An Enhanced MEIT Approach for Itemset Mining using Levelwise Pruning

Tanvi P. Patel, Warish D. Patel

Abstract

Association rule mining forms the core of data mining and it is termed as one of the well-known methodologies of data mining. Objectives of mining is to find interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. Hence, Association rule mining is imperative to mine patterns and then generate rules from these obtained patterns. For efficient targeted query processing, finding frequent patterns and itemset mining, there is an efficient way to generate an itemset tree structure named Memory Efficient Itemset Tree. Memory efficient IT is efficient for storing itemsets, but takes more time as compare to traditional IT. The proposed strategy generates maximal frequent itemsets from memory efficient itemset tree by using levelwise pruning. For that firstly pre-pruning of items based on minimum support count is carried out followed by itemset tree reconstruction. By having maximal frequent itemsets, less number of patterns are generated as well as tree size is also reduced as compared to MEIT. Therefore, an enhanced approach of memory efficient IT proposed here, helps to optimize Main memory overhead as well as reduce processing time.

Keywords: Association rule mining, Itemset mining, Itemset tree, MEIT, maximal frequent pattern

I. INTRODUCTION

Association rule mining, one of the most significant and so much explored methodologies of data mining. Objective of that is finding interesting correlations, frequent patterns, associations or casual structures among sets of items in the transaction databases or other data repositories. In various fields such as telecommunication networks, market and risk management, inventory control, where association rules are widely used. There are mainly two interestingness measures support and confidence for rule.

\[
support(A \Rightarrow B) = P(A \cup B) \\
\text{confidence}(A \Rightarrow B) = \frac{P(A|B)}{P(A)}
\]

Generally, association rule mining can be seen as a two-step process:

i. Discover each frequent itemset; all of these itemsets will occur at least as frequently as a determined minimum support count, min sup.

ii. To generate association rules from discovered frequent itemsets; all of these rules have to fulfill minimum support and minimum confidence.

A distinguished methodology to Knowledge Discovery having Databases includes the proof of identity for association rules related to database attributes. Some task are defined for enhancing the efficiency of association rules algorithms, like reducing the number of passes over the database, sampling the database, addition of extra constraints on the structure of patterns through parallelization, accelerate search by using incremental technique, prune discovered associations, investigate parallel search.

Association rules are applied to different categories of Databases, which are data mining on transactional database, generalized association rule mining with taxonomy, spatial association rule mining, Temporal association rule mining, frequent pattern mining etc.

Frequent pattern mining is used for showing the periodic connections in database. Mining all probable rules from a database, instead of, is a computationally stubborn difficulty, in consequence of operation of explosion in the several sets of elements in which frequency counts must be computed, it might include numerous passes of the database. Hence, by make use of a single database pass to execute an incomplete computation of the totals time is mandatory, keeping all these in the form of tree structure.

Here, to recover the requirement of an incremental data mining approach, itemset tree mining is introduced. This approach relay on data structure called the itemset tree. This method is efficient for answering problems correlated to effectiveness of managing updates of data, accurateness of data mining results, handling input transactions, and responding user queries. Some capable algorithms to insert transactions into the itemset tree and for counting rate of recurrence of itemsets for queries regarding the command of association among items.

One of the efficient tree structures is memory efficient itemset tree. A well-organized data structure for carrying out targeted queries for itemset mining and association rule mining is the MEIT. During transaction insertion, it employs an effective node compression mechanism for
reducing the size of tree nodes. Furthermore, during transaction insertion or query processing, it relies on an on-the-fly node decompression mechanism for restoring node content.

Here, we aim to minimize number of patterns generated, which lead us minimizing computation time and main memory consumption. This may result in less number of rule generations. This tree structure will give fast traversal as well as fast access for generating rules. So mining time also will be reduced and main memory overhead also gets reduced.

