Association Rule Mining Technique with Optimized FP−Tree Algorithm

Ling Yuan, Wasan Itwee, and Shikang Wei

***Abstract*—One of the well-known techniques FP (Frequent Pattern) technology is considered to be extensible and most professional. The FP has candidate set problems and results in an enormous number of candidate sets being generated, which means that the DB is scanned repeatedly and results in a large candidate set being examined. In association rule mining algorithms the FP-Tree algorithm is one of the most favored, especially for mining long patterns and confirming a huge set of candidates through pattern matching. This tree solves the problem of generating candidate sets and needing to scan DB repeatedly. In our implementation and from a lot of experiments, we improved and simplified the old algorithm by using a new method.**

In the optimized FP-Tree, we extract FIs without generating the conditional FP-Tree. At the same time, we can see the frequency of each item from frequency head table, and know the Root of each item from new tree, additionally reduce the branch of the tree, where each root has the real son without repeat. The process that we follow reduces the gross strides of the execution, and as well takes minimal time and size. Of course, efficient than the FP-Tree depend on procedures, we decrease the cost of time and storage space, especially in large DB and mining long patterns.

***Index Terms*—FP−Tree, Optimized FP−Tree, frequent pattern, frequent item set, Association rules.**

I. INTRODUCTION

DB has been used in many areas, such as science, government administration, business operations and a lot of other main applications. And perhaps new information extracted from the DB can help a lot of organizations get good information, useful applications, query processing and help decision-making. We can see the unstable enlargement of data in all areas, individual of the main challenges for mining community and data organization is mining information, the FI mining is induced by difficulty like market basket analysis [1]. We can consider the set of items covered by the customer as a tuple in a market basket DB and denote the items in transaction T. If a few items are taking up in transaction T, it means that some other items may be purchased, and this example of association rules mining from market basket DB status discovery rules are important for lead upcoming sales uplift and supplies up rates [2].

The main goal of association rule mining is to locate interesting links, FP, and incidental structures within objects in different data repositories or TDB transactional databases, Figure 1 explains the purpose of association rules [5].

Mining association rules are divided into two stages:

1) Determine the big item sets, including the sets of things that have T with minimum support (Min_Sup.). We should note the support over a fixed minimum threshold, identified as FIs.
2) Create the association rules for the DB that have minimum confidence (Min_Conf.). We should note the confidence over a fixed minimum threshold.

Fig. 1. Aims of associations rules

In our paper, we proceed as follows in section 1: Introduction. In section 2: Related work. In section 3: Talking about problem define, frequent item set mining, and we also talk about why there are many algorithms for association rules. In section 4: Describe the first algorithms FP-Tree and disadvantages of FP-Tree. In section 5 optimized FP-Tree, describing the algorithm of optimized FP-Tree and doing the experiments, and show the differences between the FP-Tree and optimized FP-Tree. In section 6: Experiment result with performance evaluation. In section 7: Conclusion.

II. RELATED WORK

Initially, R. Agrawal, A.Swami and T. Imielinski introduced the Association rule mining, since long time ago the algorithms concerning ARM began to execute relational data, a lot of algorithms provided such as Apriori, DHP, Tree projection and FP-Tree. One of old and important the Apriori algorithm, Which is used to locate the large item set in DB uses to approach a bottom-up and breadth-first advance, this algorithm cannot be used directly to extract difficult or complex data, an additional well-known algorithm the FP-Tree algorithm in the FP-Tree algorithm relies on the divide-and-conquer approach. First, ranking and describing the FIs in a tree called FP-Tree. After building this tree, it can also be used to compress the DB, which means that we can apply the ARM is performed on the pressed DB and to assist of this end tree. In addition, the algorithm FP does not require candidates to be created, unlike other algorithms.

As a result, in the relationship with the Apriori algorithm, but from a lot of studies, it is similar to other algorithms, but one of these flaws of the algorithm is that it generates a big

numeral of conditional FP-Trees as a process of mining, therefore, activity of the Tree algorithm is not feasible [3][4].

III. PROBLEM DEFINE

A. *Frequent Item Set Mining*

We can explain the basic concepts of Frequent Item Set Mining bellow: If we have database represented as X and item in this DB I, then: Let X = {i1. . . in} as a set of items. We say it is the basis of the item in the DB.

These Items may be any element, such as products, food special equipment items, etc.

