# Deriving Frequent Itemsets from Lossless Condensed Representation

**A. Subashini, M. Karthikeyan**

***Abstract:-** In data mining, major research topic is frequent itemset mining (FIM). Frequent Itemsets (FIs) usually generating a large amount of Itemsets from database it causing from high memory and long execution time usage. Frequent Closed Itemsets(FCI) and Frequent Maximal Itemsets(FMI) are a reduced lossless representation of frequent itemsets. The FCI allows to decreasing the memory usage and execution time while comparing to FMIs. The whole data of frequent Itemsets(FIs) may be derived from FCIs and FMIs with correct methods. While various study has presented several efficient approach for FCIs and FMIs mining. In sight of this, that we proposed an algorithm called DCFI-Mine for capably derive FIs from Closed FIs and RFMI algorithm derive FMIs to FIs. The advantages of DCFI-Mine algorithm has two features: First, efficiency, different existing algorithm that tends to develop an enormous quantity of Itemsets all through process, DCFI-Mine process the Itemsets straight without candidate generation. But in proposed RFMI multiple scan occurs due to search of item support so efficiency is less than proposed algorithm DCFI-Mine. Second, in terms of losslessness DCFI-Mine and RFMI can discover complete frequent itemset without lapse. Experimental result shows That DCFI-Mine is best deriving FIs in term of memory usage and executions time.*

***Keywords:** Deriving algorithm, Frequent itemset mining, maximal itemset, closed, itemset mining, Lossless condensed representation.*

## I. INTRODUCTION

The purpose of FIs(Frequent Itemsets) mining [1]-[3] is to determine the sets of item in database looking frequently. Those skills have been generally applicable to many real-life application are association analysis [4], text mining [6], bioinformatics [7] and web mining [5]. Numerous effective procedures has been established for mining FIs are Apriori [1], Eclat [8] and FP-Growth [7]. When FIs mining from databases type is sparse, these procedures typically have performance good because the correlation of items between in sparse database are comparatively weak, then the FIs length in the database are too comparatively short. The traditional FIs mining techniques be able to make use of the downward closure property [1], [11] to prune effectively the space of search. A typical data type Sparse transaction dataset examples are used in real-life situations, several types of dense dataset are as features of plant(e.g., Mushrooms datasets [10]), census statistical data (e.g., the Pumsb datasets [10]), and records in games steps (e.g., the Connect and Chess datasets [10]). To resolve the overhead difficulties, various study has enthusiastic to develop condensed representation of Frequent Itemsets. The experiment result of prior study [11]-[14] has present the condensed representation which can be greatly decrease the memory usage and execution times from dense dataset. Several type of condensed representation of FIs has been propose, like maximal itemsets [12], free itemsets [11], maximal itemsets [12], closed itemsets [14], and generator itemsets [13]. Between those representations, (maximal itemsets) FMIs and FCIs is one of the popular one. The frequent maximal itemsets (FMI) and FCI is a lossless reduced representation [13] of Frequent Itemsets. Through the exact algorithms, entire FIs with their support be able to derive completely from all the FMIs and FCIs. So, the whole set of FMIs, FCIs are preserves the whole data without loss of Frequent Itemsets. The various studies [13], [15]-[17] had efficiently proposed for mining FMIs, some of them think through to develop an algorithm for deriving efficiently FIs from FMIs and FCIs (derive an algorithm used for easiness). But evolve deriving efficient algorithm is a significant works for both FCIs and FMIs. When every FIs cannot stay positively mined from dense data type, an another solution, to mine whole FCIs and FMIs from dataset, next apply deriving effective algorithm to derive the set of complete Frequent Itemsets with their supports also from FMIs and FCIs.

A maximal frequent itemsets has no super subsets is called frequents. Though FIs be able to condense for mines a “border" in an itemsets lattice, as found as in [9]. The whole itemset below the borders are all frequent and above the border are infrequent. Bayardo [19] intros MaxMiner, that extend Apriori for mining only “lengthy" pattern (FMI). To decrease, the space of search. Then we able to decide that some of its subsets was also a frequent so supersets of frequency prune decreases the time of search radically, MaxMiner need several pass to acquire whole lengthy pattern. Though, the LevelWise traditional deriving algorithm [14] adopts topdown strategy and breadth-first search to derives the FIs. This one derives the FIs of (k-1) length from closed FIs and FIs of k lengths. Though, traditional derive algorithms [12] implements topdown and breadth-first search approach to derive FIs is called levelwise algorithm.

