International Journal of Engineering & Technology, 7 (2.22) (2018) 45-48

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

An enhanced constraint based technique for frequent itemset mining in transactional databases

Ramah Sivakumar [1*], Dr. J.G.R. Sathiaseelan [2]

[1]*Department of Computer Science , Bishop Heber College, Trichy-17. India*
[2]*Department of Computer Science, Bishop Heber College, Trichy-17 . India*
**Corresponding author E-mail: rmhsvkmr@yahoo.co.in*

Abstract

Mining frequent patterns is one of the wide area of research in recent times as it has numerous social applications. Variety of frequent patterns finds usage in diverse applications and the research to mine those in an optimized way is an important aspect under consideration. So far, many algorithms had been proposed for mining frequent itemsets and each has their own pros and cons. The basic algorithms used in the process are Apriori, Fpgrowth and Eclat. Many enhancements of these algorithms are ongoing process in recent times. In this paper, an enhanced Varied Support Frequent Itemset (VSFIM) algorithm is proposed which is an enhancement of FPGrowth algorithm. Unique minimum support for each item in the transaction is provided and then mining is done in the proposed approach. The performance of the proposed algorithm is tested with existing algorithms. It is found that VSFIM outperformed the existing algorithms in both processing time and space utilization.

Keywords: *Itemset, FPGrowth algorithm, Frequent Patterns, support.*

1. Introduction

In recent times, frequent pattern mining has attracted researchers' in knowledge discovery from databases due to its wide applicable areas such as association rule framing, software bug detection, web log mining and so on. The problem involves a transactional database T with transactions. Each transactions consists of items I, which is denoted as T = {A|A⊆I} that is contained in the set of transactions. Frequent pattern mining is done based on the minimum support threshold. If an itemset is greater than or equal to minimum support, then the itemset is considered as frequent itemset. Each item in the transaction may have unique nature. So setting the same min-sup for all the items cannot lead to a meaningful mining. In the proposed approach, min-sup for each item has been made unique. So the first pruning is based on the unique support constraint of each item. Only if the item satisfies the min-sup, then it is taken into account for mining process. Thereby pruning the number of infrequent items in the beginning itself. This leads the algorithm to process faster. Space optimization is one of the algorithmic goals to be achieved. As in the proposed algorithm, only the most frequent itemsets are mined, and the less frequent itemsets are pruned, leads to less space utilization and processing time.

The rest of the paper is organized as follows: Section – 2 deals with the related workofFPGrowth algorithm, section – 3 is discussed with the proposed algorithm, Section – 4 with the experimental results and interpretations and section – 5 is where the conclusion and future scope is given.

2. Related work

Frequent Itemset Mining was first proposed by Agrawal et al. (1993) [1]. Many algorithms have been proposed by many research people so far to mine frequent patterns. Out of which FPgrowth [2] algorithm is found to be one of the most preferred among the tree based algorithms for FIM. It is a high performance algorithm which employs only two passes on the database. Many enhancements have been made so far to make the algorithm even better. The algorithm works on the principle of divide and conquers strategy. The basic steps involved in mining FI's using FPGrowth are as follows:

1. Scan the database once to find the ordered itemsets in the transactions.
2. During second scanning build the FP tree and mine the frequent itemsets.

FPGrowth algorithm uses short frequent patterns to find the long frequent patterns by concatenating the frequent items. The suffix will be of least frequent count. The suffix will occupy the leaf node of the FP tree. Many enhancements and improvements have been done in FPGrowth approach. It includes depth-first approach, exploration of hyper structure mining H-mine [3] algorithm was proposed, array based implementation of prefix tree structure, Recursive Elimination algorithm [4] which executes without prefix trees and so on.

For constrained based mining, Depth Project algorithm which mines only maximal frequent itemsets and involves depth first and breadth first traversal of the itemset lattice method. Another maximal frequent itemset mining algorithm is GenMax[5] , which uses backtrack search strategy for mining. The combina-

tion of top down and bottom up strategy to identify the MFI's is incorporated in Pincer-search [6] algorithm.
Definition of terms used in this paper

- **Set:** Collection of elements.
- **Pattern or Itemset:** A set of items.
- **Support:** The frequency of an itemset in a dataset.
- **Min-support:** The minimum frequency that an itemset should have to be frequent.
- **Frequent pattern:** An itemset whose frequency is at least min-support.
- **Candidate:** Any itemset that might be a frequent pattern
- **Varied support:** Multiple support count

2.1. FPGrowth Algorithm

As stated earlier, this algorithm uses divide and conquer strategy. In the first pass the frequent – 1 items are found out based on their frequency in the database, which is compared with the minimum support threshold value. If the item is equal to or greater than the minimum support value then it is considered as frequent item. Least frequent items are those which are less than the minimum support. The items in each transaction are sorted as most frequent to least frequent items. The least frequent items are pruned in this process. Then the FP tree is constructed during the second scan of the database. After the complete FP tree is built, conditional pattern base for each item is built. This conditional FP tree is called as prefix tree. The prefix trees are built until the FP tree is empty. Frequent items are then mined from these prefix trees.

