Parallel Method for Discovering Frequent Itemsets Using Weighted Tree Approach

Preetham Kumar
Department of Information and Communication Technology
Manipal Institute of Technology,
Manipal, India
prethk@yahoo.com

Ananthanarayana V S
Department of Information Technology
National Institute of Technology Karnataka
Surathkal, India.
anvs1967@gmail.com

Abstract— **Every element of the transaction in a transaction database may contain the components such as item number, quantity, cost of the item bought and some other relevant information of the customer. Most of the association rules mining algorithms to discover frequent itemsets do not consider the components such as quantity, cost etc. In a large database it is possible that even if the itemset appears in a very few transactions, it may be purchased in a large quantity. Further, this may lead to very high profit. Therefore these components are the most important information and without which it may cause the lose of information. This motivated us to propose a parallel algorithm to discover all frequent itemsets based on the quantity of the item bought in a single scan of the database. This method achieves its efficiency by applying two new ideas. Firstly, transaction database is converted into an abstraction called Weighted Tree that prevents multiple scanning of the database during the mining phase. This data structure is replicated among the parallel nodes. Secondly, for each frequent item assigned to a parallel node, an item tree is constructed and frequent itemsets are mined from this tree based on weighted minimum support.**

Keywords-attribute;cost;component;parallel;Weight

I. INTRODUCTION

The goal of knowledge discovery is to utilize those existing data to find out new facts and to uncover new relationships that were previously unknown, in an efficient manner with minimum utilization of the space and time.

Mining association rules is an important branch of data mining, which describes potential relations among data items (attribute, variant) in databases, the well-known Apriori algorithm [3] was proposed by R. Agrawal et al., in 1993. Mining association rules can be stated as follows: Let I={i1, i2, … im } be a set of items. Let D, the task-relevant data, be a set of transactions, where each transaction T is a set of items such that $T \subseteq I$. The quantities of items bought are not considered. Each transaction is assigned an identifier, called TID. Let A be a set of items, a transaction T is said to contain A if and only if $A \subseteq T$. An association rule is an implication of the form $A \rightarrow B$, where $A \subseteq I$, $B \subseteq I$, and $A \cap B = \square$. The rule $A \rightarrow B$ holds in the transaction set D with support s, where s is the percentage of transactions in D that contain AUB (i.e., both A and B). This is taken to be the probability, $P(A \cap B)$. The rule $A \rightarrow B$ has confidence c in the transaction set D if c is the percentage of transactions in D containing A that also contain B. This is taken to be the conditional probability, P(B|A). That is, $Support(A \rightarrow B)=P(A \cap B)=s$, $Confidence(A \rightarrow B)=P(B|A) = Support(A \rightarrow B)/Support(A)=c$.

Mining of association rules is to find all association rules that have support and confidence greater than or equal to the user-specified minimum support and minimum confidence respectively [2]. This problem can be decomposed into the following sub problems:

All itemsets that have support above the user specified minimum support are discovered. These itemset are called the frequent itemsets.

For each frequent itemset, all the rules that have user defined minimum confidence are obtained. The second sub problem, i.e., Discovering rules for all given frequent itemsets and their supports, is relatively straightforward as described in [1].

Discovering frequent itemsets, considered as one of the most important tasks, has been the focus of many studies in the last few years. Many solutions have been proposed using sequential or parallel algorithms based on user defined minimum support. However, the existing algorithms depend heavily on massive computation that might cause high dependency on the memory size or repeated I/O scans of the datasets. The published parallel implementations of association rule mining inherited most of these problems in addition to the new costly communication cost most of them need. Parallel association rule mining algorithms currently proposed in the literature are not sufficient for large datasets, and new solutions, that do not heavily depend on the repeated I/O scan, less reliant on memory size, and do not require a lot of communication costs between nodes, still have to be found. In addition, the existing algorithms discover all frequent itemsets based on user defined minimum support without considering the components of the transaction such as weight or quantity, cost and other relevant information of the customer which lead to profit.

This motivated us to design a new parallel algorithm that is based on the concept of Weighted Tree.

II. PROPOSED METHOD

This algorithm is divided into two phases. The first one requires only one scan of the database and generates a

978-0-7695-3521-0/09 $25.00 © 2009 IEEE
DOI 10.1109/ICCET.2009.194

special disk-based data structure called Weighted Tree. In the second phase, the Weighted Tree obtained is reduced to contain only frequent items and then all the branches of the tree are sorted according to the increasing order of the frequency. This tree is called an ordered Weighted Tree. This tree is replicated among nodes and mined in parallel.

The weighted minimum support is the minimum weight an itemset has to satisfy to become frequent. If an itemset satisfies user defined weighted support then we say that it is weighted frequent itemset.

