terms of runtime and memory usage in most cases.

## TABLE OF CONTENTS

CHAPTER NO	TITLE	PAGENO
1	<b>Introduction</b>	8
	1.1 Data Mining	
	1.2 Association Rule Mining	
	1.3 Frequent Pattern	
	1.4 Frequent Closed pattern	
2	<b>Problem Statement</b>	11
3	<b>System Requirements</b>	13
	3.1 Hardware Requirements	
	3.2 Software Requirements	
4	<b>Literature Review</b>	15
	4.1 Limitations of Existing System	
	4.2 Proposed System	
	4.3 Need for new System	
	4.4 Benefits of New System	
5	<b>Design Methodology</b>	20
	5.1 Block Diagram	
	5.2 Modules	
6	<b>Implementation</b>	23
	6.1 N-List data structure	
	6.2 Examples	
	6.3 PPC tree	

	6.4 Frequent 1-pattern and their N-Lists	
	6.5 FCP's identified using NAFCP	
	6.6 Algorithm for finding frequent closed pattern	
7	<b>Experimental Evaluation</b>	33
	7.1 Characteristics of experimental datasets	
	7.2 Execution Time	
8	<b>Snapshots</b>	39
10	<b>Conclusion and Future Work</b>	43
11	<b>References</b>	45



## **LIST OF FIGURES**

<b>FIGURE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
<b>1</b>	<b>Run Time difference between Accident dataset</b>	<b>35</b>
<b>2</b>	<b>Run Time difference between Chess dataset</b>	<b>35</b>
<b>3</b>	<b>Run Time difference between Connect dataset</b>	<b>36</b>
<b>4</b>	<b>Run Time difference between Mushroom dataset</b>	<b>36</b>
<b>5</b>	<b>Run Time difference between Accident s incremental dataset</b>	<b>37</b>
<b>6</b>	<b>Run Time difference between Chess incremental dataset</b>	<b>38</b>
<b>7</b>	<b>Run Time difference between Connect incremental dataset</b>	<b>39</b>
<b>8</b>	<b>Run Time difference between Mushroom incremental dataset</b>	<b>39</b>

## **LIST OF TABLES**

<b>TABLE NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
<b>1</b>	<b>Sample database</b>	<b>28</b>
<b>2</b>	<b>Dataset details</b>	<b>36</b>

## **INTRODUCTION**

# INTRODUCTION

## 1.1 DATA MINING:

Data mining is the process of analyzing data from different perspectives and summarizing it into useful information. The information can then be used to increase revenue, cuts costs, or both. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Technically, data mining is the process of finding correlations or patterns among dozens of fields in large relational databases.

## 1.2 ASSOCIATION RULE MINING:

In association rule mining, the rule contain two parts, an antecedent (if) and a consequent (then). An antecedent is an item found in the data. A consequent is an item that is found in combination with the antecedent. For Example,

**Bread-->Jam[sup=0.5%, conf=70%]**

The rule support must be greater than minimum support otherwise the rule is not considered. The example rule says that if the customer buy a bread then she/ he may also purchase Jam.

## 1.2 FREQUENT PATTERN:

Frequent pattern is a group of items shared by no less than minsup transactions in the input database. It is the process of finding regularities in data. It is a pattern that occurs frequently in a database. First proposed in the context of frequent itemsets and association rule mining for market basket analysis.

### **1.3 FREQUENT CLOSED PATTERN:**

A Frequent Pattern is a pattern that appears in atleast 'minsup' in a database, where 'minsup' is a parameter set up the user. A Frequent Closed Pattern is a frequent pattern such that it is not included in another pattern having exactly the same support.

In many cases, a large number of frequent patterns are identified, making them difficult to interpret and mine rules from. One solution is to use a condensed representation that reduces the overall size of the collection of patterns and rules.

A collection of FCPs can be used to deduce all frequent patterns. The small number of FCPs compared to that of all frequent patterns greatly reduces memory and computation time requirements.

The N-list structure helps to reduce the mining time and memory usage because it is much more compact than previously proposed vertical structures. Additionally, the support of a candidate FCPs called frequent pattern can be determined through N-list intersection operations.

## **PROBLEM STATEMENT**

## **PROBLEM STATEMENT:**

Due to the usefulness of frequent closed pattern mining there is a problem of time consuming, while the construction of frequent item sets and when update new transaction information to the old transaction. So we propose an algorithm for mining frequent closed pattern using N-List data structure.

## **SYSTEM REQUIREMENT**

## **SYSTEM REQUIREMENTS**

### **3.1 HARDWARE REQUIREMENTS**

The minimum hardware configuration of the system on which the project was developed is as follows:

- Processor : Pentium IV 2.8 Ghz
  - CPU clock : 450 Mhz
  - RAM : 512 Mb
  - HardDisk : 40 GB
  - Keyboard : Any type
  - Mouse : Any type
  - Monitor : Any type/Color Monitor preferable
- 
- **SOFTWARE REQUIREMENTS**
    - Operating System : Windows 7
    - Language Used : Java(jdk1.7.0)



## **LITERATURE REVIEW**

## LITERATURE REVIEW

*Charm/dCharm(2002)* was proposed by *Zaki and Hsiao* which enumerates closed sets using a dual itemset tidset search tree which uses an efficient hybrid search that skips many level. It also uses a technique called diffset to reduce the memory footprint of immediate computations. It uses a fast hash approach to remove any “non-closed” sets. It has some disadvantage that the more time required when the minimum support is smaller.

So we have *lcm algorithm(2004)* was proposed by *Uno Et Al* to overcome this the construction of tree shaped traversal routes composed of only frequent closed itemsets which is induced by parent child relationships. By traversing the route in a depth-first manner it finds closed itemsets in polynomial time per itemset without storing the previous closed itemset in memory. It uses sparse and dense input data. The drawback is that sometimes these algorithms runs out of memory before completion.

Now we have *frequent pattern(FP) close algorithm(2005)* was proposed by *Grahne and Zhu* which discover frequent closed patterns without candidate itemset generation. It builds a compact data structure called FP tree and find frequent itemset directly from the tree. But it is costly when the mining long patterns or with low minimum support thresholds.

In order to overcome this *closet algorithm(2007)* was proposed by *Ceglar and Roddick* came into account that the algorithm s used to mine the help of tree techniques: compression frequent pattern tree structure without candidate generation, single path compression and partition based projection., but the issue is that it is less efficient for smaller itemsets.