II. RELATED WORK

A. R. Agrawal, T. Imielinski and A. Swami proposed [4] technique for mining association rules between sets of items in large databases. In this, all significant association rules generated between items in the Database. It is reducing the number of item sets by generating, closed, maximal, optimal item sets. Several algorithms are developed to reduce the number of rules using, (1) Non-redundant rules (2) Pruning techniques. The Demerits of system are Usefulness of association rules is strongly, limited by the huge amount of delivered rules, It is crucial to help the decision maker with an efficient technique for reducing the number of rules. The Merits of System are Reduce the number of item sets by generating closed, maximal optimal item sets, and several algorithms to reduce the number of rules, using non-redundant rules, and pruning techniques.

Fournier-Viger, P., Wu, C.-W., Tseng, V.S. introduce [5] Mining Top-K Association Rules. They proposed an algorithm to mine the top-k association rules, where k is the number of association rules to be found and is set by the user. Top-K Rules takes as input a transaction database, a number k of rules that the user wants to discover and the minconf threshold. It sets an internal minsup variable to 0. Then, it starts searching for rules. As soon as a rule is found, it is added to a list of rules L ordered by the support. The algorithm continues searching for more rules until no rule are found, which means that it has found the top-k rules. It is to mine the top-k rules with the highest support that meet a desired confidence. Here rule expansions method is used. They expanding rules in Top-K Rules left expansion and right expansion.

P. Fournier-Viger, V.S. Tseng introduces [6] Mining Top-K Non-Redundant Association Rules. In this there is an approximate algorithm named TNR using for mining the top-k non-redundant association rules. The algorithm is said to be approximate because it is guaranteed to find non-redundant rules. But the rules set up may not be the top-k non redundant rules. It was derived from the approach for generating association rules that is known as “rule expansions”, and includes strategies to ignore generating redundant rules. An evaluation of the TNR has excellent performance and scalability.

S. Dandu, B.L. Deekshatulu proposed [7] an improved Algorithm for Frequent Item sets Mining Based on Apriori and FP-Tree. In this, they introduce APFT (combination of Apriori and FP-tree). It includes correlated items & trims the non-correlated Itemsets. The advantage of APFT is that it doesn’t generate conditional & sub conditional patterns of the tree recursively and the results of the experiment show that it works fasts than Apriori and almost as fast as FP-growth. They have proposed to go one step further & modify the APFT to include correlated items & trim the non-correlated itemsets. This additional feature optimizes the FP-tree & removes loosely associated items from the frequent itemsets.

C.K. Leung, Q.I. Khan, Z. Li, T. Hoque introduces [8] CanTree which is a canonical-order tree for incremental frequent-pattern mining. It takes the content of the transaction database and arranging tree nodes with reference towards some canonical order, which can be determined by the user prior to the mining process or at runtime during the mining process. Hence, the building of the tree has need of only one database scan. It significantly reduces computation and time, because they easily find join paths and require only upward path traversals. It provides users with efficient incremental mining. CanTrees can be used for (i) constrained mining, (ii) incremental constrained mining, (iii) interactive mining, (iv)incremental interactive mining.

D. Burdick, M. Calimlim, J. Flannick, J. Gehrke, T. Yiu introduces MAFIA: A Maximal Frequent Itemset Algorithm [9], it is used for mining maximal frequent itemset from a transactional database; MAFIA uses a vertical bitmap representation for the transactional database. MAFIA performs best on dense datasets where large sub trees can be removed from the search space. It integrates a depth-first traversal of the itemset lattice with effective pruning mechanisms which increase performance. MAFIA is highly optimized for mining long itemsets on dense data. It includes search space pruning techniques and adaptive compression. Mafia uses two pruning strategies to remove non-maximal sets. The first is the look-ahead pruning first used in MaxMiner. The second is to check if a new set is subsumed by an existing maximal set.