- In the smallest subset, I ⊆ X is called the item set. An item set may be any set of products that can be bought (simultaneously).
- Let T = (r1. . . rn) with ∀k, 1 ⩽ k ⩽ n : rk ⊆ X be a tuple of T above X. This tuple is called the TDB, and a TDB can be a table, if we collect the products purchased by the customer at a certain time in the supermarket or shopping, we can consider each item set as T in DB, T can also be defined as a pocket or multi set of transactions [2].

B. *Frequent Item Set Mining: Formal Definition*

If we have DB given

- A set X = {i1. . . im} of items, the item basis.
- A tuple T = (t1. . . tn) of transactions on X, which is also called TDB, a number min sup. Is 0 < min sup. ⩽ n, or (equivalently) TDB, we can explain in the example: If we have database containing 5 (1,2,3,4,5) elements with 10 transactions, these transactions are:

{1, 4, 5},{2, 3, 4} ,{1, 3, 5} ,{1, 3, 4, 5},{1, 5},{1, 3, 4}, {2, 3} ,{1, 3, 4, 5},{2, 3, 5}, {1, 4, 5}

Then the Frequent Item Set for this TDB is:

When the one items are :({1}: 7, {2}: 3, {3}: 7, {4}: 6, {5}: 7)

When the two items are :({1, 3}:4, {1, 5}:6, {1, 4}:5, {2, 3}:3)

When the three items are: ({1, 3, 4}: 3, {1, 3, 5}:3, {1, 4, 5}:4)

When the four items are: ({13, 4, and 5}:2)

The FIs for TDB there are 25 = 32 possible item sets over X = {1, 2, 3, 4, 5}.

C. *Defined the association rules*

The ARM is found from the DB to introduce the necessary FP mining to determine interesting association rules and correlations for each T in DB and relational DB. For the association rule problem, there is a formal statement [5] as follows : L= {L1, L2, L3,L4,…, Lm} as an element collection called as items, such as laptop , video, printer, CD , Monitor , …and so on. Now let the DB become a collection of TDB, T⊆L. Each T is associated with the single identifier, called a transaction identifier (TID). It is recommended that if we have asset of items as X, Y the structure X → Y and X ∩ Y = Ø • X: antecedent • Y: consequent of the rule. Regardless of whether X, Y is a collection of items or a pattern in association rules, one of the two important measures is called support, and the other is called confidence, which we can define as follows:

- Support: The rule (X ⇒ Y) holds in T set DB with sup. Where sup. is the percentage of T in DB, and should contain X U Y. Now we can define the formulae for Support as

  Support (X ⇒ Y) = X ∪ Y / DB
- Confidence: The rule (X⇒ Y) has conf. in T set DB, where Conf. is the percentage of T in DB that contains X, and also contains Y. Now we can define the formulae for Confidence as

  Confidence (X⇒ Y) =P (Y|X) and that mean

  =Support(X ∪ Y) / Support (X)

D. *Algorithms for association rules*

There are many algorithms in association rules, because a big numeral of rules will be pruned after application of the Min_Sup. and Min_Conf. thresholds, so the preceding computations will be wasted, and big problems will be taken into account, better execution of the rules will be achieved and the problem discovery algorithm avoided. So as to search the FIs, the Min_Sup. of every item set must be calculated through examining each T in the dataset. In order to do this, there is a wanton force approach to limit the exponential numeral of Min_Sup.'s item sets. Counting would be computationally costly. There are many advanced algorithms for extracting FIs from the large DB, and there is a close correlation between the efficiency of the algorithm and the range of the DB that will be processed [5].

There are two types of strategy in algorithms as followed:

- Apriori techniques: to minimize the combinatorial consideration area of candidate itemsets regarding an effective pruning organization.
- FP-Tree techniques: it is used to represent the pressed data and to make it easier to handle the item sets.

IV. THE ANALYSIS OF FP-TREE ALGORITHM

The general idea of this tree is that we have two points. First, a divide-and-conquer style: break down mining responsibilities into ones, and secondly keep away from candidate generation: sub-DB examination only. After the FP-Tree is generated, the Conditional Pattern (CP) Trees are generated for each element in the FP-Tree to obtain the FIs. The reason behind the choice of this algorithm is that it is very reduced, and is certainly the least than the central DB. The mining method in this tree will decrease the cost of DB scanning

- **Why FP-Tree?**

- No candidates to generate and no candidates to test.
- Reduce duplication of DB scans and focus on compact data organization.
- Essential act includes and FP-Tree arrangement.