Now we have *closet+algorithm(2013)* was proposed by *Vo,Hong and Le* which scans the database only once to find the global frequent pattern and sort the database in support descending order and forms the frequent pattern list. then builds the frequent pattern tree using the pattern list and using depth first searching it finds the closed frequent patterns. The disadvantage is that the memory consumption is high when threshold is gradually lowered.

#### **4.1 LIMITATIONS IN THE EXISTING SYSTEM**

Mining Closed Frequent pattern consumes more time because it mines a complete rule set and when the minimum support is smaller.

For large database it needs more rescan over the original transactions.

#### **4.2 NEED FOR NEW SYSTEM**

To reduce the mining time and memory usage for the construction of frequent closed pattern.

#### **4.3 PROPOSED SYSTEM**

1. From the transactional database, remove all the items whose frequency does not satisfy the minimum threshold.
2. Then sort the remaining items in descending order of frequency.
3. Insert the remaining items in each transactions into the PPC tree with the required *pre* and *post* values associated with each node.
4. Create N-list associated with frequent 1-pattern.
5. Combines each element in the input with the remaining element.
6. Check whether these nodes satisfy theorem 1 and theorem 2.
7. If so, combine the element and continues to next element.
8. If not, the next element in the input is combined and the theorems are verified recursively.

9. Thus frequent closed item set can be obtained after the completion of combinations in transactional database.

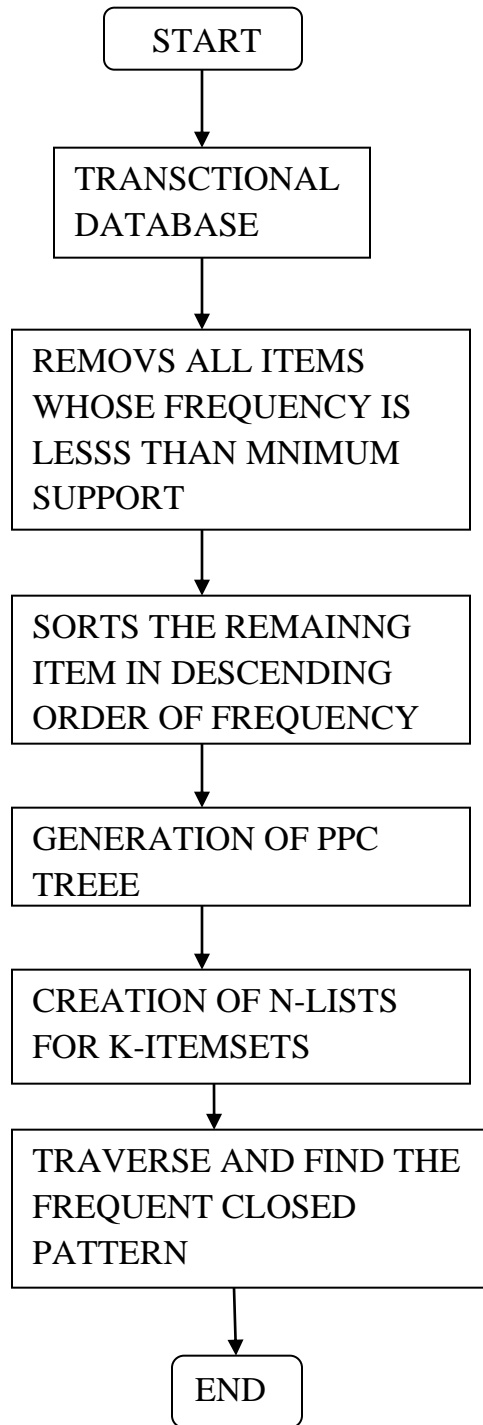
#### **4.4 BENEFITS OF NEW SYSTEM**

- It reduces the mining time for the construction of frequent closed itemsets using N-List data structure.
- For an incremental database it reduces the number of rescanning over the old database by using PPC tree.

## **DESIGN METHODOLOGY**

## 5. DESIGN METHODOLOGY:

### 5.1 BLOCK DIAGRAM



## 5.2 MODULES

### Frequent Closed Patterns:

Consider a transaction dataset DB that comprises  $n$  transactions such that each transaction contains a number of items. An example DB featuring  $n = 6$  transactions. This DB is used for illustrative purposes throughout the remainder of this paper. The support of a pattern  $X$ , denoted by  $\sigma(X)$ , where  $X \in I$  and  $I$  is the set of all items in DB, is the number of transactions containing all the items in  $X$ . A pattern with  $k$  items is called a  $k$ -pattern and  $I_1$  is the set of frequent 1-patterns sorted in order of descending frequency. For convenience, pattern  $\{A, C, W\}$  is written as ACW. Let minSup (minimum support threshold) be a given threshold. A pattern  $X$  is called a frequent pattern if  $\sigma(X) \geq \text{minSup} * n$ . A frequent pattern is called an FCP if none of its supersets has the same support. Let minSup = 50%. AW and ACW are two frequent patterns because  $\sigma(AW) = \sigma(ACW) = 4 > 50\% * 6$ . However, AW is not an FCP because ACW is its superset and has the same support.

**PPC tree:** A FP-tree-like structure called the PPC-tree is implemented. Each node in the tree holds five values, namely  $n(N_i)$ ,  $f(N_i)$ ,  $\text{childs}(N_i)$ ,  $\text{pre}(N_i)$  and  $\text{post}(N_i)$ , which are the frequent 1-patterns in  $I_1$ , the frequency of this node, the set of child nodes associated with this node, the order number of the node when traversing the tree in pre-order form, and the order number when traversing the tree in post-order form, respectively.