Revised Manuscript Received on February 05, 2020.
* Correspondence Author
**A.Subashini,** Assistant Professor in the Department of Computer Application, Government Arts College, C.Mutlur, Chidambaram, Tamil Nadu, India.
**M.Karthikeyan,** Assistant Professor in the Division of Computer and Information Science, Annamalai University, Annamalai Nagar, Chidambaram, Tamil Nadu, India.

A weakness of a LevelWise algorithm that need to preserve the whole sets of the FIs of k length and (k-1) length in memory for duration of the derive process. Hence, when there was numerous FIs of k length and (k-1) length, algorithm levelwise will takes a large memory and also running out of memory due to deriving task. Likewise, the deriving procedure of the LevelWise consist of a huge number of search tasks, results in lengthy executions time problematic. In sight of above, the proposed algorithm, named DCFI-Mine (Discovering Compact Frequent Itemset is Pattern Growth based), for effective derive FIs and with the support from FCIs. An ideas design of DCFI-Mine construct the FP-Tree first to preserve the data of FCIs, then from FP-tree the FIs is generated using FP-Growth. However, traditional algorithm FP-Tree and FP-Growth is not considered for derive task then applying straight will results improper result. So, we proposed two approaches then alter the algorithm process of FP-growth and FP-tree, constructing those to derive the correct data of FIs from the FCIs. The proposed approaches are respectively call the selecting maximum supports and replace maximum supports. Then previous method is applying through building of FP-Tree and pattern generate latter. The experiments that we performs on Dense and sparse real-life dataset to estimate the presentation on the propose algorithm called DCFI-Mine. Extensive experiment shows that DCFI-Mine is fairly efficient and derive all FIs with supports. RMFI (Recover all Frequent Itemsets from maximal Items) to recover entire FIs from set of maximal FIs. Additionally, DCFI-Mine comparing the performance is considerably better than RFMI algorithms in term of execution time and memory usages.

**Table I. An example of a transaction database**

| ***TID*** | ***TRANSACTION*** |
|---|---|
| **1** | **A E O U** |
| **2** | **E I U** |
| **3** | **A E O U** |
| **4** | **A E I U** |
| **5** | **A E I O U** |
| **6** | **E I O** |

## II. BASIC CONCEPT AND DEFINITIONS

Let assume P= {$I_1$, $I_2$… $I_N$} be a finite set of distinct items. A transaction databases Ɗ = {$t_1$, $t_2$,…, $t_K$} is a transaction sets, where every transactions $t_r$ ϵ Ɗ ($1 \le r \le K$) is a subsets of P and it has sole identifier transactions ‘r’. An itemsets is a sets of item. If entire item of an itemsets Z can contain in a transactions T, Z is supposed to be contain in T, which was denote as $Z \subset T$.

**Definition: 1 (Tidset).** Transaction identifiers set of an itemsets Z is denote as r(Z) and the set of transactions_ID(identifiers) defined as entire transaction contains Z in Ɗ.

**Definition: 2 (Itemset Length).** Length of an itemset of Z = {$I_1$, $I_2$… $I_L$} is define as L, where L was the complete distinct item in Z.

**Definition: 3 (Itemset Support of items).** Support count of an itemset of X was denoted as SCI(Z) and defined as |r(Z)|. Moreover, the support of Z is defined as SCI(Z)/| Ɗ |, where | Ɗ | is the total number of transactions in Ɗ.

**Definition: 4 (Frequent itemset).** An itemsets Z is called as frequent itemsets if and only if the supp(Z) isn’t less than a user-specified minimum support threshold θ ($0 < \theta \le 1$). Otherwise, Z is low support.

**Definition: 5 (Closed itemset).** An itemsets Z is called as closed itemsets if it has no proper superset *Y* in Ɗ *such* that SCI(Z) = SCI(Y). Else, Z is not a closed itemset. The complete set of closed itemsets is denoted as C.

**Definition: 6 (Frequent closed itemset).** A closed itemset Z is called frequent closed itemsets If and if only the support(X) was not less than the user-specified minimum support threshold θ ($0 < \theta \le 1$). The complete set of Frequent closed itemset is denote as FCI.

**Definition: 7(Maximal itemset).** An itemsets Z is called Maximal itemsets if none of its immediate superset is frequent. Else, Z is not a maximal itemset.

**Definition: 8. (Frequent Maximal itemset).** An itemset Z is called maximal itemsets iff the support of Z was not less than the user-specified minimum support threshold θ ($0 < \theta \le 1$). The complete set of Frequent Maximal Itemset is denoted as FMI.