Consider a sample database given in Table 1, in which every element of each transaction represents either quantity or cost of the respective attribute or item.

Table 1. Sample Database

TID\Atrributes	A	B	C	D
1	10	5	0	0
2	0	0	3	0
3	4	0	4	0
4	5	2	5	0
5	0	0	0	10

It may so happen that an itemset appears in a very few transactions in a large quantity or cost which leads to profit will not qualify as frequent itemset based on user defined minimum support. This results in a lose of information. In the sample database given in the Table 1, if user defined minimum support is 2 transactions then an item D is not frequent and will not appear in the set of frequent itemsets even though if it is bought in large quantity and leads to more profit than other frequent items.

Structure of Weighted tree

The idea of this approach is to associate each item with all transactions in which it occurs.The Weighted tree has two different nodes and its branch is shown in the Figure 1.

The first type of node labeled with attribute contains attribute name and two pointers, one pointing to the nodes containing transaction ids and weights and another is a child pointer pointing to the next attribute. This node represents the head of that particular branch.

The second type of node has 2 parts. First part is labeled TID represents a transaction number or id and second part of which is labeled weight, indicates quantity purchased in that transaction or cost or other components. This node has only one pointer pointing to the next object having this particular attribute.

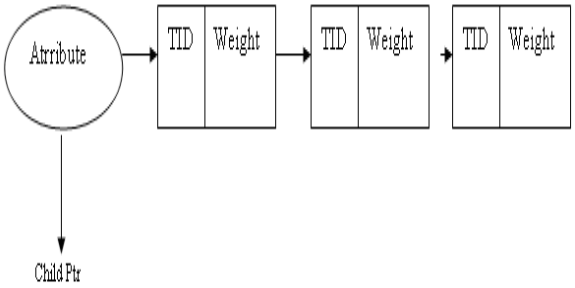

Figure 1. Branch of a Weighted Tree

Figure 2 represents the Weighted Tree corresponding to the Sample Database given in the Table 1.

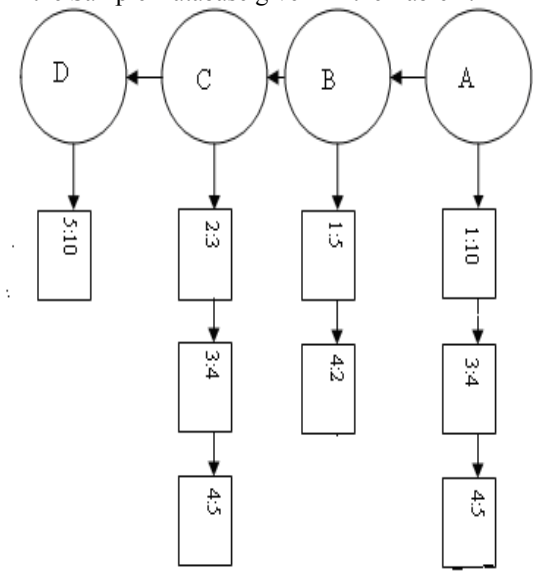

Figure 2 .Weighted Tree for Table1

The Parallel Algorithm

This algorithm involves 3 steps. They are

A. Construction of Weighted Tree

B. Removal of infrequent attributes of Weighted Tree

C. Arrange the attribute lists of Weighted Tree in an increasing order of their weighted minimum support

D. Parallel mining of frequent itemset at different nodes for assigned items based on weight.

A. Algorithm for constructing Weighted Tree

Input: The database D

Output: Weighted Tree

for each attribute weight w in a transaction t $\in$ D do

begin

Create a node labeled with weight w and add this

node to the respective attribute node.
end

B. **Removal of infrequent attributes of Weighted Tree**

Input : w_min_sup = weighted minimum support
Output: Reduced Weighted Tree.
for each attribute in a Weighted tree do
begin
if sum(weights of all nodes) < w_min_sup then
remove that branch from the tree
end

For example, In the above case if we consider min_sup=3 then only attributes A and C are frequent in the database. The attributes B and D are found to be infrequent.

If we consider w_min_sup = 10 then the attributes A, C and D will be frequent in the database. The reduced Weighted Tree of Table 1, is shown in Figure 3.