**N-list structure:** N-list associated with 1-patterns: The PP-code of each node  $N_i$  in a PPC-tree comprises a tuple of the form  $C_i = (\text{pre}(N_i), \text{pre}(N_i), f(N_i))$ . The N-list associated with a pattern  $A$ , denoted by  $NL(A)$ , is the set of PP-codes associated with nodes in the PPC-tree that equal  $A$ .

$$NL(A) = \bigcup_{\{N_i \in n(N_i)=A\}} C_i$$

where  $C_i$  is the PP-code associated with  $N_i$ . The support for  $A$ ,  $r(A)$ , is calculated by:

$$\sigma(A) = \sum_{C_i \in NL(A)} f(C_i)$$

N-list associated with k-patterns: Let  $XA$  and  $XB$  be two  $(k-1)$ -patterns with the same prefix  $X$  ( $X$  can be an empty set) such that  $A$  is before  $B$  according to the I1 ordering.  $NL(XA)$  and  $NL(XB)$  are two N-lists associated with  $XA$  and  $XB$ , respectively. For each PP-code  $C_i \in NL(XA)$  and  $C_j \in NL(XB)$ , if  $C_i$  is an ancestor of  $C_j$  and:

1. If  $\exists C_z \in NL(XAB)$  such that  $pre(C_z) = pre(C_i)$  and  $post(C_z) = post(C_i)$ , then the algorithm updates the frequency count of  $C_z$ ,  
 $f(C_z) = f(C_z) + f(C_i)$ .

2. Otherwise, the algorithm adds  $(pre(C_i), post(C_i), f(C_j))$  to  $NL(XAB)$ .

The support of  $XAB$ , denoted by  $r(XAB)$ , is calculated as follows:

$$\sigma(ABC) = \sum_{C_i \in NL(XAB)} f(C_i)$$



## **IMPLEMENTATION**

### 6.1.N-list-based algorithm for mining FCPs

Prior to introducing the proposed algorithm, a number of basic concepts are proposed. Let  $XA$  and  $XB$  be two frequent patterns ( $XA$  is before  $XB$  in frequent 1-pattern order and  $X$  can be an empty set), and  $NL(XA)$  and  $NL(B)$  be the N-lists associated with  $XA$  and  $XB$ , respectively.

**Definition 1:**

(N-list subset operation):  $NL(XB) \subseteq NL(XA)$  if and only if  $\forall C_i \subseteq NL(XB), \forall C_j \subseteq NL(XA)$  such that  $C_j$  is an ancestor of  $C_i$ .

**Theorem 1:**

If  $NL(XB) \subseteq NL(XA)$ , then  $XB$  is not a closed pattern.

**Proof:**

According to the method for determining the N-list associated with a k-pattern and  $NL(XB) \subseteq NL(XA)$ , we have:

$$NL(XAB) = \bigcup (pre(C_i), post(C_i), f(C_j))$$

where  $C_i \subseteq NL(XA)$ ,  $C_j \subseteq NL(XB)$ , and  $C_i$  is an ancestor of  $C_j$ .

Therefore,

$$\sigma(XAB) = \sum_{C_j \in NL(XB)} f(C_j) = \sigma(XB)$$

Based on the FCP definition,  $XB$  is not a closed pattern. [Theorem 1](#) is proven.

**Theorem 2.:**

If  $NL(XB) \subseteq NL(XA)$  and  $\sigma(XB) = \sigma(XA)$ , then  $XA$  and  $XB$  are not closed patterns.

**Proof:**

According to Theorem 1, we have  $\sigma(XAB) = \sigma(XB)$ . We also have  $\sigma(XB) = \sigma(XA)$  according to hypothesis  $\sigma(XAB) = \sigma(XB) = \sigma(XA)$ . Based on the FCP definition,  $XA$  and  $XB$  are not FCPs. Therefore, Theorem 2 is proven. We utilize

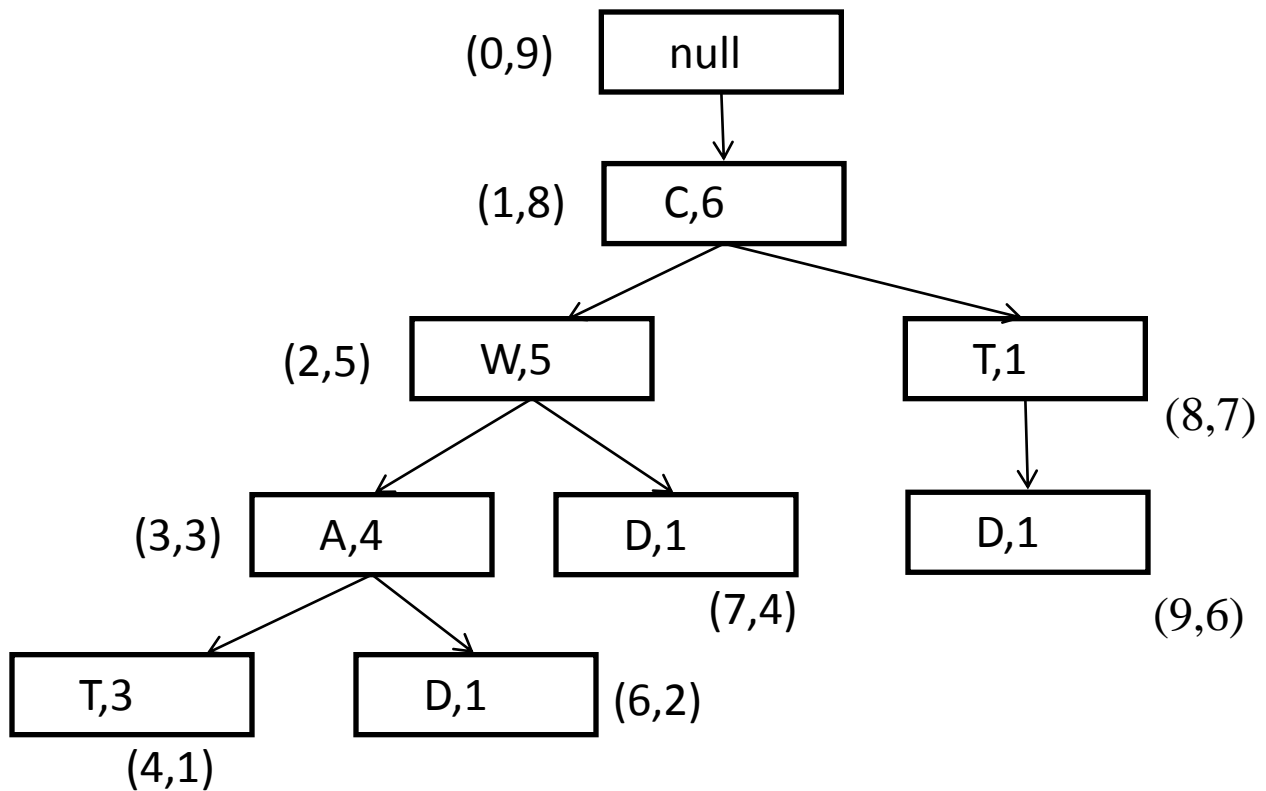
Theorems 1 and 2 in NAFCP, before the algorithm combines two patterns XB and XA, if  $NL(XB) \subseteq NL(XA)$ , then two cases are considered:

1. If  $\sigma(XB) = \sigma(XA)$ , then the algorithm updates node XB to XAB and removes XA because XB and XA are not closed patterns according to Theorem 2.
2. Otherwise, the algorithm updates node XB to XBA because only XB is not a closed pattern according to Theorem 1.