**Problem Statement for Maximal Itemset Mining:** Given a dataset Ɗ with itemset support, and a user-specified min_sup threshold, the problematic is to determine the whole set of FMIs in Ɗ. Then compared frequent closed itemset with frequent maximal Itemset from altered viewpoints. In terms of compression, the sets FMI is lesser than FCI because FCI is a subsets of FMI. A FMIs only says the user, certain of its subset can be FIs. But FCIs are additional significant as this are lossless and first eliminate redundancy [20, 21]. In terms of recovering, representation of both can be used to recover entire FIs, then it’s extra costly with FMIs. This is a reason that FMIs aren’t lossless. So, asking the support of a FIs using FMIs needs to make a surplus databases scan, though it’s not essential for closed FIs. Both FMI and FCI are represent because both of them can stay used to find entire FIs.

## III. THE PROPOSED METHOD

### 3.1 Derive Frequent Itemsets from Closed Itemsets:

In this division the proposed algorithm DCFI-Mine (Deriving Frequent Itemsets from closed itemset). In first approach of function InputFCI was developed based on the modification of FP-Growth and FP-Tree refer to [2]. And second approach RFMI function deriving FIs from Maximal frequent itemset. In first approach: The FP-Tree is a tree based structure adopted in DCFI-Mine. The DCFI-Mine, instead of transactions of the FP-Tree are used to retain the data of FCIs. The FP-Tree of every node is denotes as ‘T’ and an items denotes as I Ɛ I*, it consists of fields five: T.Names, T.Counts, T.Parents, T.Childs and T.Links. In T.Names and T.Counts field stores respectively the names of item ‘I’ and value of count ‘I’. The T.Parents and T.Childs field stores respective nodes of parent ‘T’ and a nodes of children ‘T’. The T.Links fields store a nodes link, that which point to next nodes have the items name as same ‘T’. Every FP-Tree was associates with the header tables. The information of items records in a header table and those items are every sub-itemset of FCIs (Frequent closed itemsets). The HT header table, every items I Ɛ I* in HT consist of three field: HT[I].Names, HT[I].Counts and HT[I].Links, that which correspondingly registers the names of items

'I', a values of counts 'I', and a link nodes point a FP-Tree nodes have the name of item same as I. refer[2].

## A. Construction of a FP-Tree for Deriving Task

This subdivision intros the building a FP-Tree to derive process with following four stages.

**Stage 1.** Let a C is the entire itemset of FCIs (Frequent Closed Itemset) exposed from a database Ɗ. In DFCI-Mine 'C ' scans one time and derive their support count of every items looking in C rendering towards the Selecting Maximum Support of Item (SMS-I) approach.

**Fig.1(a).FCM with Support count**

| *Frequent Closed Itemset* | *Support Count* |
|---|---|
| {EUA} | 4 |
| {EU} | 5 |
| {EI} | 4 |
| {E} | 6 |
| {EO} | 4 |
| {EUI} | 3 |
| {EUAO} | 3 |

**Fig. 1(b). An example of Selecting Maximum Support Item (SMS-I)**

| *Itemset* | *Selecting Maximum Support Item* | *Support Count* |
|---|---|---|
| {E} | Max{SCI({EUA}),SCI({EU}), SCI({E}), SCI({EO}),SCI({EUI}), SCI({EUAO})} | 6 |
| {U} | Max{SCI({EUA}),SCI({EU}), SCI({EUI}), SCI({EUAO})} | 5 |
| {A} | Max{SCI({EUA}),SCI({EUAO})} | 4 |
| {I} | Max{SCI({EI}),SCI({EUI})} | 4 |
| {O} | Max{SCI({EO}),SCI({EUAO})} | 4 |

**Method 1 (SMS-I).** Let an item be I and S(I) = {$c_1$, $c_2$, …, $c_z$} be the set of entire FCIs contain I. The support count of I was equivalent to max{SCI($c_1$), SCI($c_2$), …, SCI($c_z$)}. Fig.1 shows that left table is closed itemset with minimum support count is 3 and in right table explains the selection of maximum support item approach.

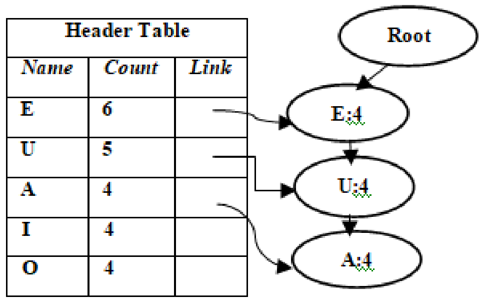
**Stage 2.** Create an H header table and items placed into header table H by a stable sort in order f. next, create a root node R of T FP-Tree. Then, scans C one time and sorts item in every FCIs by the sorted order f.