We can describe FP-Tree vs. Apriori, illustrating that a comparison between the FP-Tree and the Apriori in the case of execution time is a small support and a large support case, and in memory and in support of small cases, large scalability with the support threshold[15][9]. From many kinds of research, the runtime of FP-Tree depended on all reports, including the time of constructing FP-Tree from the original DB. In the experiments, the experimental datasets were synthesized and the experimental results were obtained on two synthetic datasets. The synthetic DB set used in the experiment was generated using Agrawal and Srikant (1994). FP-Tree vs. the original DB, Tree Projection, and Apriori illustrate the execution time comparison between the FP−Tree and others, in the case small support and a large support case. There are many experiences that combining the three algorithms(FP-Tree, Tree Projection, and Apriori)together to compare the runtime and the result, all of which have proven efficient FP-Tree[6].

We show the example of the work of the algorithm FP-Tree Consider the following transactions and the minimum support as 3, Table 1 example about DB and the Tree of these algorithms in Figure 2 and Table 2.

TABLE I: TRANSACTIONS DATABASE

TDB
PC , Hard ,Laptop , DVD ,
PC , Hard , CD, Headphone
PC , Hard ,Laptop , DVD , Keyboard
PC , Hard , Laptop , CD , Mouse
PC , Hard , Laptop , CD , DVD, Flash
PC , CD , DVD

TABLE II: CONDITIONAL PATTERN VASES

Item	cond. pattern base
PC	Null
Hard	{ (PC : 5) }
Laptop	{ (PC Hard : 4) }
CD	{(PC Hard :1) , (PC : 1) , (PC Hard Laptop : 2) }
DVD	{ (PC Hard Laptop : 2) , (PC Hard Laptop CD :1) ,(PC CD : 1)}

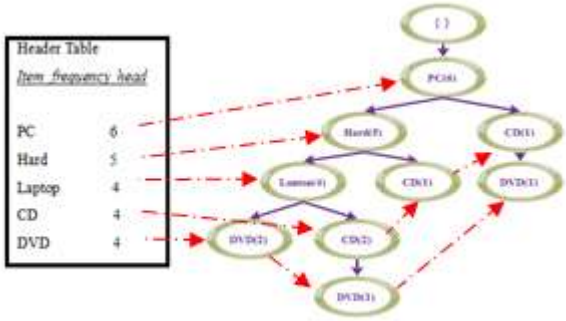

Fig. 2. Constructing the FP-Tree iteratively

A. *Disadvantages of FP-Tree*

Although the algorithm is used, it has disadvantages [3] of any other algorithm. One disadvantage of this algorithm is that a large number of conditional FP trees are generated repeatedly as a mining. The construction of the algorithm is considered expensive, for three reasons:

-Trade-off: this means that it takes time to construct.

-Also time is wasted: (that will be higher if doing Min_Sup.).

-Support for calculating once, the entire TDB is added to the FP-Tree. In FP-Tree, we generate the conditional FP-Tree to extract FIs, in addition to the branches of the tree together, each root has many sons with a lot of nodes repeated. By this way, we lose the costly in the time and storage space, especially for large DB (data base) and mining long patterns. FP-Tree is difficult to use in an interactive mining organization.

V. OPTIMIZED FREQUENT PATTERN TREE

In an optimized FP-Tree, we propose a new tree and a new algorithm for mining association rules, which mines every probable FI without generating new node and table of conditional frequent pattern trees. It also provides the frequency of FI for evaluation of the preferred ARM [7].

Mining FIs used for the ARM starting the large TDB is an essential task. There are a lot of approaches and algorithms have been used in this area, almost all of the early studies use FP-Tree to extract the FIs, which have the development scope. In our paper selection, we pay attention to FP-Tree and how to optimize the performance of this tree, as a result, the aim of this research is to discover a method to call the rules in the transactional data sets by taking into account the time and the memory exhaustion of the optimized FP-Tree.

The main idea is to suggest an improvised off the FP-Tree with a modified frequency table. This frequent pattern tree is beyond condensed than the traditional frequent pattern Tree. This reduces the expensive multiple data scans. In addition, for each item table to use the modified frequency, we are fit to arrival and check the presence of any node immediately in the tree, rather than across the entire tree.