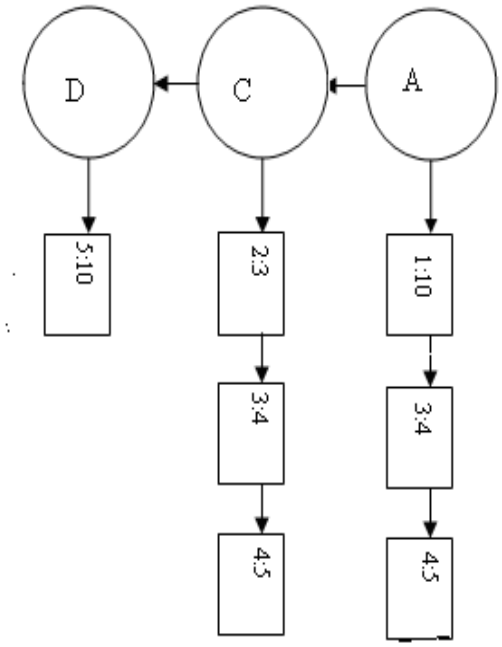

Figure 3. Reduced Weighted Tree of Table 1

C. Arrange the attribute list in an increasing order of their weighted minimum support

In Figure 4, we see that attribute D ahs weight 10, C has 12 and A has 19. The attribute lists are sorted in increasing order of their weights and is shown in Figure 4 and is called Ordered Weighted Tree.

The ordered Weighted Tree is replicated among all parallel nodes. By doing so a full set of transaction database containing only frequent items would be available to each processor to generate all globally frequent patterns with minimum communication cost at parallel node level. This replication is executed from a designated master node.

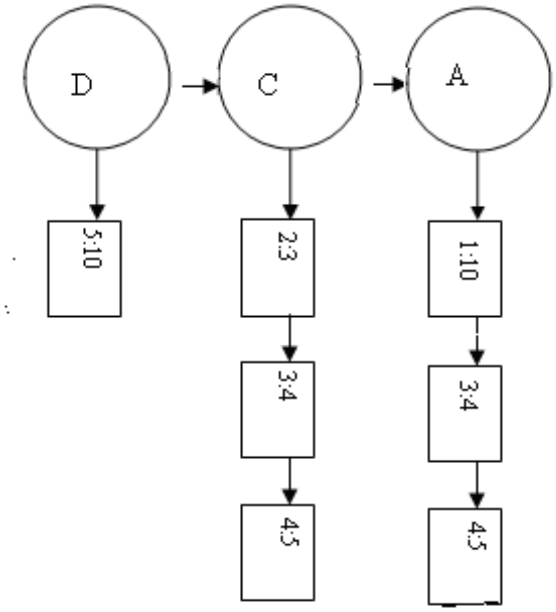

Figure 4. Ordered Weighted Tree

D. Parallel mining of frequent itemset at different nodes for assigned items based on weight.

This approach relies on distributing frequent items among the parallel nodes. After ordering the frequent items by their support, starting from the least frequent, each processor successively receives one item. The process is repeated until all items are distributed. In other words, if we have m processors, and n trees need to be built, assuming m< n then processor1 builds the tree for the least frequent item. Processor 2 builds tree for the next least frequent item, and so on up to processor m. After that processor 1 takes item m+1 and so on until all n items are distributed.

Each parallel node is responsible for generating all frequent patterns related to the frequent items associated to that node. To do so each node reads sub-transactions for each frequent items directly from the Ordered Weighted Tree, then builds independent and relatively small trees for each frequent item in the transaction database called item tree. Each parallel node mines separately each one of these item trees as soon as they are built. The trees are discarded as soon as mined. Finally, all frequent patterns generated at each node are gathered into master node to produce the full set of frequent patterns.

The each node of the item tree consists of item number and its count and two pointers called child and sibling

pointers. The child pointer points to the following item and sibling pointer points to the sibling node.

In our example, if we need to mine the Ordered Weight Tree in Figure 4, using 2 processors machine with weighted minimum support is equal to 10, the starting node for each parallel node would be D, the first frequent item. Processor 1 finds all frequent patterns related to items D and A. Processor 2 would generate all frequent patterns related to item C.

The first an item tree is built for item D. In this tree for D, all frequent items which are more frequent than D and share transactions with D participate in building the tree. The tree starts with the root node containing the item D. For each sub-transaction containing the item D with other items that are more frequent than D, a branch is formed starting from the root node.

If more than one frequent items share the same prefix, they are merged into one branch and their count fields are incremented. Figure 5 illustrates trees for frequent items D, C and A.

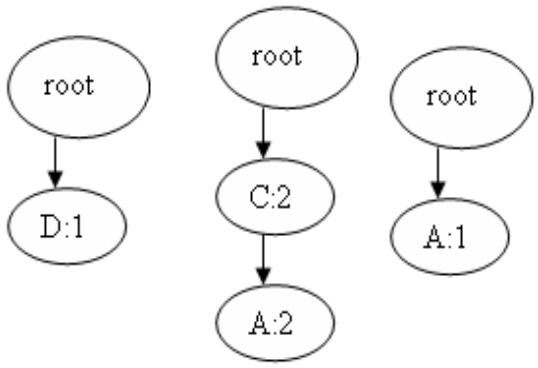

Figure 5. Item Trees for items D, C, A

The tree for any frequent item say f contains only nodes labeled with items that are more frequent or as frequent as f and share at least one transaction with f. Based on this definition, if A has weight greater than B then tree for B is larger than the tree for A. In other words, the higher the support of a frequent item, the smaller its tree is.

