International Journal of Engineering and Management Research e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume- 9, Issue- 5 (October 2019)
www.ijemr.net https://doi.org/10.31033/ijemr.9.5.22

Mining Algorithm for Weighted FP-Growth Frequent Item Sets based on Ordered FP-Tree

Yuanyuan Li[1] and Shaohong Yin[2]
[1]Master, School of Computer Science and Technology, Tianjin Polytechnic University, CHINA
[2]Associate professor, School of Computer Science and Technology, Tianjin Polytechnic University, CHINA

[2]Corresponding Author: 13834814101@163.com

ABSTRACT

FP-growth algorithm is a classic algorithm of mining frequent item sets, but there exist certain disadvantages for mining the weighted frequent item sets. Based on the weighted downward closure property of the weighted model, this paper proposed a method to reduce the use of storage space by constructing a weight ordered FP-tree, so as to improve the generation efficiency of weighted frequent item sets.

***Keywords*—** Data Mining, Association Rules, Ordered FP-Tree, Weighted Model, Weighted Ordered FP-Tree

I. INTRODUCTION

Association rule mining occupies a very important position in data mining, the purpose of which is to find out the part that people are interested in from the massive data, so as to make better use of the useful information extracted. The Apriori algorithm[1] was proposed by Agrawal et al. in 1994, but it needs to scan the database multiple times and generate a large number of candidate sets. Therefore, in 2000, Han et al. proposed the FP-growth algorithm[2] based on FP-tree data structure for the deficiencies of the Apriori algorithm. The algorithm only needs to scan the database twice to avoid generating a large number of candidate sets.

In the process of mining association rules, fp-growth algorithm spends most of its time on the construction and traversal of fp-tree and conditional fp-tree. Therefore, if we can improve the efficiency in this aspect, the overall efficiency of the algorithm will be greatly improved. Based on the analysis in this aspect, an optimization is proposed for the algorithm itself, namely replacing traditional fp-tree with ordered fp-tree, so that every node and child nodes in the tree are arranged from small to large according to the sequence number of items, reducing the construction time of the tree.

Classical algorithms default to transactions of equal importance, but in real life the importance of each transaction is different and unevenly distributed. To solve this problem, weighted association rules mining algorithm emerged. In this paper, fp-growth algorithm is studied by using the weighted model based on fp-tree[3] proposed by Chen wen in 2012. On the basis of the weighted model, a new algorithm for frequent pattern mining of ordered fp-tree is proposed by using the method of constructing ordered fp-tree.

II. ORDERED FP-TREE

Fp-growth algorithm recursively mines frequent pattern trees by storing frequent pattern information in frequent pattern trees, obtaining frequent pattern sets, and then obtaining association rules. The algorithm is mainly divided into two steps :(1) fp-tree construction; (2) mining based on fp-tree. This paper mines association rules by constructing ordered fp-tree.

Each node of the ordered fp-tree consists of four domains, namely, the node serial number item-id, the number of paths to node nodes count, the joint pointer domain pc-link of parent node pointer and child node pointer, and the joint pointer domain node-link of sibling node pointer and node pointer of the same name. Among them, the item - id instead of the node name, serial number according to the frequent items support after descending order, records relating to the node represents in the serial number in the table. The reason why the joint pointer field is adopted is that the pointer of the child node and the sibling node are only used in tree construction, while the pointer of the parent node and the pointer of the same name are only used in mining. Therefore, when establishing an ordered fp-tree, pc-link points to the pointer of child node and node-link points to the pointer of brother node. When mining frequent patterns, pc-link points to the parent node pointer and node-link points to the node pointer of the same name. This improves the space utilization rate. The children of the same parent node are sorted from small to large according to the size of item-id, thus constructing an ordered fp-tree.

The difference between traditional fp-tree and ordered fp-tree is as follows :(1) the node in traditional fp-tree saves the node name item-name, while the node in ordered fp-tree saves the node serial number item-id, and changes the node serial number into the node name at the last output. (2) nodes in traditional fp-tree are arranged out of order, while nodes in ordered fp-tree are arranged in ascending order of item-id.