## 6.2.Examples:

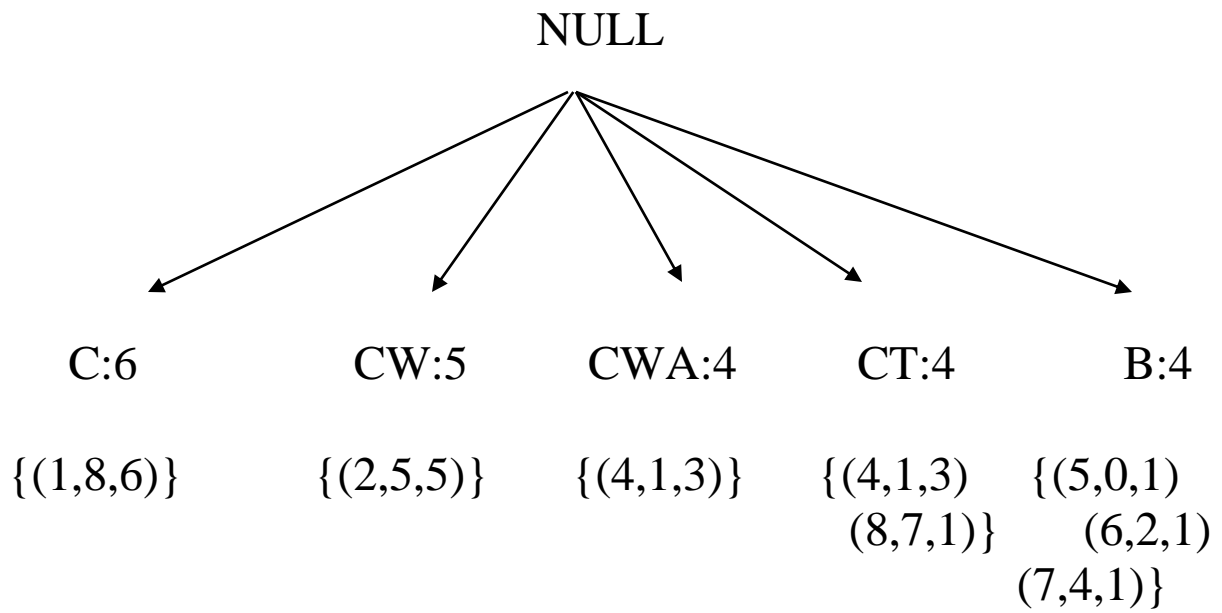
TRANSACTIONS	ORDERED FREQUENT ITEMSETS
1	C,W,A,T
2	C,W,D
3	C,W,A,T
4	C,W,A,D
5	C,W,A,T
6	C,T,D

## 6.3.PPC TREE:

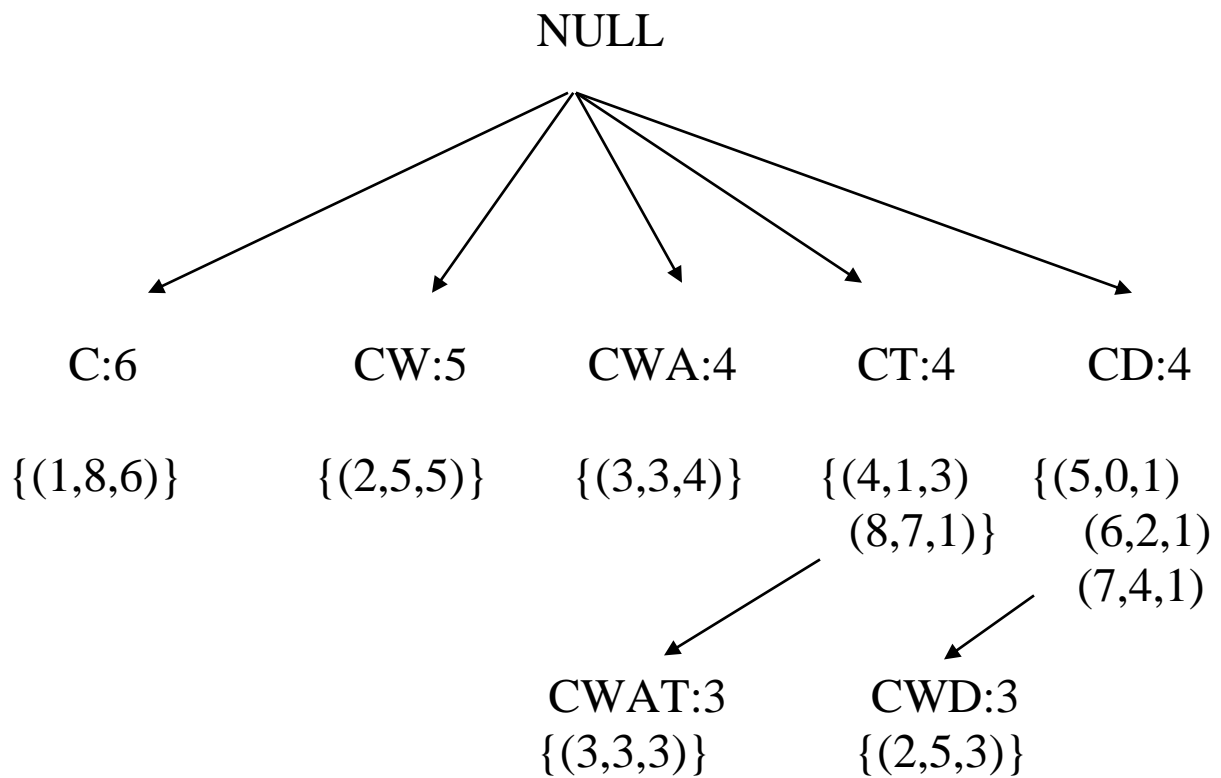




#### 6.4.FREQUENT 1-PATTERN AND THEIR N-LISTS:



6.5.FCP'S IDENTIFIED USING NAFCP FOR EX WITH  
MINSUP=50%:



## **6.6. Algorithm for finding frequent closed pattern:**

Input: Dataset, minimum support .