Input: A Ordered Weighted tree, w_min_sup

Output: set of all frequent itemsets at node i.

for every assigned item I, construct item tree for I containing transactions associated with I at node i.

Traverse item tree for I

for each maximal path from a root to a leaf in the tree for an item I

begin

T= {elements from I to leaf with weight}

if sum of weights of elements of [T] >= w_min_sup

MI= {elements from I to leaf with weight}

end

Apply downward closure property[1] to get all frequent itemsets from MI which contains all maximal frequent itemsets. If w_min_sup =10 then applying above algorithm, we see that at node 1, Tree for D does not contain any other node except itself. Therefore it is ignored. Similarly, tree corresponding to A contains only one node and is also ignored. Therefore at node 1 only frequent itmestes are one itemsets, {D} and {A}.

Similarly, at node 2, item tree corresponding to C is built. The maximal path corresponding to this tree is {A, C}. The weight of {AC} = 18 is greater than user defined weighted minimum support. Hence it is frequent. Also all its subsets are frequent by using downward closure property.

Hence Fw = { {A}, {C}, {D},{ A, C}}.

III. THEORETICAL ANALYSIS

A. Construction of Weighted Tree

In a transaction database D, if there G transactions, then G transactions have to be read by the algorithm to construct Weighted Tree. Therefore, this tree can be constructed in O (G) steps.

If the average length of the transaction t is equal to m weights, then there will Gm nodes in the tree. In the worst case, if there are M weights and G transactions in D, then there will be GM nodes in a Weighted Tree.

B. Reduction of Weighted Tree

If there are m attributes, then reduction of Weighted Tree is in O (m).

If there are k infrequent items then Reduced Weighted Tree will contain (m-k) attribute lists with at most G (m-k) nodes.

C. Arrange the attribute list in an increasing order of their weighted minimum support

If there are n attributes in the reduced Weighted tree then arranging them in an increasing order of the weight is in O (n2).

D. Parallel mining of frequent itemset at different nodes for assigned items based on weight.

If there are n frequent items then there will be n item trees. Each processor will construct one tree at a time. If there are m processors and n trees, m<n then this step is in O(n/m).

Merits

In our research we have implemented this algorithm and found that this algorithm is better than Apriori and FP-tree algorithm.

(i)This method is space as well as time efficient than Apriori since it involves candidate generation method and requires multiple scan over the database. Whereas, our method requires only one scan of the database.

(ii) The algorithm FP-Tree requires 2 scans to discover all frequent itemsets and mining patterns involve construction of

the condition based tree, which involves header table. Our method uses item tree to mine maximal frequent itemsets and does not involve header table.

IV. CONCLUSION

The Parallel algorithm for discovering frequent itemsets based on weight is a new method and is found to efficient when compared to Apriori and FP-tree. As such, we are still working on it with the aim of extending the application of this algorithm to various kinds of databases.

V. REFERENCES

[1] Arun K Pujari "Data mining Techniques": *Universities Press(Indis) Private Limited*. Hyderbad, 20003, India.

[2] J. Han and M. Kamber, "Data MiningConcepts and Techniuqes": San Franscisco, CA:. 2004, *Morgan Kaufmann Publishers*.

[3] Han, J., Pei, J., Yin, Y. "Mining Frequent Patterns without Candidate Generation", *Proc. of ACM-SIGMOD International Conference Management of Data.* Dallas, TX, 2000, 1-12.

[4] Han J, Pei Jian, Runying Mao(2004) " Mining Frequent Patterns without Candidate Generation: A Frequent Pattern Tree Approach" , *Data Mining and Knowledge Discovery,* Kluwer Academic Publishers, Netherland*"* 53-87.

[5] Agrawal Charu and Yu Philip,(1998) " Mining Large Itemsets for association rules. *Bulletin of the IEEE Computer Society Technical committee on Data Engineering,* 21, No.1.

[6] Ananthanarayana V. S, Subramanian, D.K., Narasimha Murthy M.(2000). "Scalable, distributed and dynamic mining of association rules" *In Proceedings of HIPC'00.* Springer Verlag Berlin,Heidelberg, 559-566.

[7] O.R.Zaiane, M. El-Hajj, and P. Lu. "Fast parallel association rule mining without candidacy generation" In Proc. of the IEEE 2001 International Conference on Data Mining, San Jos, CA, USA, December 2001