ISSN No. 0976-5697

Volume 2, No. 5, Sept-Oct 2011

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

FRI Miner: A Tool for Frequent Regular Itemsets Mining

KHK Sairaj*
Dept. of Information Technology
UCEV - JNTUK, Vizianagaram
Andhra Pradesh, India
khksairajit@gmail.com

Dr MHM Krishna Prasad
Associate Professor & Head, Dept. of IT
UCEV - JNTUK, Vizianagaram
Andhra Pradesh, India
krishnaprasad.mhm@gmail.com

Abstract: Frequent itemsets plays a crucial role in many data mining tasks that aimed to find interesting patterns. Frequent Pattern mining is used for find frequent itemsets among items in a given data set and the exact frequency of each itemset. An objective of frequent pattern mining is to develop a systematic method for the given data set and find frequent items. Obtained frequent items can be used for enhancing the business return of interest. The Synoptic/Concise representation of frequent Itemsets sacrifice/presents readability and direct interpretability by a data analyst of the concise patterns extracted/picked up. This paper presents, *FRI Miner*, a tool for Frequent Regular Itemsets Mining, designed to evaluate RegularMine and FPGrowth, and experimented on real time datasets.

Keywords: Frequent Regular Itemsets, Association rules, FPGrowth, Data mining

I. INTRODUCTION

Frequent patterns mining is a fundamental and essential problem in many data mining applications. Frequent pattern mining has been focused by the data mining research community for over a decade and it plays an essential role in many data mining tasks that try to find interesting patterns from databases, such as association rules, correlations, sequences, episodes, classifiers, clusters and many more of which the mining of association rules is one of the most popular problems. Itemsets, an itemset is exactly, a set of items or a group of elements that represents together a single entity. An itemset is said frequent regular if it is frequent (as defined in [1] thanks to the support measure) and if it appears regularly (thanks to a measure of regularity/periodicity which considers the maximum compressed at which the pattern occurs). Itemsets, which are observed frequently, no less than a user-specified threshold, is called Frequent patterns itemsets, e.g., a set of items, such as computer and software, that appear frequently together in a transaction data set, is a frequent itemset [2]. A subsequence, such as buying first a PC, then a digital camera, and then a memory card, if it occurs frequently in a shopping history database, is a (frequent) sequential pattern. Lot of research observed in the literature aimed for developing efficient and scalable algorithms for frequent itemset mining in transaction databases to numerous research frontiers, such as sequential pattern mining, structured pattern mining, correlation mining, market basket analysis, web usage mining and their broad applications.

Frequent pattern mining was first proposed by [3], for market basket analysis in the form of association rule mining. It analyses customer buying habits by finding associations between the different items that customers place in their “shopping baskets”. For instance, if customers are buying milk, how likely are they going to also buy cereal (and what kind of cereal) on the same trip to the supermarket? Such information can lead to increased sales by helping retailers do selective marketing and arrange their shelf space. In literature, there are dozens of algorithms used to mine frequent itemsets. Some of them, very well known, started a whole new era in data mining. They made the concept of mining frequent itemsets and association rules possible. Others variations are mainly aimed to bring enhancement mainly in terms of processing time.

Author [4] proposed Itemset Code algorithm, which is used for finding the frequent itemsets (frequent page sets). The ItemsetCode algorithm is that it discovers the small frequent itemsets in a very quick way, thus the task of discovering the longer ones is enhanced as well. The algorithm SM-Tree (*State Machine-Tree*) [5] discovers the frequent page sequences and the algorithm PDTree (*Pushdown Automaton-Tree*) [6] discovers the tree-like patters. Both of the algorithms exploit the benefit of using automata theory approach for discovering the frequent patterns. The SM-Tree algorithm uses state machines for discovering the sequences, and the PD-Tree algorithm uses pushdown automatons for determining the support of the tree patterns in a tree database. As the two algorithms are built on automata, hence the two suffers with general constraints of automata.

APRIORI [7] is popularly used data mining technique for mining frequent itemsets and associations. Since its introduction, many scholars have improved and optimized the Apriori algorithm and have presented new Apriori-like algorithms. Apriori uses a breadth-first search strategy to count the support of itemsets and uses a candidate generation function which exploits the downward closure property of support.