The following is the construction algorithm of

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume- 9, Issue- 5 (October 2019)
https://doi.org/10.31033/ijemr.9.5.22

ordered fp-tree:

Input: a transaction database D with minsup minimum support.

Output: ordered fp-tree

The transaction database D is scanned to obtain the support count of frequent items, and the frequent item set is obtained. The frequent items are arranged in descending order according to the support count, which is denoted as L.

Create a root node root "null" for ordered fp-tree, each transaction in the transaction database T perform the following operations: delete the transaction does not meet the minimum support, the frequent items in descending order support count, with a serial number in the L item - id instead of every frequent items, the serial number to start from 0 ascending, remember after sorting sequence for [p | p], p for the first element in the sequence, p for other elements. Then call the insert_tree(p| p,root) method.

The implementation method of ert_tree(p| p,root) is as follows: if root does not have child nodes, remember the inserted node is node, then node.item-id=p, node.count=1, and the parent pointer node points to root. Otherwise, find the insertion location of p in the child node of root. If the same node node as p is found, the value of node.count will be increased by 1. If the same node is not found, create a new node node, set the values of its various fields, and insert the node before the next node of p. If P is not null, insert_tree(P |P,root) is recursively called.

The following is the transaction database based on table I and the ordered fp-tree constructed by the above ordered fp-tree construction algorithm, as shown in Figure 1.

TABLE I
THE TRANSACTION DATABASE

Transaction	Transaction set	Ordered transaction set
T1	A,C,G	A,G,C
T2	A,B,E,F	A,E,F
T3	E,I	E
T4	A,D,E,G	A,E,G,D
T5	A,C,E,G	A,E,G,C
T6	A,F,D	A,D,F

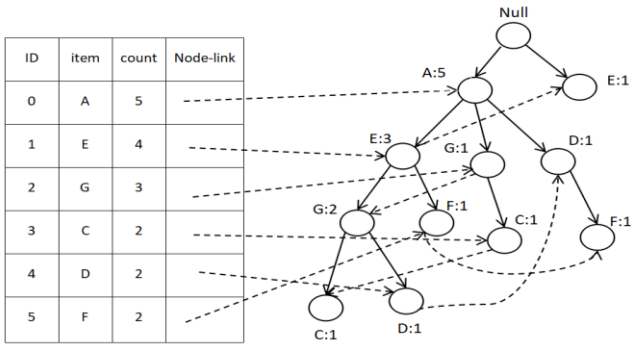

Figure 1: Ordered fp-tree and item entry table

After the construction of the ordered fp-tree, the values of the two joint pointer fields need to be changed so that pc-link points to the parent node and node-link points to the node with the same name, i.e. the next node of the same item.

III. WEIGHTED MODEL

The project set I = { i1, i2, ..., ik} is a set of k different items. The transaction database D={T1, T2,..., Tn}, where every transaction Ti(i = 1 , 2, ..., n) contains a unique transaction identifier TID and a subset of I.

Definition 1 Let each item ij(j=1,2,...,k) in item set I={i1,i2,...,ik} have a weight W(j), 0<=W(j)<=1. The project item set X also has a corresponding weight, which is recorded as $WI(X) = \sum_{ij \in X} W(j) / |X|$. The weight of the transaction t in the transaction database D is denoted by $WT(t) = \sum_{ij \in t} Wij / |t|$. The weighted support for association rule $A \Rightarrow B$ is denoted as $W\sup(A \cup B) = (\sum_{t \in T, A \cup B \subset T} WT(t) / \sum_{t \in T} WT(t))$.

Definition 2 Let the minimum weighted support be $W\min sup$. If the item set X is a frequent item set, the weighted support is not less than the minimum weighted support, i.e. $W\sup(X) \geq W\min sup$.

Theorem 1 If item sets X and Y are respectively a subset of I, i.e. $X \subset I$, $Y \subset I$ and $X \subset Y$, then $W\sup(X) \geq W\sup(Y)$.

Proof: Let the transaction data set containing X, Y be Tx, Ty, because $X \subset Y$, for $\forall t \in Ty$, there must be $t \in Tx$, so $\sum_{t \in T, X \subset t} WT(t) \geq \sum_{t \in T, Y \subset t} WT(t)$, $W\sup(X) \geq W\sup(Y)$ can be obtained.