**Stage 3.** Every arranged FCI Inserting into T FP-Tree by calling the function InputFCI. The pseudo code of function InputFCI was presented in Fig. 2. Every time when a FCI, X = {$I_1$, $I_2$… $I_K$} is recovered, the two variables T and w are sets to R root node and 1 respectively. Moreover, set a variable $I_w$ to $I_1$, where $I_1$ is first items of closed itemset X. next, calling the InputFCI (T, $I_w$) function insert X in to the T FP-Tree. The Function InsertFCI(T, $I_t$) process work as following. If the T nodes has a T′(child nodes) so that T′.Names = $I_w$, then an algorithms put on the replacement maximum support (RMS) approach for setting T′.Counts.

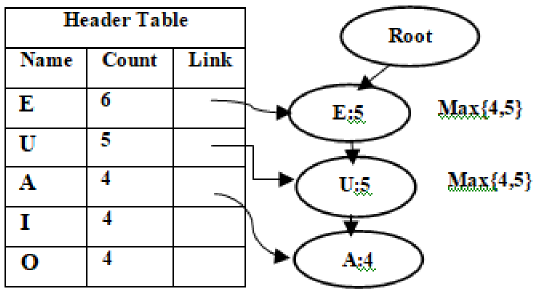
**Fig.2. The Pseudo code of the Function**

```
Function : InputFCI(T,I_W)
01 If(T has a child T' such that T'.names == Iw)
02     then T'.counts max{T'.counts, SCI(X)};
03 else
04     Create a new child node T' and
05     Set T'.names Iw, T'.counts SCI(X), T'.parent T;
06 Set T'.links to the lastly created node having the same
   item  name as Iw;
07 w ← w +1, If(w k), call InputFCI(T, Iw);
```

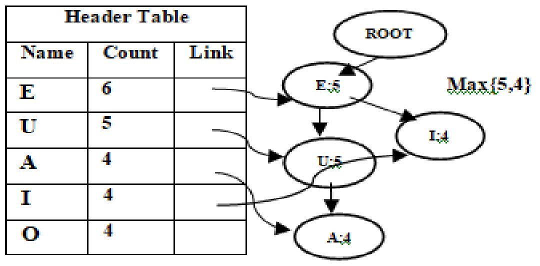
**Method 2 (RMS).** If SCI(X) >T′.Counts, then swapping T.Counts by SCI(X). Else, if SCI(X) isn't greater than T′.Counts, then it creates a replacement node T′ and it sets T child node. Besides, the T′.Counts and T′.Links set toward support counts of X and finally nodes was created which have items in same name as $I_w$, respectively in lines 3 and 6 .Following, increments the variable w by 1. then, if w isn't greater than K, then call the repetitive function InputFCI (T′, $I_w$) for process the $_w$-th items of X in Lines 7.In Fig.3 show an example ,insert closed itemset Figure 3(a),3(b)-interchange maximum support count for item E:5 and U:5,same as in 3(c) and 3(d) also.



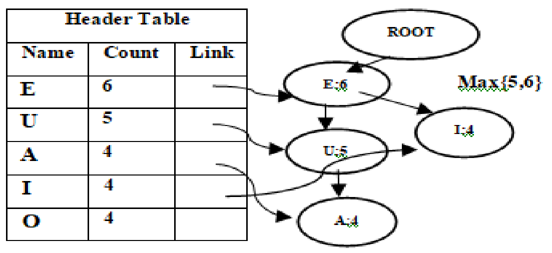
**(a). Input{EUA}:4**



**(b). Input {EU}**



**(C). Input{EI}**



**(d).**

**Input{E}**

**Fig.3 .An example replacing Maximum Support (RMS)**

**Stage 4.** All through FP-Tree structure process, entire nodes having the item name same will be link together via the link nodes of HT header tables and associated node in T FP-Tree. Over those link nodes, DCFI-Mine allows towards nodes travel have items name as same powerfully.

### B. Deriving Frequent Itemsets:

Afterward constructing T FP-Tree, HT header tables then called function DFCI-Mine (HT, F, β) towards deriving FIs from FP-Tree T, the variables α is initialized to φ. This process of the function DFCI-Mine (HT, F, α) working as following.

**Stage 1**. This algorithm visit every items of I in HT header tables. For every visited items, the algorithms output{β ∪ I} then HT[I].Counts as FIs and with their support count. Over the link nodes in HT[I].Links, DFCI-Mine pass through entire node having the name of same item as I in T FP-Tree.