CHARM [8] is another algorithm for mining all frequent closed itemsets. There are some innovative ideas developed by this algorithm that are of a certain value. CHARM simultaneously explores the itemset space and transaction space, rather than only the itermset search space. It also uses a highly efficient hybrid search method that skips many levels of the IT-tree (itemset-tidset tree) to quickly identify the frequent closed itemsets, instead of having to enumerate many possible subsets.

FP Growth[9][10] Frequent itemsets play an essential role in many data mining tasks that try to identify the frequent or interesting patterns from datasets, such as association rules, correlations, sequences, episodes, classifiers, clusters and many more of which the mining of association rules is one of the most popular problems. The Apriori algorithm has its own disadvantages that made other scholars think of new aproaches to mine frequent patterns. The two main downsides are the possible need of generating a huge number of candidates if the number of frequent one-itemsets is high or if the size of the frequent pattern is big the database has to be scanned repeatedly to match the candidates and determine the support. What if we find a way to mine the frequent patterns without candidate generation? This would be a big improvement over Apriori. This is what the frequent-pattern growth (FP-growth) algorithm does. It adopts a divide-and-conquer strategy and a frequent-pattern tree. The FP-growth algorithm is currently one of the fastest approaches to find frequent item sets. FP–Tree is a compact data structure. This FP-Grow algorithm compresses a large database into a compact, frequent–pattern–tree (FP Tree) structure. FP–Tree structure stores all necessary information about frequent itemsets of the dataset.

Hence, in this paper, author presents *Frequent Regular Itemset Miner* (*hereinafter*, FRI Miner), a tool designed for obtaining the Frequent Regular Itemsets and evaluating the performance of RegularMine technique with FPGrowth.

II. FRI MINER

FRI Miner is mainly focused in the generation of frequent regular itemsets. The significant/intended objective/purpose of concise representations is to alleviate the problems due to extracting, storing/accumulating and post processing a huge/massive amount of frequent patterns.

A. Frequent itemset:

Discovery of frequent itemsets [11] aims to find all itemsets occurring sufficiently many times in a given transaction database. An example of a transaction database is a collection of documents represented as sets of words occurring

$$\frac{\{a_1,\ldots\ldots,a_h\}}{a_1?,\ldots\ldots,a_h?}\text{N1} \quad \frac{\{a\}^+}{a}\text{N2} \quad \frac{a,a?}{a}\text{N3} \quad \frac{a,\{a,X\}^+}{a,X^*}\text{N4} \quad \frac{a?,\{a,X\}^+}{\{a,X\}^+}\text{N5}$$

(a) For the normal form of extended itesets.

$$\frac{R,X,Y^- \quad Y\cap X\neq\emptyset}{R,X,(Y\setminus X)^-}\text{S1} \quad \frac{R,X,Z^* \quad Z\cap X\neq\emptyset}{R,X,(Z\setminus X)^*}\text{S2} \quad \frac{R,\emptyset^-}{fail}\text{S3} \quad \frac{R,\{a\}^- \quad a\notin R}{R\setminus\{a?\}[\{a,X\}^-\rightarrow X^*]}\text{S4}$$

$$\frac{R,\{a,Y\}^- \quad a\notin R \quad Y\neq\emptyset}{R\setminus\{a?\}[\{a,X\}^-\rightarrow X^*],Y^* \quad R\setminus\{a?\}[\{a,X\}^-\rightarrow X^-],a,Y^-}\text{S5}$$

(b) For splitting non-compositional itemsets.

$$\frac{R \quad R,a}{R,a?}\text{M1} \quad \frac{R \quad R,Y^+}{R,Y^*}\text{M2} \quad \frac{R,b,a? \quad R,a}{R,\{a,b\}^+}\text{M3} \quad \frac{R,Y^+,a? \quad R,a}{R,\{a,Y\}^+}\text{M4}$$

$$\frac{R,Y^+ \quad R,a,Y^*}{R,\{a,Y\}^+}\text{M5} \quad \frac{R,Y^+ \quad R,Z^+,Y^*}{R,\{Z,Y\}^+}\text{M6}$$

(c) For merging extended itemsets.

Figure 1: Rewriting rules, R[X → Y] means that every extended item matching X in R is replaced by Y .

in them. There, an itemset is a set of words and the frequency of an itemset is the fraction of the documents containing all the words in the itemset.