K. Gouda, M.J. Zaki introduces [10] GenMax. It is an efficient Algorithm for Mining Maximal Frequent Itemsets which uses back-track search based algorithm. It uses a novel technique named “progressive focusing” for checking maximality. It is widely used for mining the exact set of maximal patterns efficiently and to exclude non-maximal itemsets, and uses “Diffset propagation” for fast frequency checking. (i) Superset checking optimization, (ii) Frequency testing optimization, (iii) Diffsets propagation. Mafia [9] mines a superset of the MFI, and requires a post-pruning step to eliminate non-maximal patterns. In contrast GenMax integrates pruning with mining and returns the exact MFI.

A. Hafez, J. Deogun, V. V. Raghavan proposed [3] the A Item-Set Tree: Data Structure for Data Mining. It is complete transaction lattice. Each node on the lattice represents a possibly large item-set. A count is attached to each node to reflect the frequency of item-sets. It is an enhancement in data capturing technology which leads to exponential growth in amounts of data being stored in information systems. This growth in turn has forced researchers to seek new techniques for extraction of knowledge implicit or hidden in the data.

M. Kubat, A. Hafez, V.V. Raghavan, J.R. Lekkala, W.K. Chen introduce [2] concept of item-set Trees for Targeted Association Querying. In this paper we can have
faster detection of itemsets and association rules. Here it establishes support, search for items, and generation of rules. Here it includes the concept of an itemset tree and presents an algorithm that generates data structure from a set of market baskets. It explains how to use it to process three different query types then a reports experiment illustrate the technique’s performance in terms of the amount of computation needed to response a query and scrutinizes the costs of building an itemset tree from data. It establishes the existence of a one-to-one mapping between the space of itemset trees and the space of market basket databases. This mapping guarantees that the resulting itemset tree does not depend on the order in which the market baskets have been presented. As shown in figure 1, itemset tree can constructing by following.

Figure 1. Itemset Tree Construction

P. Fournier-Viger, E. Mwamikazi, T. Gueniche, U. Faghihi proposed [1] Memory Efficient Item-set Tree (MEIT) for Targeted Association Rule Mining. It is incrementally updatable by putting in new transactions. An effective node compression mechanism is used for reducing the size of tree nodes. An on-the-fly node decompression mechanism is used for restoring node content. They have designed the MEIT based on three observations that are formalized by the following three properties. [1]

i. In an IT, transactions are insert by traversing branches in a top-to-bottom manner. It inserted both, by creating new node or by incrementing support of existing node.

ii. Queries on an IT are always processed by traversing tree branches in a top-to-bottom manner.

Let k be an IT node and parent(k) be its parent. The relationship \( i(\text{parent}) \subseteq i(k) \) holds between k and its parents. Therefore, it can be said that for any ancestor x of k, \( i(x) \subseteq i(k) \).

Figure 2 shows the construction of memory efficient itemset tree. From both figure 1 and figure 2 we can conclude that, IT is larger in amount of memory up to 60% than MEIT. But, MEIT takes almost twofold time for tree construction than IT.

Section 2 gives the description of an itemset tree, memory efficient itemset tree and various methodologies related to mining itemset tree. Section 3 explains how to use it to develop an enhanced memory efficient tree. Section 4 defines the algorithmic steps for developing enhanced memory efficient tree (E-MEIT). Section 5 then reports experiments illustrating the technique’s performance in terms of the amount of time needed for construction of tree and rule generation as well as memory consumed by it. Section 6 offers some concluding remarks and ideas for future research.

III. PROPOSED WORK

Existing methodology is capable for solving problems related to efficiency of managing data updates, accurateness of data mining results, handling input transactions, and responding user query is done. But in generation of tree structure, it contain redundant node which may having lower support count than minimum support count. We can generate frequent patterns mining from MEIT tree structure. Using GenMax approach making levels accordingly we generate MFI (Maximal Frequent Itemsets). From obtained MFI, rules are generated which are in less number of rules as compare to batch methodology. Traditional tree structure has slow access, more number of rules generation and slower traversal.