In other words, in Optimized FP-Tree, we extract FIs that do not have the FP-Tree, we can see the frequency for each item from frequency head table, and know the root of each item from new tree, additionally reducing the branch of the tree, each root has the real son without repetition. By this way, we decrease the cost of time and storage space, especially for big databases and mining long patterns.

The process includes: FP-Tree construction and frequency of each item table generation, there are only two cases when an item is not added to tree:

- When the frequent item is not current in the transaction, all transactions in that transaction are canceled.
- When there is no straight link between the item in the reflection and the present root, and the item also exists in the FP-Tree, then it is canceled.

A. The method for optimized FP-Tree

In the second algorithm, the optimized FP-Tree scan the transaction database TDB once. Collect F, the set of frequent items, find the support of each frequent item in sets, and test if the item in TDB are less than supported, then delete it from TDB and table frequent item after delete all items less Min_Sup. arrange TDB, and table frequent items beginner the big frequent until small frequent.

In the second algorithm, optimized FP-Tree works as the style of the division each T into two parts A and B, which should be treated as A is the father and the most commonly used, and must be affixed on all transfers, part B it is also divided into two parts on august second, the rest is considered to be sons or branches of the second father, here is considered as follows:

- If the father is among the most common element after the main course, the father is credited to the tree and destroyed by one, but for the group of new branch is compounded if the tree or never added.
- If the father is not among the most common compounded, his sons all do not increase the frequency to him, taking into account the children of new branches here, and add the same frequency of existing value, rather than create a new node, but the work of a new path.

B. Algorithm for optimized FP-Tree

- **Input:** TDB and Min_Sup. Threshold
- **Output:** Optimized FP-Tree, Frequency of each item
- **Step1**
  - **[a]:** Scan TDB once. Collect F, the set of frequent items, find the support of each FI
  - **[b]:** For each transaction in DB
    If the item in TDB less than support
    Then delete it from TDB and from table frequent item
  - **[c]:**Arrange TDB and table frequent items beginner the large frequent until small frequent
- **Step 2:** Create R= NULL.
- **Step 3:** For each T in TDB go to step 4
- **Step4:** Should a descending ordered T by [A|B]. A is the first item and B is have a rest of the items in each T in DB.
- **Step5:** Now cheeked If A= the a large amount of FI, go to step6 and step7.Otherwise go to step 8.
- **Step6:** If R has a direct new node as child
  - **[a]:** Move the root from R to A.
  - **[b]:** Enlarge the F of the item A in tree by 1.
- **Step7:** For each item reset in second part B:
  - **[a]:** For each new item generate a new node from old root. And add to the frequency of item by 1.
  - **[b]:** If B has no new node from the existing root to its node, Where Bi is an item in B and already exists in FP-Tree then Bi is a old node and not add in tree .
  - **[c]:** If B new root but no new node then add only Bi to tree.
  - **[d]:** If B not new root then Increase the frequency B1 in tree by 1.
- **Step8:** don't add this T

C. Implementation of new algorithm with example

In our implementation we take the same database in Table1, and calculate the Support for each item, it is shown in Table 3.

TABLE III: ITEMS WITH SUPPORT

Candidates	support
PC	6
Hard	5
Laptop	4
CD	4
DVD	4
Headphone	1
Keyboard	1
Mouse	1
Flash	1

Transaction 1: PC Hard Laptop DVD

The FIs from the "table1 frequent items" is {PC, Hard, Laptop, DVD}. The big root of the optimized FP-Tree is produced and it is "NULL". As "PC " is the straight node of the R, F is greater than before by 1, depending on the algorithm (explains in step 6), new items are reset on different items or nodes, and the F of FP-Tree appears, such as {PC: 1}. And then complete another child {Hard :1 } , { Laptop :1} and {DVD:1}, it explains in Figure 3 transaction a.

Transaction 2: PC Hard CD

The FIs from the "table 1 frequent items" is {PC ,Hard ,CD}, we can notes the items "PC" and "Hard" are in the similar branches of the tree, but the item "CD" is not the same branch, and the "CD" is a new node, that means the root is shifting, the new root become is item "Hard ". Note that there are no isolated edges starting from "Hard" to "CD", the item is inserted from root PC. As is qualified (explained in step 7 of algorithm 2), in this deal, a new item appears "CD". Depending on (explained in step 6), the old root {PC: 1} is altered and becomes {PC: 2}. A fresh node will be generated from the root "PC". Adding Hard: 1 to tree that represents the association between two items Optimized. Here items "PC" and "Hard" is previously existed and "CD" is regarded as a new node of all this transaction in tree b in Figure 3.