Theorem 2 (weighted downward closure property) When $X \subset I$, $Y \subset I$, $X \subset Y$ and X and Y have the same prefix, if X is not frequent, Y must be infrequent.

Proof: From theorem 1, we know that $W\sup(X) \geq W\sup(Y)$. If X is not frequent, we can know $W\sup(X) < W\min sup$ by definition 2. So $W\sup(Y) < W\min sup$, i.e. Y is not frequent.

The weighted model proposed here satisfies the weighted downward closure property, ensuring that the subset of frequent item sets is also frequent, and the parent-set of infrequent item sets is also infrequent, so that frequent item sets can be merged and infrequent item sets can be pruned. Applying the weighted model to the weighted association mining algorithm, we can optimize the algorithm process to mine the weighted frequent item sets.

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume- 9, Issue- 5 (October 2019)
https://doi.org/10.31033/ijemr.9.5.22

IV. IMPLEMENT OF WEIGHTED ORDERED FP-TREE ALGORITHM

Taking table I transaction database as an example, the weighted ordered fp-tree frequent pattern mining algorithm includes the following four steps:

Step 1: scan the database and sort the project in descending order of support.

Scanning the transaction database of Table I, we get the project set I = { A, B, C, D, E, F, G, I} . We arrange the items in the project set in descending order of support. When the two projects have the same support, they are arranged in alphabetical order. This arrangement ensures that the sorting results are consistent each time. The ordering of the items and their weights are shown in Table II.

TABLE II
PROJECT NAME AND ITS WEIGHT

Project name	Weight
A	3
E	4
G	2
C	3
D	4
F	5
B	1
I	1

Step 2: scan the database to normalize the transaction weight.

Scanning transaction database calculates the weight of each transaction, i.e. $WT(t) = \sum_{ij \in t} Wij / |t|$, for example, t1 = { A, C, G} , WT(t1) = (3 + 3 + 2) / 3 = 2.67. For the same reason, the weight of each transaction is shown in the third column of Table 3.In order to solve the problem that the weight is greater than 1, the transaction weight is normalized, which is also beneficial to the improvement of the algorithm efficiency. The sum of the transaction weights is $W(T) = \sum_{t \in T} WT(t)$, and the transaction weights are normalized to calculate the weight ratio of each transaction, for example, t1 = { A, C, G} , WT(t1) = 2.67, W(T) = 18.67, WT(t1) / W(T) = 0.1430. Similarly, the normalized transaction weights are shown in the fourth column of Table III. Also, WT(t1) / W(T) + WT(t2) / W(T) + ... + WT(ti) / W(T) = 1.

TABLE III
TRANSACTION WEIGHT TABLE

Transaction	Item set	Transaction weight	the normalized transaction weight
T1	A,C,G	2.67	0.1430
T2	A,B,E,F	3.25	0.1741
T3	E,I	2.50	0.1339
T4	A,D,E,G	3.25	0.1741
T5	A,C,E,G	3.00	0.1607
T6	A,F,D	4.00	0.2142
The sum of the transaction weights		18.67	1.0000

Step 3: construct the weighted ordered fp-tree.

Weighted ordered fp-tree is defined as follows:

(1) The structure of weighted ordered fp-tree is similar to that of the above ordered fp-tree. Each node is still composed of four domains, with the difference that the count recording the path number of node arrival points is changed to the sum of the normalized weights of the transaction set to which the node belongs. The four domains are respectively the node serial number item-id, the sum count of the normalized weight of the transaction set to which the node belongs, the pc-link joint pointer domain of the parent node pointer and child node pointer, and the node-link joint pointer domain of the sibling node pointer and the node pointer of the same name.

(2) The structure of item header table is node-id, node-link and weight. Where, node-id is the item sequence number of the item header table; Node-link points to the first node with the same value as the node-id field. Weight records the sum of the normalized weights of the transaction set to which the item belongs. For example, the node id = '2', 'G', WI (' G ') = 0.1430 + 0.1741 + 0.1607 = 0.4778.