Here, taking an example to understand the structure of existing trees. In an IT, transactions are insert by traversing branches in a top-to-bottom manner. It inserted both, by creating new node or by incrementing the support of an existing node.

As, there is too large amount of data to fit in main memory, it expands to the disk. Our goal is to develop a search tree that will minimize disk accesses time and memory storage as well. As a result, enhanced MEIT (E-MEIT) is developed here for minimizing time for construction and access as well as memory for storing the items in the node. E-MEIT has root node which is null set. The root kept in main memory, so that a disk-read on the root is never required. E-MEIT uses reordered list for construction of tree in that each item stored at node along with their support count. In reordered list we do not consider item which has less support count than defined Min_sup. It makes level with reference to Minimum support count. It does back-track search traversal for obtaining MFI (Maximal Frequent Itemset). Those obtained MFI is used for generation for rule generation with use of Min_conf. So, as result we get association rules from E-MEIT acquiring less memory on disk and in reduced time compare to batch methodologies.

As a result, proposed technique enhances scalability for performing targeted queries. So, to reduce memory and time the proposed system will provide an efficient
approach to deal itemset mining and association rules in data mining. Here taking an example for enhanced MEIT.

### TABLE I. TRANSACTIONAL DATASET [1]

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>{1,4}</td>
</tr>
<tr>
<td>T2</td>
<td>{2,5}</td>
</tr>
<tr>
<td>T3</td>
<td>{1,2,3}</td>
</tr>
<tr>
<td>T4</td>
<td>{1,2,4}</td>
</tr>
<tr>
<td>T5</td>
<td>{2,5}</td>
</tr>
<tr>
<td>T6</td>
<td>{2,4}</td>
</tr>
</tbody>
</table>

For that count each item’s frequency count and make a reordered_list from Table 1 by arranging each frequent itemset with its frequency count in decreasing order. Here, all items with their frequency count from given dataset:

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

Here, minimum support is 2. Where, item \{3\} has less frequency count than defined. So, prune it for tree generation. Now, making E-MEIT from obtained reordered_list:

### TABLE II. REORDERED_LIST

<table>
<thead>
<tr>
<th>Transaction ID</th>
<th>Items</th>
</tr>
</thead>
<tbody>
<tr>
<td>T1</td>
<td>{1,4}</td>
</tr>
<tr>
<td>T2</td>
<td>{2,5}</td>
</tr>
<tr>
<td>T3</td>
<td>{2,1}</td>
</tr>
<tr>
<td>T4</td>
<td>{2,1,4}</td>
</tr>
<tr>
<td>T5</td>
<td>{2,5}</td>
</tr>
<tr>
<td>T6</td>
<td>{2,4}</td>
</tr>
</tbody>
</table>

Here, by taking minimum support 2, we get Maximal Frequent Itemset = \{1, 2, 4\}, \{2, 5\}

As a result, maximal frequent itemsets (MFI) are found. So, it can be concluded that in spite of having all frequent itemsets, only having MFI. Hence, fewer amounts of itemsets are used for rule generation using confidence threshold. As a result, time and memory constrains are reduced.

### IV. PROPOSED ALGORITHM

**Input:** Transaction Dataset, Min_sup, Min_conf  
**Output:** Association Rules

1. Scan the database  
2. Calculate frequency count of each item and generate Frequent-1 itemset along with their frequency count  
3. Obtain reordered_list with discarding item having Support<Min_sup  
4. Maintain reordered_list and construct E-MEIT accordingly  
5. Obtain MFI after leveling in E-MEIT  
6. For each MFI generate association rules from it using confidence constraint  
   - If support α⇒β  
     Then, confidence=(sup(α))/(sup(β))  
7. If conf≥Min_conf  
   Then, generate association rules  
8. Save the rules in output file

Above mentioned algorithmic steps are used for construction of E-MEIT and generating association rules. Below shown Figure 4 defines the workflow.
In Figure 4, itemsets are identified from the set of transactions from the user. Then count frequency count of each item and making a list, which has items along with their frequency count in decreasing order. By discarding items having less support compare to minimum support and make reordered list. From that E-MEIT constructed. Obtain maximal frequent itemset by leveling. Generating association rules from maximal frequent itemset based on confidence constraint. In order to having MFI compare to having frequent pattern itemset, we obtain less number of itemsets. So as a result of that memory consumption and time will get reduced than traditional approach.