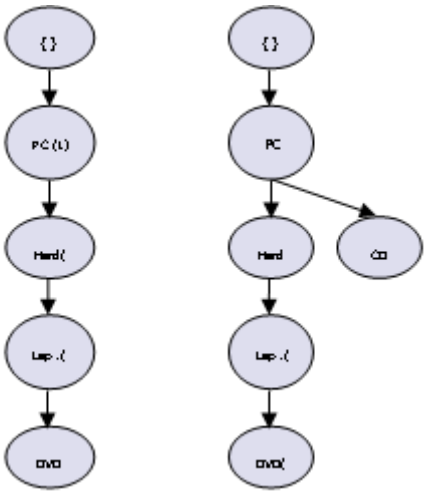

Fig. 3. Build transaction a and transaction b

Transaction 3: PC Hard Laptop DVD

The FIs from the"table1 frequent items" is {PC, Hard, Laptop, DVD}. "PC" is the straight node of the original root, frequency is greater than before by 1(explained in step 6). And then complete the other child {Hard : 3 }, { Laptop :1} and { DVD:1}, this optimized afford the additional fixed correlations along items, the association rule generations will be flawless. As 5 and 8 steps in algorithm 2, it explains in Figure 4.

Transaction 4: PC Hard Laptop CD

The FIs from the "table1 frequent items" is {PC, Hard, Laptop, CD}. "PC" for the same reason above, then its frequency is greater than before by 1(explained in step 6). And then complete the other child {Hard : 4 }, { Laptop :1} and { CD:1}, as 5 and 8 steps in algorithm2, it explain in Figure 4.

Transaction 5: PC Hard Laptop CD DVD

The FIs from the"table1 frequent items" is {PC, Hard, Laptop, CD, DVD}. "PC" is greater than before by 1. Here find a new node "DVD" from the node "CD" then add this line.

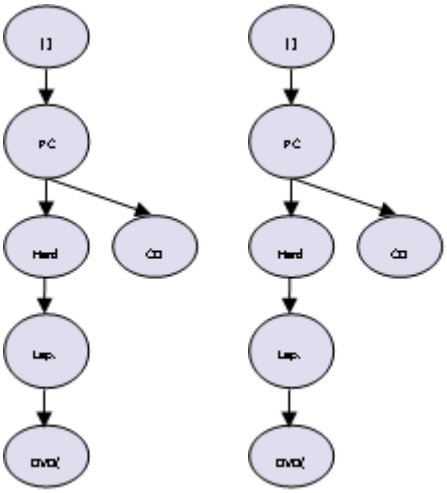

Fig. 4. Build transaction c and transaction d

And then complete the other child {Hard : 5}, { Laptop :1} and { CD:1}, as 5 and 8 steps in algorithm 2, it explain in Figure 5 .

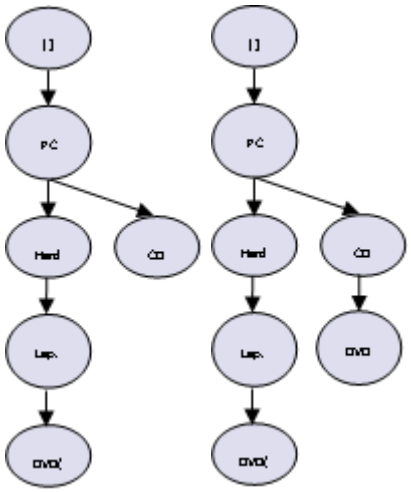

Fig. 5. Build transaction e and transaction f

Transaction 6: PC CD DVD

The FIs from the "table1 frequent items" is {PC, CD, DVD}. "PC" is greater than before by 1. And then complete the other child {CD: 1} and {DVD: 1}, as 5 and 8 steps in algorithm 2, it explain in Figure 5.