Similarly, e.g., Let T be the transaction database and σ be the user specified minimum support. An itemset $X \subseteq A$ is said to be frequent with respect to σ, if $s(X)T \geq \sigma$.

B. FRI Miner:

The FRI Miner, designed for mining and obtaining frequent patterns to represent the frequent regular itemsets. The RegularMine Algorithm proposed by Salvatore Ruggieri[12] is adopted, for mining a set of regular itemsets that is a concise representation of frequent itemsets. The dedicated aspiration of concise representations is to alleviate the difficulty due to extracting, storing and post processing a huge amount of frequent patterns.

```
Algorithm 1: RegularMine
Input: a transactional database D
Output: a set $R_{out}$ of frequent regular itemsets
    Extract frequent closed itemsets CS from D and, for each
 C ∈ CS, the free sets in [C]
Rout ←Ø
for every C ∈ CS do
        Let $X_1, \ldots ,X_n$ be free sets in [C]
        $R=U_{i=1\ldots n}$ Covering $(X_i, X_1^-, \ldots , X_{i-1}^-,C^*)$
        R ← R ∪ Merging(R)
```

```
Algorithm 2: Merging
Input: a set of $R_{in}$ of pairwise disjoint extended itemsets
Output: a set $R_{out}$ of pairwise disjoint extended itemsets
equivalent to $R_{in}$, obtained by the merging rules M1-M6
R ← $R_{in}$
$R_{out}$ ← ∅
While R ≠ ∅ do
  k ← max{|R ∩ I| | R ∈ R}
  repeat
    let $R_1, R_2$ ∈ R such that|$R_2$ ∩ I | = k
and |$R_1$ ∩ I| ∈ [k –1, k]
    I ← $R_1 \cap R_2$
    if $R_1$ = I and $R_2$ = I, a then //rule M1
      R ← I, a?
    else if $R_1$ = I and $R_2$ = I, Y+ then //M2
      R ← I, Y ⋆
    else if $R_1$ = I, b, a? and $R_2$ = I, a then //M3
      R ← I, $\{a, b\}^+$
    else if $R_1$ = I, $Y^+$, a? and $R_2$ = I, a then //M4
      R ← I, $\{a, Y\}^+$
    else if $R_1$ = I, $Y^+$ and $R_2$ = I, a, Y⋆ then //M5
      R ← I, $\{a, Y\}^+$
    else if $R_1$ = I, $Y^+$ and $R_2$ = I, $Z^+$ , Y⋆ then //M6
      R ← I, $\{Z, Y\}^+$
   end if
   if any of the rules M1-M6 was applied then
     R = R \ {$R_1$, $R_2$} ∪ {R}
   end if
 until for every $R_1$,$R_2$ no rule M1-M6 applies
 K ← {R ∈ R | |R ∩ I| = k}
 R ← R \ K
 $R_{out}$ ← $R_{out}$ ∪ K
   end while
```

Starting from a frequent closed itemset and the free sets in its class of θ-equivalence, we obtained the frequent itemsets in the class in terms of pairwise disjoint non-compositional itemsets. The *Covering* and the *Merging* procedures we rewrite the non-compositional itemsets into equivalent pairwise disjoint regular itemsets. The overall procedure, called RegularMine, is presented as Algorithm 1.

The rewriting rules M1– M6 allow for *Merging* two or more extended itemsets into a single one is reported in Figure_1(c). For representation of the normal form of the extend itemsets by the rewriting rules N1-N5 is presented in Figure_1 (a).