2.2. Example of FPGrowth Algorithm

Calculation of Minimum support for 30%

Min.support=(30/100*10) = 3

Here 30 indicates the percentage of calculation and 10 is the number of transactions.

Table:1 Sample database

TID	ITEMS
10	a,b
20	a,e,f
30	c,d
40	a,b,h
50	c,d
60	a,c
70	a,b
80	e,f
90	c,d,g
100	a,b

Procedure FP-growth(Tree, α)
{
(1) if Tree contains a single prefix path
(2) then {
(3) let P be the single prefix-path part of Tree;
(4) let Q be the multipath part with the top branching node replaced by a null
root;
(5) for each combination (denoted as β) of the nodes in the path P do
(6) generate pattern β ∪ α with support = minimum support of nodes in β;
(7) letfreq pattern set(P) be the set of patterns so generated; }
(8) else let Q be Tree;
(9) for each item ai in Q do {
(10) generate pattern β = aiU α with support = ai .support;
(11) construct β's conditional pattern-base and then β's conditional FP-tree Treeβ ;
(12) if Treeβ = Ø
(13) then call FP-growth(Treeβ , β);
(14) letfreq pattern set(Q) be the set of patterns so generated;
}
(15) return(freq pattern set(P) Ufreq pattern set(Q) U (freqpattern set(P) × freq pattern set(Q)))
}

Figure:1 FPGrowth algorithm

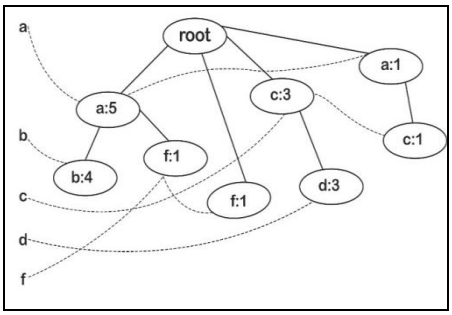

Figure 2: Complete FP tree with header table

The root node is always null for a complete fp-tree. Fp-tree is then mined instead of mining the database as it contains the frequency details of the itemsets. Then the conditional pattern base for the individual items is framed and prefix treesare built for each itemset to find the frequent itemsets.

3. Methodology

3.1. Proposed Algorithm (VSFIM)

In the proposed approach multi varied support count for each unique items are given instead of setting a single minimum support for all the items. In the first level of pruning itself, the items are pruned based on the varied support count set for each item. Based on the compared itemsets, the header table and the ordered itemlist are constructed. From the ordered itemlist, FPtree is built for each transaction on the second pass. This technique reduces the number of items in the first phase itself.

Given below is the flow diagram of proposed VSFIM algorithm.

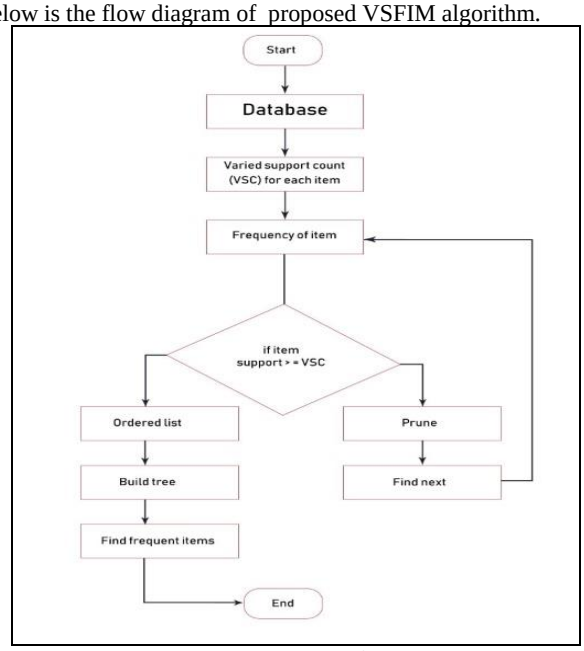

Figure 3: Flow diagram of VSFIM

3.2. VSFIM Algorithm

1. Begin
2. Compute I and f(i) where i∈I.
3. Construct varied support table for each item
4. Compare the frequency with related support
5. Sort I in descending order of varied support count and prune infrequent items
6. Create root of FP tree with root as NULL.
7. For all transactions $t_i \in T$ do
8. Arrange frequent items of t_i in descending order based on varied support count to form the ordered list of items
9. Add_Node (t_i);
10. End for
11. CP-Growth (Tree, θ);
12. End;