**Stage 2.** Let $\alpha = \{\alpha_1, \alpha_2, \ldots, \alpha_n\}$ a set of node I have its name of item is same. For every node $\alpha_i \in \alpha$ $(1 \le i \le n)$, this algorithm visit from parent nodes of $\alpha_i$ to R root node of T. Then, this algorithms collect the item of each visited nodes. Those item form a sequences $C_i$ $(1 \le i \le n)$ and that sequences is called conditional itemsets. Each conditional itemset $C_i$ was associated with a counts called count of conditional itemsets and was denote as $CCI(C_i)$. This count of conditional itemsets $C_i$ is primarily set to $\alpha_i$.Counts. Then handling entire nodes in α in same manner, conditional databases of the itemsets{αUI} was acquired. Let as $Ɗ_{\{\alpha \cup I\}} = \{C_1{:}CCI(C_1), C_2{:}CCI(C_2), \ldots, C_m{:}CCI(C_n)\}$ denoted the conditional databases of {α ∪ I}.

**Stage 3.** Scan the conditional databases once $Ɗ_{\{\alpha \cup I\}}$ to discover entire distinct item in the conditional itemset. For every items X, this algorithms evaluates the support counts of frequent itemsets{β ∪ I ∪ X} through the proposed Selecting Maximum Support for itemsets(SMS-IS) approach.

**Method 3. (SMS-IS)**.Let X be an items appeared in $Ɗ_{\{\beta \cup I\}}$ then a set of entire conditional itemsets be $C^* = \{C'_1{:}CCI(C'_1), C'_2{:}CCI(C'_2), \ldots\ldots\ldots, C'_h{:}CCI(C'_h)\}$ contain X in $Ɗ_{\{\alpha \cup I\}}$. Then sup_count of a frequent itemsets{α ∪ IUX}wasequivalent to $\max\{CCI(C'_1), CCI(C'_2) \ldots CCI(C'_h)\}$ , wherever $CCI(C'_1)$ was the conditional itemset count of $C'_1 (1 \le i \le n)$. Fig.4, shows a conditional databases of itemsets{O} are {EUA}:3 and {E}:4. These are the two conditional itemsets {EUA}:3 and {C}:4. In the steps of algorithm, the worth of the variables α is currently φ. The conditional itemsets covering the items{E} is {{EUA}:3, {E}:4}. The support count of {φUOUE}={OE} is Max{CCI({EUA}), CIC({E})} = max{3, 4} = 4. Therefore, this algorithms output{OE} as a FIs and its support count is 4.

**Fig.4(a) Conditional Database of {O}**

| Conditional Database of {O} | |
|---|---|
| *Conditional Itemset* | *Conditional Itemset Count* |
| **{EUA}** | **3** |
| **{E}** | **4** |

**Fig.4. An example of the Selecting Maximum Support for ItemSet (SMS-IS)**

**Max{5,4}**

| *FIs* | *Selecting Maximum Support* | *Sup_count* |
|---|---|---|
| **{OE}** | **Max{CCI({EUA}),CCI({C})}** | **4** |
| **{OU}** | **Max{CCI({EUA})}** | **3** |
| **{OA}** | **Max{CCI({EUA})}** | **3** |

**Stage 4.** This algorithm locates item in $Ɗ_{\{\alpha \cup I\}}$ into a novel HT' header tables in the sort order f. Then, algorithms scan conditional databases $Ɗ_{\{\alpha \cup I\}}$ for a next time and sort item in every conditional itemsets in $Ɗ_{\{\alpha \cup I\}}$ given to f.

**Stage 5.** Create a novel R′ nodes as the root of a novel T′ FP-Tree for the itemsets{α ∪ I}. Insert each sort conditional itemsets in $Ɗ_{\{\alpha \cup I\}}$ into T′ to build the conditional FP-Tree of itemsets{α ∪ I}. All time after a sort conditional itemsets J is insert then sets the variable N and $I_w$ into R and $I_1$, next calling a function InputFCI(T, $I_w$). When construct T′, entire node in T′ have the name of items same that was link organised by the link nodes.

**Stage 6.** When constructing T′ FP-Tree has node other than the R′ root nodes, next to derive FIs call the function DFCI-Mine (H′, T′, α ∪ I).