The Covering procedure, the given non-compositional itemset R_{in}, computes a set R_{out} of extended itemsets equivalent to R_{in}. The approach follows the rewriting rules S1 − S5 reported in Figure_1 (b).

```
Algorithm 3: Covering
Input: a non-compositional itemset $R_{in}$ of the form $X$,
$Y_1^-, \ldots, Y_n^-, Z^*$
Output: a set $R_{out}$ of pairwise disjoint extended itemsets
equivalent to $R_{in}$, obtained by the splitting rules S1-S5
$R \leftarrow X, (Y_1 \setminus X)^-, \ldots, (Y_n \setminus X)^-, (Z \setminus X)^\star$ //rules S1, S2
$R_{out} \leftarrow \emptyset$
if $\forall i.Y_i \setminus X \neq \emptyset$ then //rule S3
  $\mathcal{R} \leftarrow \{R\}$
  while $\mathcal{R} \neq \emptyset$ do
    let $R$ be in $\mathcal{R}$
    $\mathcal{R} \leftarrow \mathcal{R} \setminus \{R\}$
    while $\exists Y^- \in R$ do
      let $Y^- = \{a_1, \ldots, a_k\}^-$ be in $R$
      $R' \leftarrow R \setminus \{a_1?\}$
      $R \leftarrow R'$ where every $\{a_1,X\}^-$ is replaced by $X^\star$
//rules S4, S5
          if $k > 1$ and $\{a_1\}^- \in R'$ then //rules S5, S3
$R'' \leftarrow R'$ where every $\{a_1,X\}^-$
is replaced by $X^-$
$\mathcal{R} := \mathcal{R} \cup \{(a_1,R'')\}$
end if
end while
$R_{out} \leftarrow R_{out} \cup \{R\}$
end while
end if
```

III. EXPERIMENTAL WORK

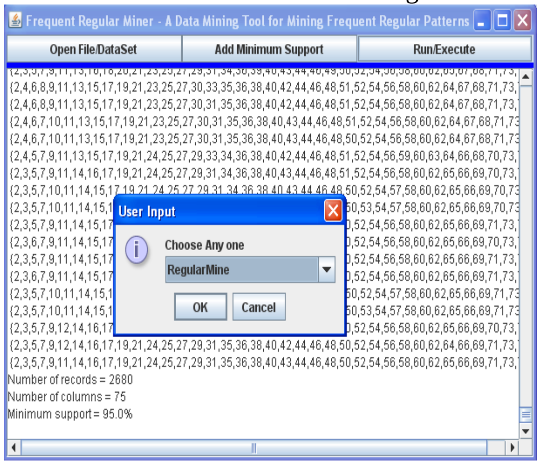
The experiments are performed using the Frequent Regular Itemsets Miner by adopting the RegularMine procedure on standard dense (mushroom, chess) datasets obtained from FIMI public repository [13].

The FRI Miner is implemented using Java (jdk 1.6) and all the experiments are performed on PC with Intel(R) Core (TM)2 Duo CPU with 2.93GHz processor and 1GB RAM running on the WINDOWS XP SP3 operating system.

In this paper, authors evaluated the RegularMine with FPGrowth using the FRI Miner and the obtained results are shown in Fig-4(a) and Fig-4(b). as per observations, the performance of RegularMine is well at high minimum support values and the FRI Miner is working efficiently even in complex cases with high minimum support, which is observed in Fig-4(a) and Fig-4(b).