Figure 4: VSFIM algorithm

3.3. Procedure Add_node

```
Add_node (ti,T)
{
1. If item i has child J such that item[J]=item[i] then
2. Count[J]:=count[J]+1;
3. Else
4. Add new node J with count:=1
5. Super node linked to I and same item nodes are linked
   with node links by next.
6. End if
7. If i≠ Øthen
8. Add_node(i,J)
9. End if
10. Return;
11. }
```

Figure 5: add_node procedure of VSFIM

3.4. Procedure CP-Growth

```
CP-Growth (Tree, θ)
{
1. If tree contains a single path P then
2. For all combination ∂ of the nodes in path P do
3. Generate ∂ U θ with minimum support of nodes in ∂
4. End for
5. Else
6. For all Ii in the header table do generate itemsets ∂:= Ii U
   θ with support:=support[Ii]
7. Construct ∂ s conditional pattern base and then ∂s condi-
   tional FP-tree Tree∂
8. If Tree ≠ Øthen
9. CP-Growth (Tree∂,∂)
10. End if
11. End for
12. End if
13. Return;
}
```

Figure 6: CPGrowth procedure of VSFIM

The three major steps involved in proposed VSFIM algorithm are:

(1) Find the frequent – 1 itemsets using the varied support count of each item. The items which do not satisfy the min.support will be pruned.

(2) Build the complete FP-tree from each transaction using Add_node method. Each node contains the item name, item count and the link to the next node.

(3) By recursive execution of CP_growth method the conditional pattern base and prefix tree for each item is built based on the *min-support* from the complete fp-tree. Frequent itemsets are found and stored in the output file.

Example – 1

Transaction database shown in Table -1, is taken for explanation. The minimum support and the frequency count for each item is shown below:

Table 2: Support Comparison

Item	Min. support	Frequency count	Header table
a	5	6	a
b	4	4	b
c	3	4	c
d	3	3	d
e	2	1	-
f	2	2	f
g	2	1	-
h	3	2	-

The steps involved in the process are:

1. The minimum support of each item is compared with its frequency of occurrence in the database and the items in each transaction are ordered according to their *min- support* count in decreasing order. This process is done during the first pass on the database.
2. During this process the least frequent items are pruned so that the FP tree is built based on the frequent -1 items. This technique reduces the space and time complexity of the algorithm.
3. Tree building and finding FIM is the next step.

The ordered list of the database:

Table 3: Ordered list

TID	ITEMS	Ordered list
10	a,b	a,b
20	a,e,f	a,f
30	c,d	c,d
40	a,b,h	a,b
50	c,d	c,d
60	a,c	a,c
70	a,b	a,b
80	h,f	f
90	c,d,g	c,d
100	a,b	a,b

In Table-3, thedecreased order of items are listed where items e, g and h are pruned as they are not frequent.

3.5. Tree Building

By using the Add_node method for each transaction in the database the compact datastructure FP-tree is built by adding nodes. Each node has its item_ name,item_count and link to the next node. By calling the function recursively the complete FP-tree is built. Conditional pattern base and prefix trees for each item is built and the tree is mined to find the frequent patterns using CP-Growth method.

4. Results and Discussion

In this section the performance of the proposed algorithm is compared with the existing Apriori, FPGrowth and the hybrid algorithm CanTree. The proposed VSFIM algorithm was implemented in java and tested using NetBeans IDE. The dataset used were Mushroom and chess which are real time datasets available at the FIMI, pattern mining dataset repository. These datasets are also available in UCI datamining repository.