Output: the set of all frequent closed patterns



```

1 Construct_PPC_tree( $DB, minSup$ ) to generate  $R, I_1, H_1$ , and  $threshold$ 
2 Generate-N-list( $R$ )
3 Find-FCIs( $I_1$ )
4 return  $FCIs$ 

```

```

1 Procedure Generate-N-list( $R$ )
2 Let  $C \leftarrow \langle pre(R), post(R), f(R) \rangle$ 
3 Let  $H = H_1[n(R)]$  is the node of  $n(R)$  in  $H_1$ 
4 Add  $C$  to  $NL(H)$ 
5  $f(H) += f(R)$ 
6 for each  $child$  in  $childs(R)$  do
7   Generate-N-list( $child$ )

```

```

1 Procedure Find-FCIs( $Is$ )
2 for  $i \leftarrow |Is| - 1$  to 0 do
3    $FCIs_{next} \leftarrow \emptyset$ 
4   for  $j \leftarrow i-1$  to 0 do
5     begin for
6       if N-list-check( $NL(Is[i]), NL(Is[j])$ ) = true then
7         if  $f(Is[i]) = f(Is[j])$  then
8            $Is[i] = Is[i] \cup Is[j]$ 
9           remove  $Is[j]$ 
10           $i--$ 
11        else
12           $Is[i] = Is[i] \cup Is[j]$ 
13          for each  $f$  in  $FCIs_{next}$ 
14            update  $f = f \cup Is[j]$ 
15          continue
16        end for
17     $FCI \leftarrow Is[i] \cup Is[j]$ 

```

```

18    $NL(FCI)$  and  $f(FCI) \leftarrow \text{N-list-intersection}(NL(Is[i]), NL(Is[j]))$ 
19   if  $f(FCI) \geq \text{threshold}$  then
20     add  $FCI$  to  $FCIs_{\text{next}}$ 
21   if  $\text{Subsumption-check}(Is[i]) = \text{false}$  then
22     add  $Is[i]$  to  $FCIs$ 
23     add index of  $FCI$  in  $FCIs$  to  $\text{Hash}[f(FCI)]$ 
24   Find-FCIs( $FCIs_{\text{next}}$ )

```

```

1   Function N-list-check( $N_1, N_2$ )
2   let  $i = 0$  and  $j = 0$ 
3   while  $j < |N_1|$  and  $i < |N_2|$  do
4     if  $\text{pre}(N_2[i]) < \text{pre}(N_1[j])$  and  $\text{post}(N_2[i]) > \text{post}(N_1[j])$  then
5        $j++$ 
6     else
7        $i++$ 
8   if  $j = |N_1|$  then
9     return true
10  return false

```

```

1   Function Subsumption-check( $FCI$ )
2   let  $Arr \leftarrow \text{Hash}[f(FCI)]$ 
3   for  $i \leftarrow 0$  to  $|Arr|-1$  do
4     if  $FCI \subseteq FCIs[Arr[i]]$  then
5       return true
6   return false

```

## **EXPERIMENTAL RESULT**

## 7. EXPERIMENTAL RESULTS

### 7.1. CHARACTERISTICS OF EXPERIMENTAL DATASETS

The experimental datasets had different features. The Accidents, Chess, Connect, Mushroom datasets had many number of transactions and type but had very few numbers of items.

### 7.2 DATASET USED

DATASET	TYPE	NO. OF TRANS	NO. OF ITEMS
Accidents	Sparse	140183	468
Chess	Dense	3196	76
Connect	Dense	67557	130
Mushroom	Dense	8124	120

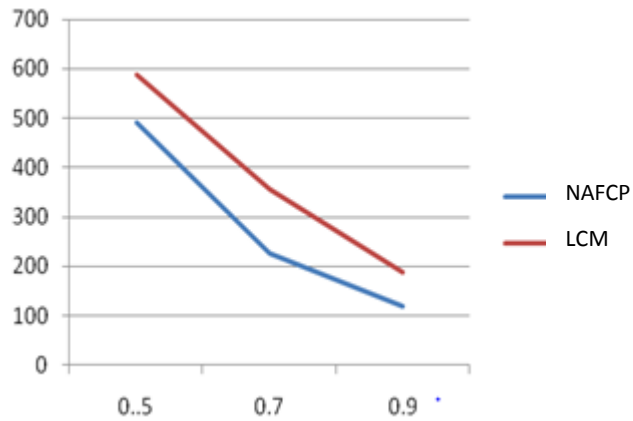
### EXPERIMENTAL RESULT:

The below figure shows the execution time difference between the weighted class association rule mining using lattice and weighted class association rule mining using hashing data structure.

The X axis shows the support and Y axis shows the execution time.

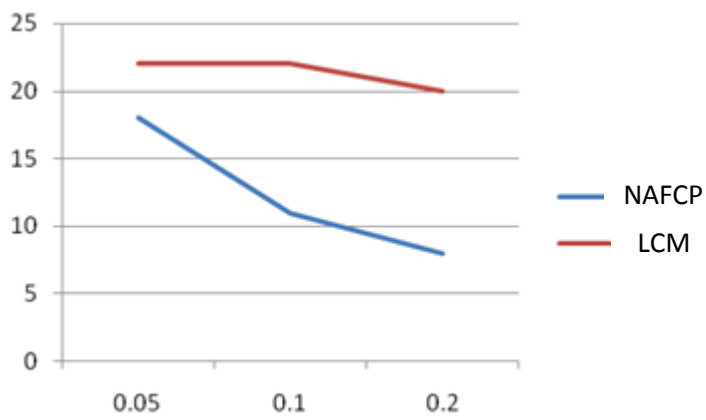
## ACCIDENT DATASET:

Figure 1: Execution time difference between NAFCP and LCM for Accident dataset.