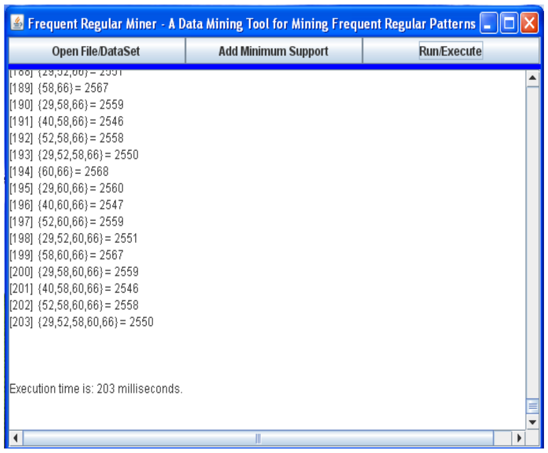
A. *FRI Miner Execution:*

The FRI miner can be experimented very easily as it is facilitated with UI controls, and having the option to select/modify the threshold, i.e. Minimum Support and opting the user to choose the one of the available algorithms.



```
{2,4,6,8,9,11,13,15,17,19,21,23,25,27,30,33,35,36,38,40,42,44,46,48,51,52,54,56,58,60,62,64,67,68,71,73,
{2,4,6,8,9,11,13,15,17,19,21,23,25,27,30,31,35,36,38,40,42,44,46,48,51,52,54,56,58,60,62,64,67,68,71,73,
{2,4,6,7,10,11,13,15,17,19,21,23,25,27,30,31,35,36,38,40,43,44,46,48,51,52,54,56,58,60,62,64,67,68,71,73
{2,4,6,7,10,11,13,15,17,19,21,23,25,27,30,31,35,36,38,40,43,44,46,48,50,52,54,56,58,60,62,64,67,68,71,73
{2,4,5,7,9,11,13,15,17,19,21,24,25,27,29,33,34,36,38,40,42,44,46,48,51,52,54,56,59,60,63,64,66,68,70,73,
{2,3,5,7,9,11,14,16,17,19,21,24,25,27,29,31,34,36,38,40,43,44,46,48,51,52,54,56,58,60,62,65,66,69,70,73,
{2,3,5,7,10,11,14,15,17,19,21,24,25,27,29,31,34,36,38,40,43,44,46,48,50,52,54,57,58,60,62,65,66,69,70,73
{2,3,5,7,10,11,14,15,1                                           50,53,54,57,58,60,62,65,66,69,70,73
{2,3,5,7,9,11,14,15,17                                           0,52,54,56,58,60,62,65,66,69,71,73,
{2,3,6,7,9,11,14,15,17                                           0,52,54,56,58,60,62,65,66,69,70,73,
{2,3,5,7,9,11,14,15,17                                           0,52,54,56,58,60,62,65,66,69,71,73,
{2,3,6,7,9,11,14,15,17                                           0,52,54,56,58,60,62,65,66,69,71,73,
{2,3,5,7,10,11,14,15,1                                           50,52,54,57,58,60,62,65,66,69,71,73
{2,3,5,7,10,11,14,15,1                                           50,53,54,57,58,60,62,65,66,69,71,73
{2,3,5,7,9,12,14,16,17                                           0,52,54,56,58,60,62,65,66,69,70,73,
{2,3,5,7,9,12,14,16,17,19,21,24,25,27,29,31,35,36,38,40,42,44,46,48,50,52,54,56,58,60,62,64,66,69,71,73,
{2,3,5,7,9,11,14,16,17,19,21,24,25,27,29,31,35,36,38,40,43,44,46,48,50,52,54,56,58,60,62,65,66,69,71,73,
Number of records = 2680
Number of columns = 75
Minimum support = 95.0%
```

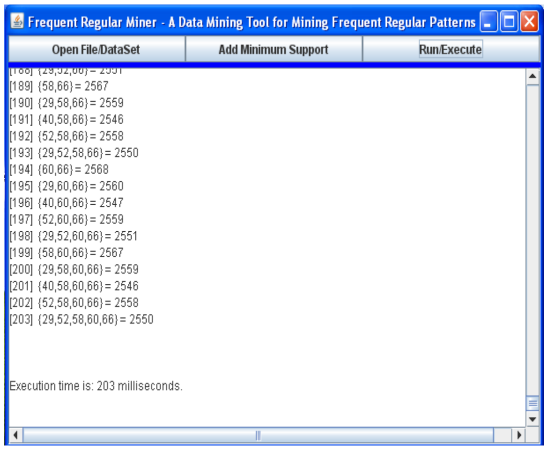
Figure 2(a) choosing the RegularMine


Frequent Regular Miner - A Data Mining Tool for Mining Frequent Regular Patterns
Open File/DataSet
Add Minimum Support
Run/Execute

```
[189]  {58,66}= 2567
[190]  {29,58,66}= 2559
[191]  {40,58,66}= 2546
[192]  {52,58,66}= 2558
[193]  {29,52,58,66}= 2550
[194]  {60,66}= 2568
[195]  {29,60,66}= 2560
[196]  {40,60,66}= 2547
[197]  {52,60,66}= 2559
[198]  {29,52,60,66}= 2551
[199]  {58,60,66}= 2567
[200]  {29,58,60,66}= 2559
[201]  {40,58,60,66}= 2546
[202]  {52,58,60,66}= 2558
[203]  {29,52,58,60,66}= 2550

Execution time is: 203 milliseconds.
```