Mushroom Dataset:
The number of transactions: 8416
File size: 598KB

Chess Dataset:
Transactions count from database : 3196
File Size: 338KB

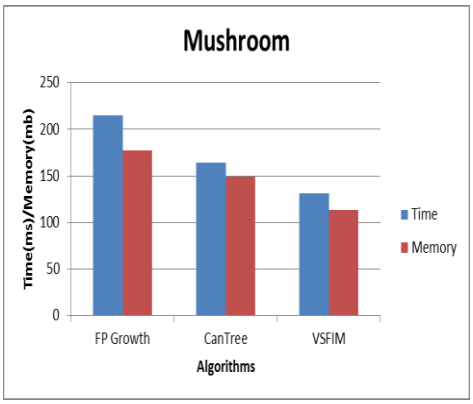

Figure 7: Execution time and memory for mushroom dataset

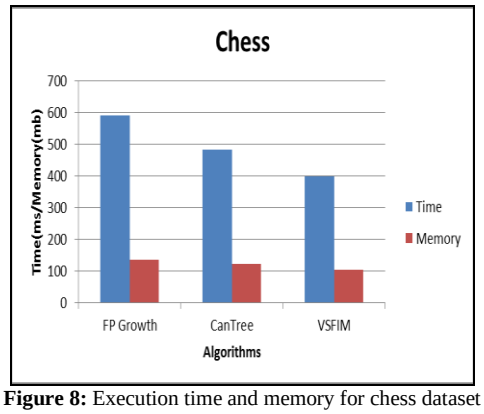

Figure 8: Execution time and memory for chess dataset

Figure -7shows the execution time and memory space occupied for all the 3 algorithms using Mushroom dataset. Figure-8 shows the execution time and memory space occupied for all the 3 algorithms using chess dataset. It is inferred that VSFIM took less time and space to process compared with other algorithms.

5. Conclusion and Future Scope

The algorithm VSFIM proposed in this paper efficiently finds the frequent itemsets. It is proved by the comparisons with the existing FPGrowth and CanTree[12] algorithms. The user can define their ownsupport for each item. This leads to a meaningful mining process. This varied support technique greatly reduces the processing time and space utilized for mining. It eliminates the infrequent items in the beginning of the processing, hence only the frequent items are used for tree building. As the number of frequent items increases, the amount of time taken and the memory also increases. These are directly proportional to one another. As a future work the algorithm can be tested in various dense and sparse datasets and enhanced by incorporating the varied support count technique in building prefix trees.

References

[1] M.S. Chen, J. Han, P.S. Yu, "Data mining: an overview from a database perspective", *IEEE Transactions on Knowledge and Data Engineering*, 1996, 8, pp. 866-883.

[2] J. Han, M. Kamber, *Data Mining: Concepts and Techniques*, Morgan Kaufmann Publisher, San Francisco, CA, USA, 2001.

[3] Jian Pei, Jiawei Han, Hongjun Lu,, ShojiroNishio, Shiwei Tangand DongqingYang,"H-Mine: Fast and space-preserving frequent pattern mining in large databases", IIE Transactions (2007) 39, 593–605

[4] Christian Borgelt,"Simple Algorithms for Frequent Item Set Mining", Advances in Machine Learning II pp 351-369

[5] KaramGoudaEmailauthorMohammed J. Zaki, "GenMax: An Efficient Algorithm for Mining Maximal Frequent Itemsets", Data Mining and Knowledge Discovery November 2005, Volume 11, Issue 3, pp 223–242

[6] Dao-I Lin ; Z.M. Kedem,"Pincer-search: an efficient algorithm for discovering the maximum frequent set", IEEE Transactions on Knowledge and Data Engineering (Volume: 14, Issue: 3, May/Jun 2002)

[7] Tahrima Hashem a, Md. Rezaul Karim a, Md. Samiullah a, Chowdhury Farhan Ahmed, "An Efficient Dynamic Superset Bit-Vector Approach for Mining Frequent Closed Itemsets and their Lattice Structure", Elsevier,September 22, 2016

[8] Huong Bui a , Bay Vo, Ham Nguyen d , Tu-Anh Nguyen-Hoang , Tzung-Pei Hong, "A weighted N-list-based method for mining frequent weighted itemsets", Expert Systems With Applications(2017), https://doi.org/10.1016/j.eswa.2017.10.039https://doi.org/10.1016/

[9] Bay Vo ,Sang Pham, Tuong Le ,Zhi-Hong Deng, "A novel approach for mining maximalfrequent patterns",http://dx.doi.org/10.1016/j.eswa.2016.12.0230957-4174/©2016ElsevierLtd.

[10] Md. RezaulKarima,, Michael Cocheza, , OyaDenizBeyanb, Chowdhury Farhan Ahmed, Stefan Deckera, "Mining Maximal Frequent Patterns in Transactional Databases and DynamicData Streams: a Spark-based Approach", Information Sciences, December 1, 2017

[11] Ramah Sivakumar, J.G.R.Sathiaseelan, "A Performance based Empirical Study of the Frequent Itemset Mining Algorithms", International Conference on Power, Control, Signals and Instrumentation Engineering (ICPCSI-2017), IEEE Page 350-355

[12] Ramah Sivakumar, J.G.R.Sathiaseelan, "A hybrid algorithm for mining frequent itemsets in transactional databases",International Conference on Recent Advances In Computing And Communication 2018, In Print.