### 3.2 Recover all Frequent Itemset from Maximal Items:

In this section, the proposed algorithm named as RMFI (Recovers all Frequent Itemset from maximal Items) is efficient to derive entire FIs from set of maximal FIs. The min_sup threshold and the set of FMIs are taken as input. RMFI recover entire FIs is outputs respecting to min_sup. RMFI process as follows. The set of maximal frequent itemset (FMI) allow recovering all FIs. FMIs defines, that a maximal frequent itemset not have proper superb subsets that's a frequent itemset. Thus, if an Itemsets is frequent, it is also a proper subsets of maximal itemsets. Figure.5 represents simple algorithms for recovering entire FIs from the set of FMIs. It produce entire subset of all the FMIs. Furthermore, it makes a check to identify if a FIs has previously been outputs in line 3 because a FIs can be a subset of other than one maximal Itemset. Afterward FIs had recovered and an additional scan in database can be executed to calculate their support, if required.

**Fig. 5 Algorithm to derive all FIs from FMIs.**

```
Function: RMFI (I/P: a set of Maximal Itemset FMI)
01  For each FIs j ϵ FMI
02      For each frequent k(j)
03          If k has not been output
04              Then output k.
```

## IV. RESULT AND DISCUSSION

In this division, we estimate the presentation of the proposed algorithm DFCI-Mine, RFMI algorithm and compared the both algorithms. This determination of two algorithms are same, which derives FIs with their supports from FMIs and FCIs. This experiment is directed on a computer equipped with Intel i5 with 8 GB of RAM, compile on Windows 10 OS. This algorithm was executed in Java programming language. Two real-life dataset that is Mushrooms (Dense) and BMS_WebView_1 (sparse) are used for the performance estimate. All the dataset are download from SPMF [10].

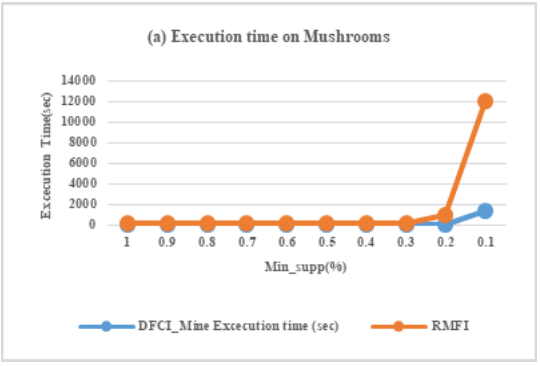
**Table II: Characteristics of Datasets**

| *Dataset* | *Number of distinct Items* | *Number of transaction* | *Type* |
|---|---|---|---|
| **Mushrooms** | **119** | **8124** | **Dense** |
| **BMS_WebView _1** | **59,602** | **497** | **Sparse** |

In Table II show the dataset characteristics. The executions time and its memory usage of algorithms was estimated. We extent the algorithm maximum memory usage by Java API. Figures.6 shows the executions time and the memory usage of compare algorithm. The executions time of DFCI-Mine and RFMI on the dense dataset that is Mushrooms are shown in Table III and in Fig. 6(a), 6(b) and on Sparse datasets BMS_WebView_1 are presented in Table IV and in Fig. 7(a) and 7(b), respectively. As exposed in those figure, the performance of DFCI-Mine is best compared to RFMI. For example, using Mushroom datasets, when the min_sup threshold is lesser than 1%, RFMI even can't complete the entire derive tasks due to running out of memory. However, our proposed DFCI-Mine takes minimum time to derive all FIs and with support from FCIs because of RFMI take on many scans for searching and used top-down methods to deriving all FIs from FMIs. RFMI occupy much times for search the similar item. Furthermore, it keep two level of itemset in the memory, that which occupies great memory space during the derive task. However, DFCI-Mine implements pattern growth with divide-and-conquer system to deriving all FIs, which divides the entire derive task into minor separate from one and so, normally it take less times than RFMI. In figures, we can witness that presentation between the RFMI and DFCI-Mine is lesser when the threshold was higher. It happens because that when the threshold was higher, the number of FCIs is lesser and the length of FCIs are relatively short. As shown in the experiment, no matter about the executions time or memory usages, the propose DFCI-Mine and RFMI generally has better performance of deriving all FIs.