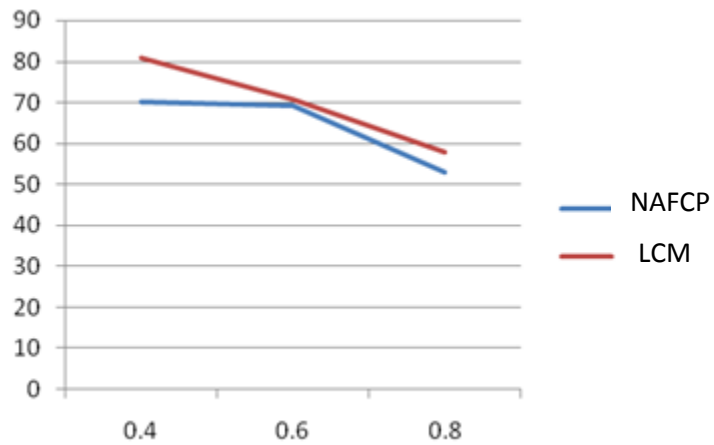
## CHESS DATASET:

Figure 2: Execution time difference between NAFCP and LCM for Chess dataset.



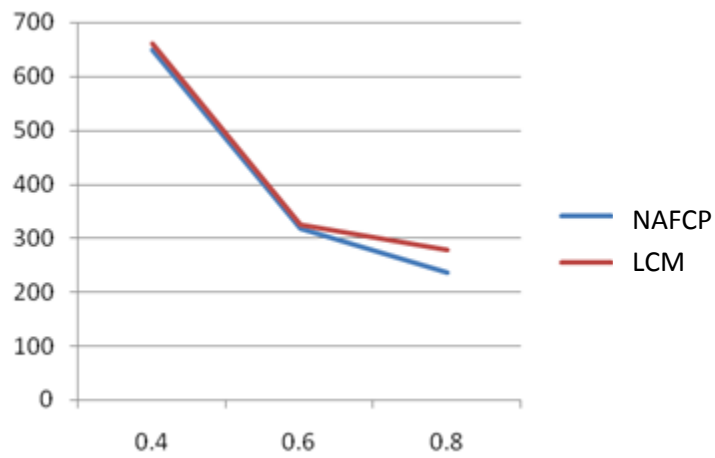
### CONNECT DATASET:

Figure 3: Execution time difference between NAFCP and LCM for Connect dataset.



### MUSHROOM DATASET:

Figure 4: Execution time difference between NAFCP and LCM for Mushroom dataset.

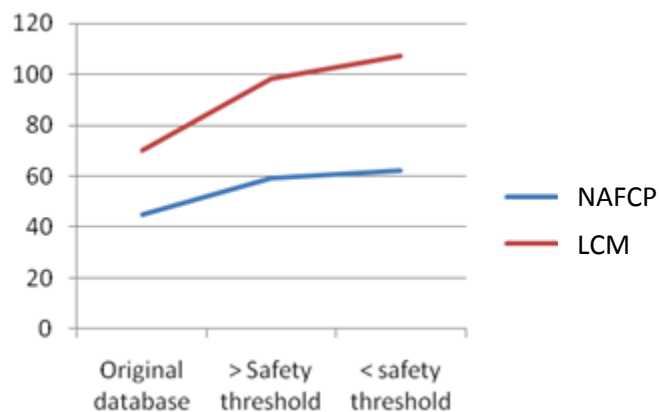


### Execution time difference for incremental database:

The below figure shows the execution time difference between the algorithm of NAFCP and LCM. The X axis shows incremental database details and Y axis shows the execution time.

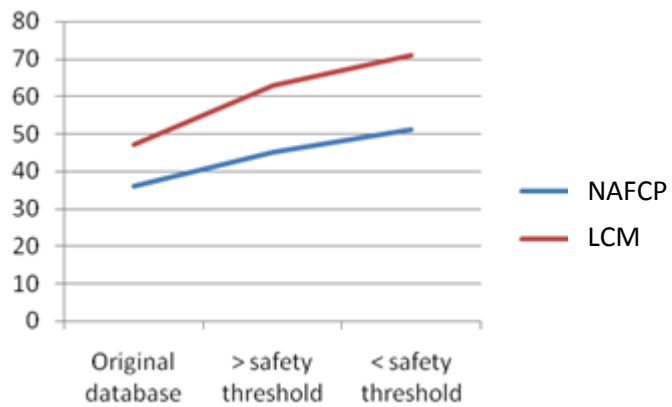
### ACCIDENT INCREMNTAL DATASET:

Figure 5:- Execution time difference between NAFCP and LCM for accident incremental dataset.



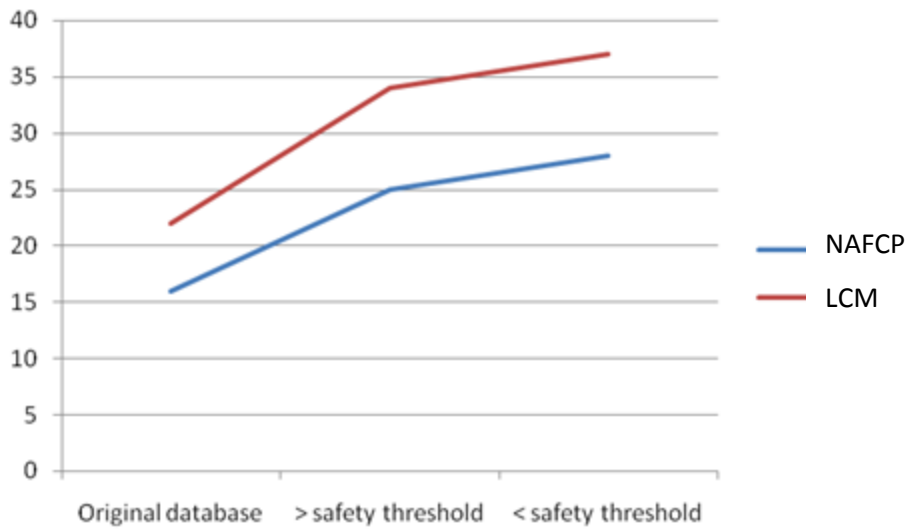
### CHES INCREMNTAL DATASET:

Figure 6:- Execution time difference between NAFCP and LCM for chess incremental dataset.