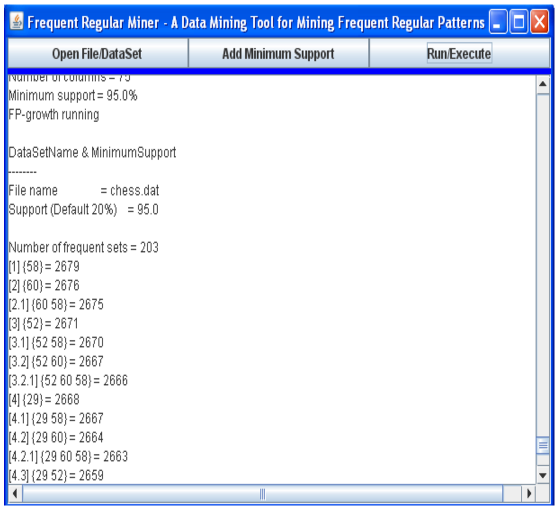
Figure 2(b) Obtained output from the RegularMine


Frequent Regular Miner - A Data Mining Tool for Mining Frequent Regular Patterns
Open File/DataSet
Add Minimum Support
Run/Execute

```
[189]  {58,66}= 2567
[190]  {29,58,66}= 2559
[191]  {40,58,66}= 2546
[192]  {52,58,66}= 2558
[193]  {29,52,58,66}= 2550
[194]  {60,66}= 2568
[195]  {29,60,66}= 2560
[196]  {40,60,66}= 2547
[197]  {52,60,66}= 2559
[198]  {29,52,60,66}= 2551
[199]  {58,60,66}= 2567
[200]  {29,58,60,66}= 2559
[201]  {40,58,60,66}= 2546
[202]  {52,58,60,66}= 2558
[203]  {29,52,58,60,66}= 2550

Execution time is: 203 milliseconds.
```

Figure 3(a) choosing the FP Growth


Frequent Regular Miner - A Data Mining Tool for Mining Frequent Regular Patterns
Open File/DataSet
Add Minimum Support
Run/Execute

```
Number of columns = 75
Minimum support = 95.0%
FP-growth running

DataSetName & MinimumSupport
--------
File name            = chess.dat
Support (Default 20%)    = 95.0

Number of frequent sets = 203
[1] {58} = 2679
[2] {60} = 2676
[2.1] {60 58} = 2675
[3] {52} = 2671
[3.1] {52 58} = 2670
[3.2] {52 60} = 2667
[3.2.1] {52 60 58} = 2666
[4] {29} = 2668
[4.1] {29 58} = 2667
[4.2] {29 60} = 2664
[4.2.1] {29 60 58} = 2663
[4.3] {29 52} = 2659
```

Figure 3(b) obtained output from the FP Growth

B. *Execution Time:*

The experimental results, which are obtained from the FRI Miner is shown in the Fig-4(a) and Fig-4(b). Following figures shows the obtained performance results with FP Growth algorithm for mushroom and chess datasets.

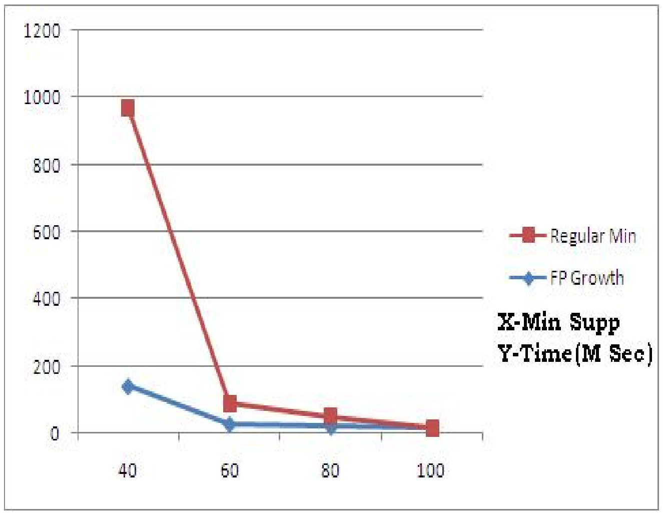


Figure. 4(a) Execution time for mining frequent regular itemsets on mushroom dataset

Fig. 4(a) show the obtained results for evaluating the FRI Miner with FP Growth performed on mushroom dataset. As shown in fig-4(a), the execution time of *FP Growth with the FRI Miner is performing well at higher minim support*.