The following is a weighted FP-tree construction algorithm:

Input: transaction database D, minimum weighted support Wminsup

Output: Weighted FP-tree

Scan the transaction database, calculate the weighted support of each item, and establish the item header table in descending order of weighted support.

We create a root node (i.e. root) with the initial value set to "Null" . For each transaction T in the transaction database, the following operations are performed: deleting items that do not satisfy the minimum weighted support degree, frequent items are sorted in descending order of support degree, and the sorted transaction T is [i, I] , where i is the first element of the transaction and I is a collection of remaining elements. Recursively call updataFPtree([i, I] , root) . The method is performed as follows: First, look up i in the child node of root. If the same node as i is found, the count field of the Node records the weight of the normalized transaction set of the Node; If the same node as i is not found, we create a new node and set its count field value to the normalized weight of the transaction set of Node. The new node is linked to its parent node root, and is linked to the same node through the node chain structure. If I is not empty, recursively call updataFPtree([i, I] , root) .

Let Wminsup = 0.2, the constructed weighted

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume- 9, Issue- 5 (October 2019)
https://doi.org/10.31033/ijemr.9.5.22

ordered fp-tree is shown in Figure 2.

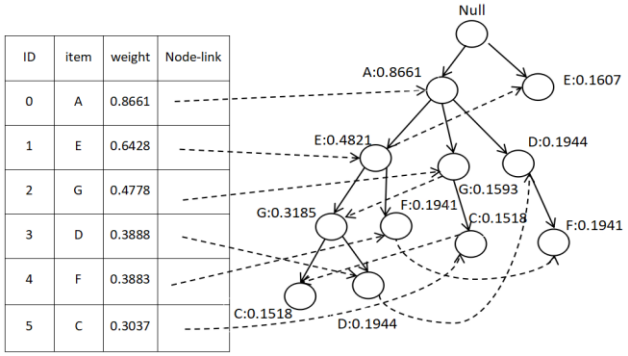

Figure 2: Weighted ordered fp-tree

Step 4: mining weighted ordered fp-tree.

According to theorem 2, the item set with the same prefix has weighted downward closure property, so we start from the second item of the head table to generate the condition pattern base of each item, and construct the weighted condition FP-tree according to the condition pattern base. Table IV shows the corresponding weighted condition FP-tree and frequent mode.

TABLE IV
WEIGHTED CONDITION FP-TREE AND FREQUENT MODE

Project name	Weighted condition FP-tree	frequent mode
E	<A:0.8661>	{A,E:0.8661}
G	<A:0.5197,E:0.3214>	{A,G:0.5197},{E,G:0.3214}, {A,E,C:0.3214}
D	<A:0.3464>	{A,D:0.3464}
F	<A:0.3464>	{A,F:0.3464}
C	<A:0.3464>	{A,C:0.3464}

V. ALGORITHM ANALYSIS AND EXPERIMENTAL RESULTS

By normalizing transaction weights, the weighted model used in the proposed algorithm is more efficient than the traditional weighted model, and the weighted downward closure of the weighted model saves some mining time, thus improving the efficiency of the algorithm. By using the ordered fp-tree data structure to store data, candidate item sets need not be generated repeatedly, which reduces The Times of scanning the database and saves the I/O overhead. In addition, the process of discovering long frequent item sets is transformed into recursively discovering some short frequent item sets and then concatenating suffixes, thereby reducing the search overhead.

Experimental performance comparison was made between the weighted ordered fp-tree algorithm and the weighted fp-tree algorithm (the weighted ordered fp-tree algorithm is denoted as WOFP algorithm, and the weighted fp-tree algorithm is denoted as WFP algorithm), and the experimental results were shown in Figure 3. The experimental environment: Windows10 operating system, 8GB of memory, and Pycharm2018 development environment. The transaction set is generated by python code for 100,000 transactions, each with a maximum length of 10.