### CONNECT INCREMNTAL DATASET:

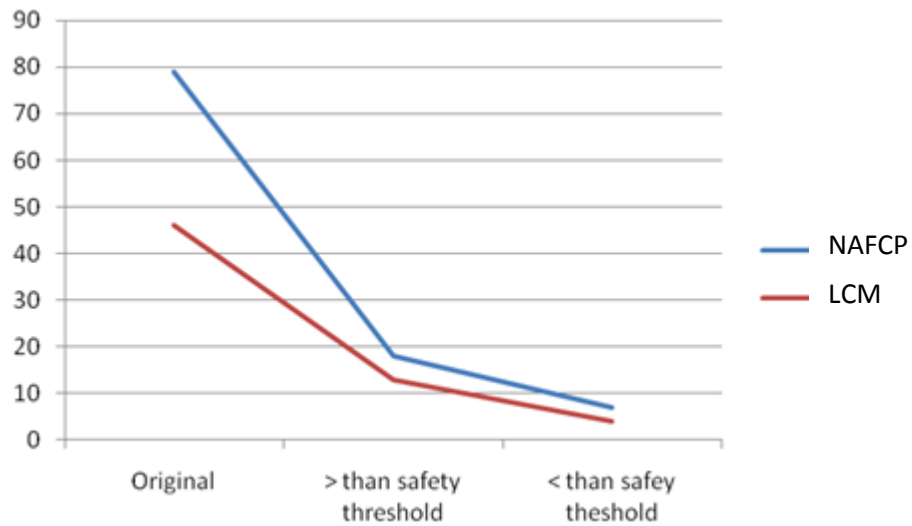
Figure 7:- Execution time difference between NAFCP and LCM for connect incremental dataset.





## MUSHROOM INCREMNTAL DATASET:

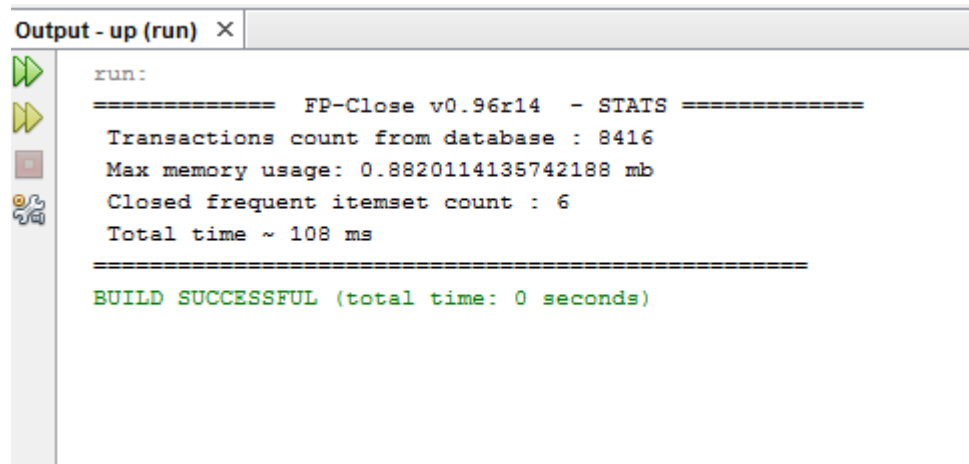
Figure 8:- Execution time difference between NAFCP and LCM for mushroom incremental dataset.



## **SNAPSHOTS**

## 8. SNAPSHOTS

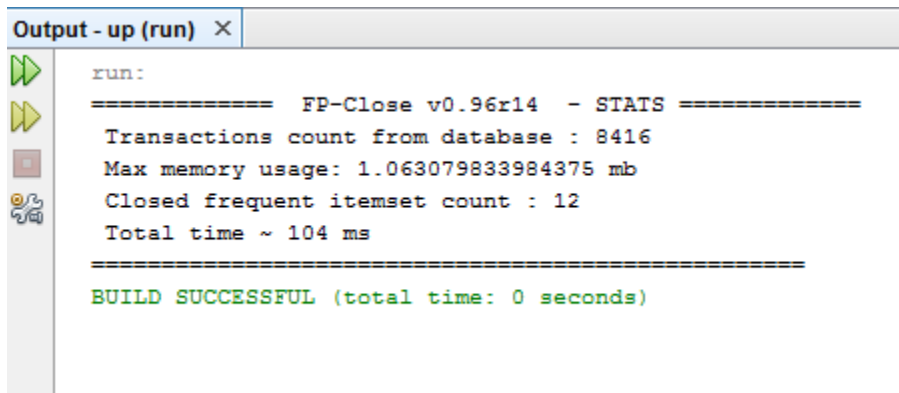
Figure 9: Mushroom dataset –NAFCP Algorithm- Minimum SUPPORT:0.9



```
run:
===== FP-Close v0.96r14 - STATS =====
Transactions count from database : 8416
Max memory usage: 0.8820114135742188 mb
Closed frequent itemset count : 6
Total time ~ 108 ms
=====
BUILD SUCCESSFUL (total time: 0 seconds)
```

```
90 97 #SUP: 7768
90 36 97 #SUP: 7576
90 36 #SUP: 8200
90 94 36 #SUP: 8192
90 94 #SUP: 8216
90 #SUP: 8416
```

Figure 10: Mushroom dataset –NAFCP Algorithm- Minimum SUPPORT:0.7



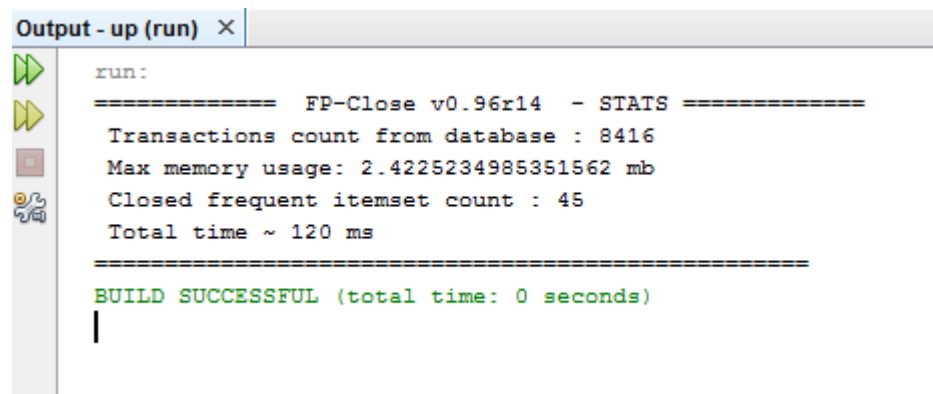
```
run:
===== FP-Close v0.96r14 - STATS =====
Transactions count from database : 8416
Max memory usage: 1.063079833984375 mb
Closed frequent itemset count : 12
Total time ~ 104 ms
=====
BUILD SUCCESSFUL (total time: 0 seconds)
```

```

90 97 38 #SUP: 6464
90 94 36 97 38 #SUP: 6272
90 94 36 38 #SUP: 6608
90 94 38 #SUP: 6632
90 38 #SUP: 6824
90 36 97 #SUP: 7576
90 94 36 97 #SUP: 7568
90 97 #SUP: 7768
90 36 #SUP: 8200
90 94 36 #SUP: 8192
90 94 #SUP: 8216
90 #SUP: 8416