As we have noted in chapter three, the final tree is created to see all the important items in the database and to find out links between those elements of a new node for each element, and the main goal is to extract schedule condition of the tree (conditional FP-tree), which shows the shortest path from the tree for each element, according to the frequency sequence previously in table FIs and dependency of the line item is the last form of improved tree built-up optimized, you can see small tree branches where they delete the excess branches, according to the terms of the proposed algorithm, taking into account that extracting schedule requirement directly from the tree without the need to build the table and to search for all the branches, and to find short way which has found the frequency of each element. In advance, this saves time in processing and storage capacity in the memory. Note that in the example we use the small database with 5 items without any other explanation of the difference between the two algorithms, In next chapter, we will use the big database because the reason for using the optimized is very useful, when we use the big database that means the parting become the biggest.

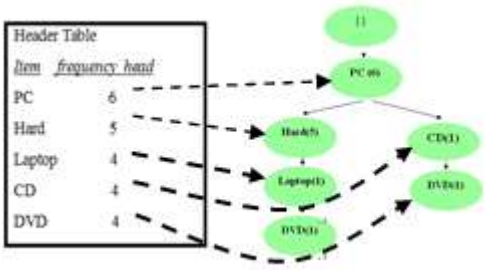

Fig. 6. Optimized FP-Tree

VI. EXPERIMENT RESULT WITH PERFORMANCE EVALUATION

There are many experiments performed on the old algorithm FP-Tree, Compared to our algorithm optimized

FP-Tree, we have created a new database with new items equals the number of transactions and items in the old experiments.

The first experimental analysis of a data set between the old algorithm FP-Tree containing 5665T and 12 items and the proposed algorithm optimized FP-Tree. The old algorithms is implemented using Net Beans IDE 7.4. The data set used in this analysis was obtained from machine learning repository UCI [6] [8], and the implementation of our algorithm was obtained in our program in java language. We assume that the new database with 5665 transactions and 12 items and the Min_Sup. taken is 0.15. The proposed optimized FP-Tree algorithm consistently takes less time to find the association rules than the FP-Tree algorithm. We assume that each of the 10 reads is different from the items in the permutation database, and that these readings are performance different between the two algorithms. We found in our experiments that the total time it takes in first reading to build the new tree = 42ms, it is explained in Figure 7.

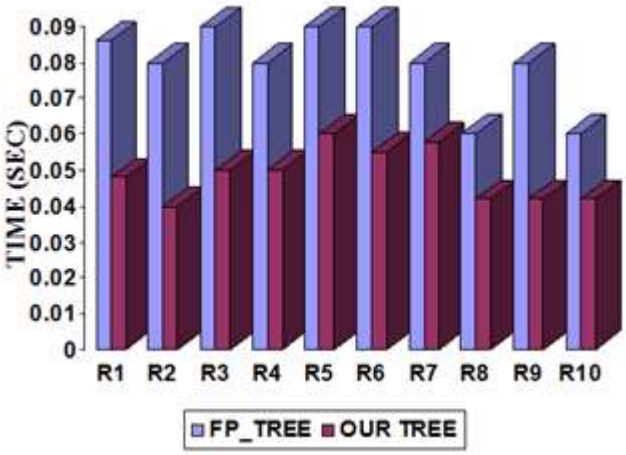

Fig. 7. Experimental analysis with 5665 transactions and 12 items

After that, we have conducted experiments and found a difference between the database memory storage, in the case of the use of data consisting of 10 T, calculating the area size old and new tree base, and found a difference is evident in the area. We have to experiment with different data consisting of 10 T until 10000 T follows the map supposed rules ranging from 5 items to 40 items, and assumes that the final experiment of database has 10000 T with 40 items and the end optimized FP-Tree is shown in Figure 8.

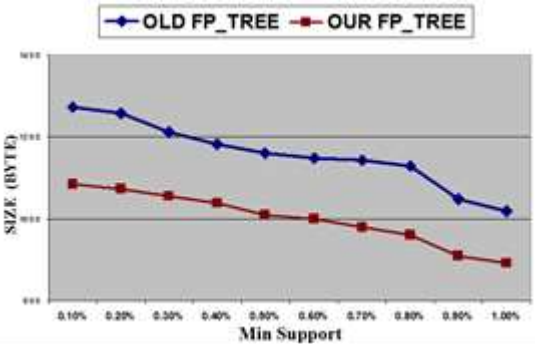

Fig. 8. Experimental analysis DB with 10000T and 40 items with minsup.