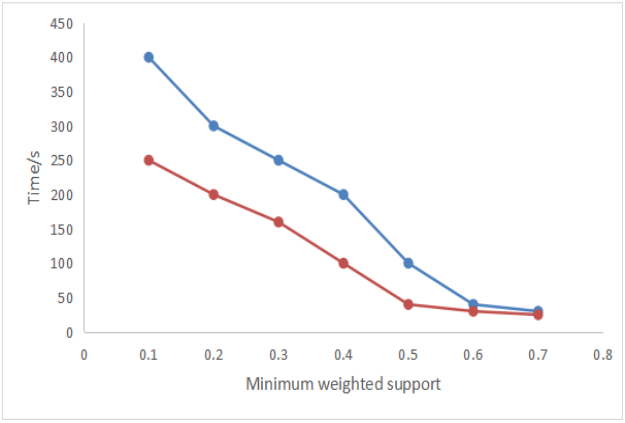

Figure 3: Experimental result

In the experiment, with the minimum weighted support increasing from 0.1 to 0.7, the execution time of the two algorithms gradually decreased. With the same minimum weighted support, the improved WOFP algorithm performs faster than the WFP algorithm.

VI. CONCLUSION

This paper improves the weighted fp-tree algorithm and replaces the traditional fp-tree with ordered fp-tree, which makes full use of the structure space of the tree and saves the time of mining fp-tree. In addition, weight is added to the data to make the data more convincing. Compared with the weighted fp-tree algorithm, the improved algorithm has better performance.

REFERENCES

[1] Agrawal, Rakesh & Srikant, et al. (1994). Fast algorithms for mining association rules. *In: International Conference on Very Large Data Bases. Santiago, Chile*, pp. 487-499.

[2] Han J, Pei J, & Yin Y. (2000). Mining frequent patterns without candidate generation. *In: ACM SIGMOD International Conference on Management of Data. Dallas, TX, United states*, pp. 1-12.

[3] Wen Chen. (2012). Mining algorithm for weighted frequent pattern based on Fp tree. *Computer Engineering, 38*(6), 63-65.

[4] C.H. Cai, W.C. Ada, W.C. Fu, & C.H. Cheng, et al. (1998). Mining association rules with weighted items. *In:*

e-ISSN: 2250-0758 | p-ISSN: 2394-6962
Volume- 9, Issue- 5 (October 2019)
https://doi.org/10.31033/ijemr.9.5.22

Proceedings of the International Database Engineering and Application Symposium. Cardiff, UK, pp. 68-77.
[5] F. Tao, F. Murtagh, & M. Farid. (2003). Weighted association rule mining using weighted support and significance framework. *In: 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '03. Washington, DC, United states*, pp. 661-666.
[6] Zhaopeng Pan, Peiyu Liu, & Jing Yi. (2018). An improved fp-tree algorithm for mining maximal frequent patterns. *In: 10th International Conference on Measuring Technology and Mechatronics Automation (ICMTMA). Changsha, China*, pp. 309-312.
[7] T. Saha, et al. (2018). Association rules to analyze hospital resources with mortality rates. *In: 5th International Conference on Business and Industrial Research (ICBIR). Bangkok, Thailand*, pp. 51-56.
[8] Yan Shi & Yan Fu. (2006). Algorithm for frequent pattern mining based on fp reference tree/list. *Computer Science, 33*(6), 206-209.
[9] K. Sun & F. Bai. (2008). Mining weighted association rules without preassigned weights. *IEEE Transactions on Knowledge and Data Engineering, 20*(4), 489-495.
[10] Yan Wang, Haiyan Xue, & Lingling Li, et al. (2010). An improved algorithm for mining weighted frequent itemsets. *Computer Engineering and Applications, 46*(23), 135-137+197.
[11] Haitao Hao & Yuanyuan Ma. (2016). Research on ecommerce commodity recommendation system-based on mining algorithm of weighted association rules. *Modern Electronic Technology, 39*(15), 133-136.
[12] Shuai Yue & Shaohong Yin. (2018). Study on frequent patterns mining based on sorted FP-Tree and two-dimensional table. *Journal of Harbin University of Commerce(Natural Science Edition), 34*(6), 692-697.
[13] Hongguang Xiao, Guoqun Deng, & Wen Tan, et al. (2018). A weighted association rules mining algorithm based on matrix compression. *Measurement and Control Technology, 37*(3), 10-13.