```

Figure 11: Mushroom dataset –NAFCP Algorithm- Minimum SUPPORT:0.5

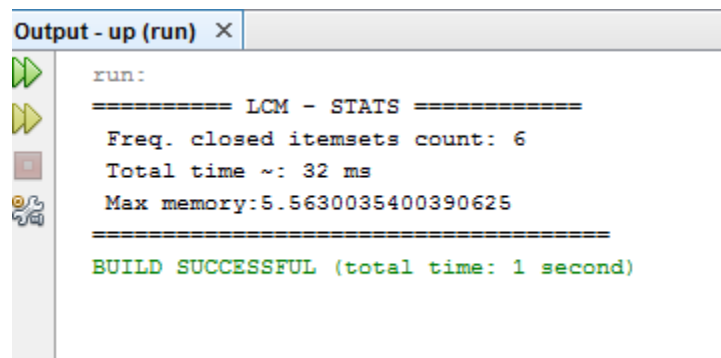


```

run:
===== FP-Close v0.96r14 - STATS =====
Transactions count from database : 8416
Max memory usage: 2.4225234985351562 mb
Closed frequent itemset count : 45
Total time ~ 120 ms
BUILD SUCCESSFUL (total time: 0 seconds)

```

Figure 12: Mushroom dataset –LCM Algorithm- Minimum SUPPORT: 0.9

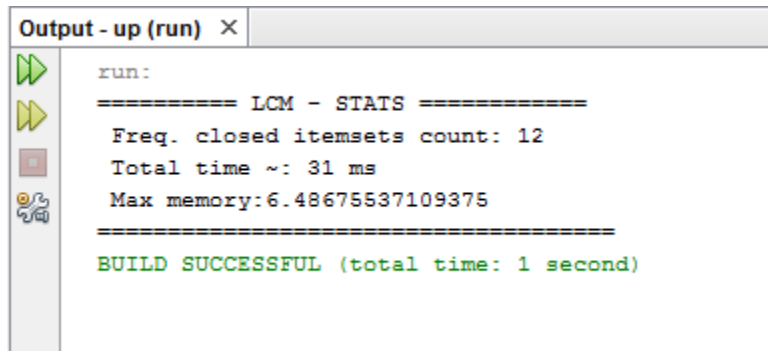


```

run:
===== LCM - STATS =====
Freq. closed itemsets count: 6
Total time ~: 32 ms
Max memory:5.5630035400390625
BUILD SUCCESSFUL (total time: 1 second)

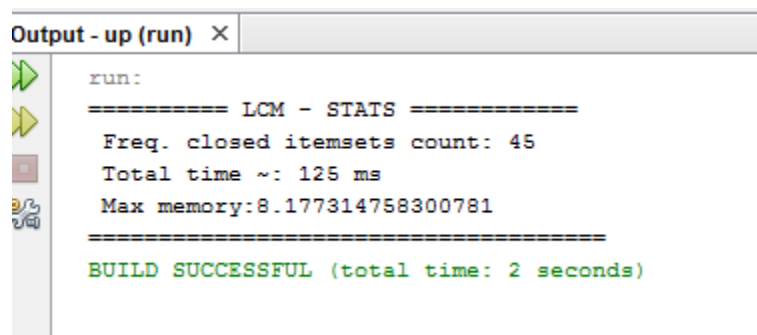
```

Figure 13: Mushroom dataset –LCM Algorithm- Minimum SUPPORT: 0.7



```
run:
==== LCM - STATS =====
Freq. closed itemsets count: 12
Total time ~: 31 ms
Max memory: 6.48675537109375
====
BUILD SUCCESSFUL (total time: 1 second)
```

Figure 14: Mushroom dataset –DCI Algorithm- Minimum SUPPORT: 0.5



```
run:
==== LCM - STATS =====
Freq. closed itemsets count: 45
Total time ~: 125 ms
Max memory: 8.177314758300781
====
BUILD SUCCESSFUL (total time: 2 seconds)
```

## **CONCLUSION AND FUTURE WORK**

## CONCLUSION AND FUTURE WORK

This paper proposed an effective algorithm for mining FCPs using the N-list structure in terms of mining time and memory usage. First, two theorems supporting NAFCP, based on the N-list structure, were developed. Then, using these theorems NAFCP was proposed. Finally, in order to confirm the effectiveness of the proposed algorithm in terms of runtime and memory usage, experiments were conducted on a number of datasets NAFCP and DCI. The experimental results show that NAFCP outperforms DCI in terms of runtime and memory usage in most cases. Using the proposed algorithm, expert and intelligent systems that use FCPs can improve their performance.

In future we are going to focus on optimizing the N-list structure to improve mining time and memory usage of mining the mining of frequent maximal patterns, top-rank-k FCPs .

## **REFERENCE**



## REFERENCE:

- An N-List based algorithm for mining frequent closed pattern, Tuong Le, Bay Vo.
- Deng, Z. H., & Lv, S. L. (2015). PrePost+: An efficient N-lists-based algorithm for mining frequent itemsets.
- Pei, J., Han, J., & Mao, R. (2000). CLOSET: An efficient algorithm for mining frequent closed itemsets. In Proceedings of the 5th ACM-SIGMOD workshop on research issues in data mining and knowledge discovery (pp. 21–30). Dallas, Texas, USA.
- Vo, B., Le, T., Coenen, F., & Hong, T. P. (in press). Mining frequent itemsets using the N-list and subsume concepts. International Journal of Machine Learning and Cybernetics.
- Zaki, M. J., & Hsiao, C. J. (2002). CHARM: An efficient algorithm for closed itemset mining.