**Table III: Mushrooms (dense) Datasets on both proposed algorithms**

| **Min_Supp (%)** | **Execution Time(s)** | | **Memory Usage(MB)** | |
|---|---|---|---|---|
| | ***DFCI-Mine*** | ***RFMI*** | ***DFCI-Mine*** | ***RFMI*** |
| **0.1** | **1415** | **12063** | **107.96** | **425.18** |
| **0.2** | **92** | **969** | **246.66** | **446.04** |
| **0.3** | **4** | **170** | **255.17** | **446.04** |
| **0.4** | **1** | **183** | **267.17** | **446.04** |
| **0.5** | **1** | **161** | **275.76** | **446.04** |
| **0.6** | **1** | **200** | **285.17** | **446.04** |
| **0.7** | **1** | **137** | **294.26** | **446.04** |
| **0.8** | **0** | **172** | **294.26** | **446.04** |
| **0.9** | **0** | **146** | **294.26** | **446.04** |
| **1** | **0** | **146** | **294.26** | **446.04** |



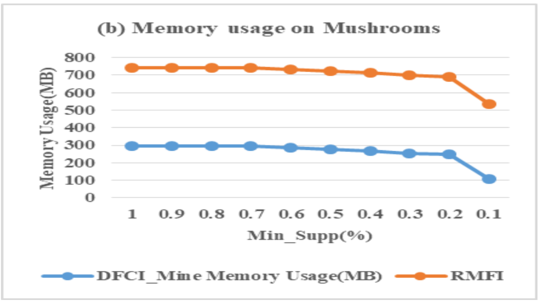
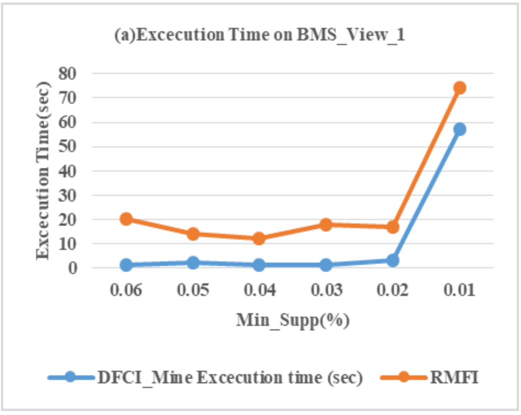


**Fig.6 Execution Time and Memory Usage of DFCI-Mine and RMFI on Dense dataset**

**Table IV: BMS_View_1 Datasets on both proposed algorithms**

| **Min_Supp (%)** | **Execution Time(s)** | | **Memory Usage(MB)** | |
|---|---|---|---|---|
| | ***DFCI-Mine*** | ***RFMI*** | ***DFCI-Mine*** | ***RMFI*** |
| **0.01** | **57** | **74** | **31.36** | **79.51** |
| **0.02** | **3** | **17** | **38.85** | **79.51** |
| **0.03** | **1** | **18** | **46.72** | **71.5** |
| **0.04** | **1** | **12** | **54.63** | **63.69** |
| **0.05** | **2** | **14** | **62.73** | **55.96** |
| **0.06** | **1** | **20** | **70.13** | **20.3** |



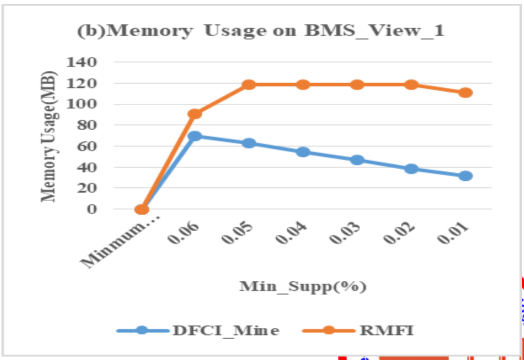


**Fig.7 Excecution Time and Memory Usage of DFCI-Mine and RMFI on sparse dataset**

## V. CONCLUSION

We propose an algorithm, named DFCI-Mine (Deriving Frequent Itemsets from closed Itemsets) and RFMI (Recover all FIs from maximal Itemsets) for the FIs deriving task. Likewise, we proposed two Methods in DFCI-Mine, named selecting maximum support of Itemsets (SMS-I) and replace maximum support (RMS), and join them into DFCI-Mine for efficient derive the whole FIs with its supports. The SMS-IS approach is proposed for computing the exact support with FIs by data from FCIs. In RMS approach is implemented through building FP-Tree, then its proposed set of the accurate info of FP-tree node. RMFI recover entire FIs from the set of maximal FIs. This one takes input as min_sup threshold with the set of FMIs. RMFI gives the entire FIs with respect to min_sup. Experiment result shows that the execution time and memory usage of DFCI-Mine are significantly better than that of the RFMI.