Fig. 4(b) show that the experimental work on the FRI Miner with FP Growth on chess data set. The execution time is almost similar in many cases on this dataset *and observed considerable performance at higher minimum support cases.*

IV. CONCLUSIONS

In our everyday working life, the patterns are that are easy to understand, to manipulate and to reason about the data analyst, not necessarily data mining experts, so it is of our primary importance for a general acceptance of the knowledge discovery methods. Hence, Frequent Regular Itemset Miner (FRI Miner), A Tool for Frequent Regular Itemsets Mining, running with the RegularMine for identifying frequent regular itemsets proposed. Frequent Regular Itemset Miner, also designed to evaluate the RegularMine technique with FPGrowth and experimented on the real time datasets. From the experimental observation, it is proved that the Frequent Regular Itemsets miner is efficient for finding the Frequent Regular Itemsets.

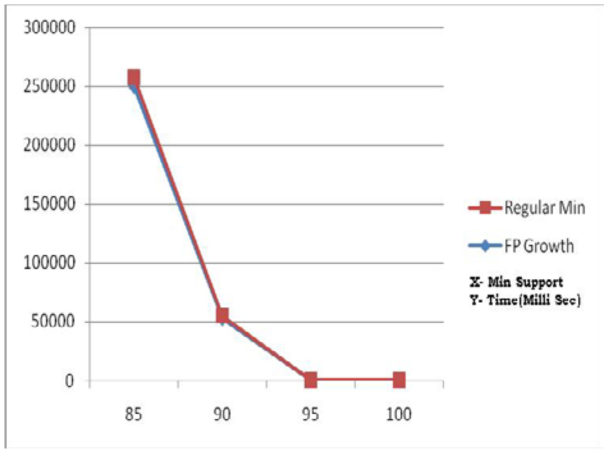


Figure. 4(b) Execution time for mining frequent regular itemsets on chess dataset

There are still many interesting research issues related to the extensions of frequent regular itemsets mining, such as mining structured patterns by further development of these approaches, mining approximate or fault-tolerant patterns in noisy environments, frequent-pattern-based clustering and classification, and so on.

V. REFERENCES

[1] K. Amphawan, P. Lenca, and A. Surarerks, "Efficient mining Top-k regular-frequent itemset using compressed tidsets", PAKDD, Shenzhen, China, pp. 159-170, 2011

[2] J. Han, H. Cheng, D. Xin, X. Yan, Frequent Pattern Mining: Current Status and Future Directions, Data Mining and Knowledge Discovery, Vol. 14, No. 1. (2007)

[3] R. Agrawal, T Imielinski, A Swami, Mining association rules between sets of items in large databases, In: Proc. of the 1993ACM-SIGMODinternational conference on management of data, Washington, DC, pp 207–216

[4] R. Ivancsy, I. Vajk, "Frequent Pattern Mining in Web Log Data", Journal of Acta Polytechnica Hungarica Vol. 3, No. 1, 2006

[5] I. Renata and V.Istvan, "Efficient sequential pattern mining algorithms", In: Proc. of the 4th WSEAS International Conference on Artificial Intelligence, Knowledge Engineering Data Bases, pp. 33:1-33:6, 2005

[6] R. Iváncsy and I. Vajk, "PD-Tree: A New Approach to Subtree Discovery.", WSEAS Transactions on Information Science and Applications, Vol. 2, Num. 11, 2005, pp. 1772-1779

[7] R. Agrawal and R. Srikant, Fast Algorithms for Mining Association Rules, VLDB'94, International Conference on Very Large Data Bases pp. 487 – 499, 1994

[8] Mohammed Javeed Zaki, Ching-Jiu Hsiao, CHARM: An Efficient Algorithm for Closed Itemset Mining, SDM – SIAM - ICDM, Chicago, 2002

[9] C. Borgelt, An Implementation of the FPgrowth Algorithm, School of Computer Science, Ottovon Guericke University of Magdeburg Universitätsplatz 2, 39106 Magdeburg, Germany

[10] J. Han, J. Pei, Y. Yin: Mining Frequent Patterns without Candidate Generation. SIGMOD Conference 2000: pp. 1-12

[11] T. Mielikäinen, P. Panov, S. Džeroski, Itemset Support Queries using Frequent Itemsets and Their Condensed Representations, HIIT BRU, University of Helsinki

[12] S.Ruggieri, Frequent Regular Itemset Mining, KDD 2010, pp. 263-27

[13] B. Goethals. Frequent Itemset Mining Implementations Repository.http://fimi.ua.ac.be