V. EXPERIMENTAL ANALYSIS

Both approaches are tested in Eclipse tool. In Result analysis, performance evaluation of both approaches is evaluated based on two measures that are time and memory with using maximal frequent itemset and generating less number of association rules compare to batch methodologies.

![Figure 5. Time Comparison (In ms)](image)

![Figure 6. Memory Comparison (In mb)](image)

Here, we calculate total time by addition of time for MEIT and time for rule generation. And for calculating total time for E-MEIT, addition of time for E-MEIT, time for MFI and time for rule generation. These total times for MEIT and proposed E-MEIT are used for comparison and generating graph.

![Graph showing comparison](image)

Form above experiment results it can be concluded that, Proposed E-MEIT approach identifies less number of itemsets in lesser time as compare to existing MEIT approach. Proposed approach reduces the maximum memory space usage in most of cases.

The proposed E-MEIT approach requires less execution time as compared with memory efficient itemset tree and traditional itemset tree approach because MEIT approach utilizes more execution time for itemset generation. Moreover, the time used for proposed E-MEIT approach is less than the traditional approaches. Here, results for MEIT and E-MEIT for memory and time constrain.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Time (In ms)</th>
<th>Memory (In mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MEIT</td>
<td>E-MEIT</td>
</tr>
<tr>
<td>Mushroom</td>
<td>978</td>
<td>492</td>
</tr>
<tr>
<td>Chess</td>
<td>13250</td>
<td>10403</td>
</tr>
<tr>
<td>Connect</td>
<td>62133</td>
<td>14974</td>
</tr>
<tr>
<td>Accident</td>
<td>7913</td>
<td>6811</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Time (In ms)</th>
<th>Memory (In mb)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MEIT</td>
<td>E-MEIT</td>
</tr>
<tr>
<td>Mushroom</td>
<td>978</td>
<td>492</td>
</tr>
<tr>
<td>Chess</td>
<td>13250</td>
<td>10403</td>
</tr>
<tr>
<td>Connect</td>
<td>62133</td>
<td>14974</td>
</tr>
<tr>
<td>Accident</td>
<td>7913</td>
<td>6811</td>
</tr>
</tbody>
</table>

During the tree construction, MEIT approach inserts less items but tree constriction consumes more time whereas EMEIT requires shorter time duration as compare to existing one.

VI. CONCLUSION

In area for mining of data or itemsets, there is a challenging matter for storing itemsets in such a way that we can have easy access for stored data. Itemset tree is kind of tree structure which is incrementally updatable and efficiently storing items. Memory efficient itemset tree is modified on the basis of traditional itemset tree. This modified tree structure is efficient for memory constrains, but it acquires mostly twice amount of time compare to traditional structure.

Hence, it can be concluded that MEIT approach is not suitable option when time constrains are of main concern. So we proposed an enhanced approach for that called E-MEIT tree structure which obtaining maximal frequent itemsets. In our approach, one reordered list is maintained here for construction for enhanced tree structure and also prune items using minimum support count. By making level in E-MEIT, it has obtained maximal frequent itemset from backtrack search methodology. Association rule generation is done on basis of defined confidence threshold. So, as a result we can have less number of itemsets are obtained from E-MEIT as compare to traditional approaches. In this way, it is efficient regarding to memory and time. In future, more optimized tree structure for improvements on various parameters can be achieved using other techniques in data mining.
REFERENCES