After this experiment, we found the difference between the two trees and the size measured by unit byte, taking into account that transfers between elements of type and amount of support is equal in each experiment, and that between the two trees, the difference is returned to the fact that the old tree adds all belonging to the movement, which generates a large tree element Lama proposed the tree, there are conditions attached the join elements to the algorithm, which reduces the added elements of the tree, and this in turn reduces the space used in memory for this, we measure the difference between old one and our tree. And our tree relies on the value of Min_Sup., the experimental analysis DB with 10000T and 40 items has different value of Min_Sup. From 0.10% up to 1%, as shown in Figure 9.

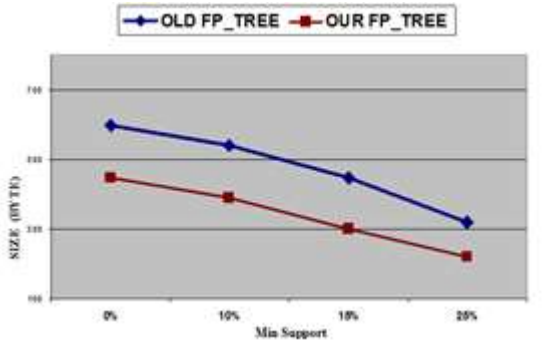

Fig. 9. Experimental analysis DB with 33990T and 15 items with minsup.

VII. CONCLUSION

In our paper, we choose the FP-Tree because this algorithm loes not generate the candidate set. However, candidate set ;eneration is still costly, which reduces together time and ;pace complexities extremely. We have improved the)erformance of the old algorithm. The new algorithm creates ı tree containing a single node for each element in general ·xcept for some relationships that contain more than one node, and as a result, data can be very intense represented. Our algorithm is released from the generation of conditional FP-Tree. However, conditional generation is still costly, essentially when there a huge number of patterns and long patterns are worked, numerous recursive calls are in the implementation which reduces together time and space complexities extremely, and contains a single node tree that avoids repeating many calls In the implementation. Avoiding repetition in reduced time and space, especially in large databases that contain many transactions with different relationships between items. Our process is, of course, more efficient than the current FP-Tree based on actions and it reduces the gross steps of the procedure.

REFERENCES

[1] COMPUTER SCIENCE AND ENGINEERING DEPARTMENT THAPAR UNIVERSITY A BETTER APPROACH TO MINE FREQUENT ITEMSETS USING APRIORI AND FP-TREE APPROACH, PATIALA – 147004, June, 2011.

[2] He Jiang, Xiumei Luan, Xiangjun Dong, “Mining Weighted Negative Association Rules from Infrequent Itemsets Based on Multiple Supports”, IEEE, International Conference on Industrial Control and Electronics Engineering, 2012.

[3] Raorane A.A. Kulkarni R.V. and Jitkar B.D., "Association Rule – Extracting Knowledge Using Market Basket Analysis", Research Journal of Recent Sciences, pp. 19-27,Vol. 1(2), Feb, 2012.
[4] International Journal of Modern Engineering Research (IJMER) Effective Positive Negative Association Rule Mining Using Improved Frequent Pattern, www.ijmer.com Vol.3, Issue.2, March-April, 2013.
[5] Scalable Frequent Pattern Mining using Relational Databases Utrecht University Faculty of Science Department of Information and Computing Sciences, August, 2014.
[6] Dongxu. FP Growth With C Implementation. A datamining algorithm [Online]. Available: https://github.com/wangdx/FP-Growth (accessed 16th October 2014).
[7] Vandit Agarwal1, Mandhani Kushal2 and Preetham Kumar3 AN IMPROVISED FREQUENT PATTERN TREE BASED ASSOCIATION RULE MINING ,by International Journal of Data Mining & Knowledge Management Process (IJDKP) Vol.5, No.2, March, 2015.
[8] Efficient Frequent Pattern Tree Construction , International Journal of Advanced Computer Research (ISSN (print): 2249-7277 ISSN (online): 2277-7970) Volume 4 Number 1 Issue 14 March, 2014.

Ling Yuan, she was born on November.04.1975, Wuhan, P.R.China.
She received her bachelor and master degree in Wuhan University. And she received her Ph.D degree in National University of Singapore. From Aug.2008, she began to be an associated professor in School of computer science, Huazhong University of Science Technology. Her research interests are data analysis, software engineering.