## REFERENCES

1. R.Agarwal and R.Srikant." Fast algorithms for mining association rules,"In: Proceedings of International Conference on Very Large Data Bases, pp. 487–499 (1994).
2. J.Han, J.Pei and Y.Yin." Mining frequent patterns without candidate generation," In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp.1–12 (2000).
3. J.S.Park, M.S.Chen and P.S.Yu. "An effective hash-based algorithm for mining association rules". In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 175–186 (1995).
4. S.Gupta and R.Mamtora. "A survey on association rule mining in market basket analysis" International Journal of Info. Computer Technology. 4(4), 409–414 (2014).
5. Q.Zhang and R.Segall. Web Mining: "A survey of current research, techniques and software". International Journal of Information Technology Decision Making 7(4), 683–720 (2008).
6. J.Liu, J.Shang, C.Wang, X.Ren and J.Han. "Mining quality phrases from massive text corpora". In: Proceedings of ACM SIGMOD International Conference on Management of Data, pp. 1729–1744 (2015).
7. S.L.Ting, C.C.Shum, S.K.Kwok, A.H.C.Tsang and W.B.Lee. "Data mining in biomedicine: current applications and further directions for research," Journal of Software Engineering Applications.150–159 (2009).
8. M.J.Zaki. "Scalable algorithms for association mining". IEEE Trans. Knowl. Data Eng. 12(3),372–390 (2000).
9. H. Mannila and H. Toivonen. "Levelwise search and borders of theories in knowledge discovery". In Data Mining and Knowledge Discovery, Vol. 1, 3(1997), pages 241-258.
10. P.Fournier-Viger, A.Gomariz, T.Gueniche, A.Soltani, C.Wu and V.S.Tseng. SPMF: a Java open-source pattern mining library. J. Mach. Learn. Res. 15, 3569–3573 (2014).
11. J.F.Boulicaut, A.Bykowski and C.Rigotti. "Approximation of frequency queries by means of free-sets". In: Zighed, D.A., Komorowski, J., Żytkow, J. (eds.) PKDD 2000. LNCS (LNAI), vol. 1910, pp. 75–85. Springer, Heidelberg (2000).
12. K.Gouda,M.Zaki. GenMax: "an efficient algorithm for mining maximal frequent itemsets". Data Mining. Knowl. Discovery. 11(3), 223–242 (2005).
13. C.Lucchese,S.Orlando,R.Perego. "Fast and memory efficient mining of frequent closed itemsets". IEEE Trans. Knowl. Data Eng. 18(1), 21–36 (2006).
14. N.Pasquier,Y.Bastide,R.Taouil and L.Lakhal. "Discovering frequent closed itemsets for association rules". In: Proceedings of 7th International Conference on Database Theory, pp. 398–416 (1999)
15. M. J. Md. Zubair Rahman, P. Balasubramanie, P. Venkata Krihsna. "A hash based mining algorithm for maximal frequent item sets using linear probing"2009.
16. D. Burdick, M. Calimlim, J. Gehrke, "MAFIA: A Maximal Frequent Itemset Algorithm for Transactional Databases," In Proc. ICDE 2001, pp. 443-452, 2001.
17. E. Boros, V. Gurvich, L. Khachiyan, and K. Makino, "On the Complexity of Generating Maximal Frequent and Minimal Infrequent Sets," STACS 2002, pp. 133-141, 2002.
18. D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, and T. Yiu, "MAFIA: A Performance Study of Mining Maximal Frequent Itemsets," In Proc. IEEE ICDM'03 Workshop FIMI'03, 2003. Available as CEUR Workshop Proc. series, Vol. 90, R.J. Bayardo, "Efficiently mining long patterns from databases", In SIGMOD, 1998.
19. V.S.Tseng, C.W.Wu, P.Fournier-Viger, P.S.Yu. "Efficient Algorithms for Mining the Concise and Lossless Representation of High Utility Itemsets," IEEE Transactions on Knowledge and Data Engineering, 27(3): 726-739 (2015).
20. C.W.Wu, P.Fournier-Viger, P.S.Yu and V.S.Tseng. "Efficient Mining of a Concise and Lossless Representation of High Utility Itemsets". Proceedings of IEEE International Conference on Data Mining, pp.824-833 (2011).

## AUTHORS PROFILE

**A. Subashini** completed her M.Phil degree in the year 2008 at Annamalai University, at present doing her PhD degree in Anamalai University. She has fourteen years of teaching experience. Presently she is working as Assistant Professor in the Department of Computer Application at Government Arts College, C.Mutlur, Chidambaram. Her research interests are Neural Networks and Data Mining.

**Dr. M. Karthikeyan** received the PhD degree from Annamalai University. Presently he is working as Assistant Professor in the Division of Computer & Information Science, Faculty of Science, Annamalai University. He published ten research papers in International journals and eight research papers in national journals. He has nineteen years of teaching experience and five years of research experience. His area of specialization includes Neural networks & Fuzzy systems, Data Mining and Digital Image processing.