International Journal of Advanced Research in Engineering and Technology (IJARET)
Volume 11, Issue 12, December 2020, pp. 3380-3397, Article ID: IJARET_11_12_319
Available online at https://iaeme.com/Home/issue/IJARET?Volume=11&Issue=12
ISSN Print: 0976-6480 and ISSN Online: 0976-6499
DOI: https://doi.org/10.34218/IJARET.11.12.2020.319

WEIGHTED FREQUENT ITEMSET MINING IN TRANSACTIONAL DATABASES USING AN ENHANCED WEIGHT TECHNIQUE

Dr. Ramah Sivakumar
Assistant Professor, Department of Computer Science
Bishop Heber College (Affiliated to Bharathidasan University),
Tiruchirappalli, Tamil Nadu, India

ABSTRACT

Frequent Itemset Mining (FIM) is one of the lime lighted domains in pattern mining. Normally, frequent Itemsets (FI's) are mined based on their occurrences' and not on items' importance. Each item has its own uniqueness. Hence, finding all the frequent itemsets does not lead to a meaningful mining process in certain situations. In real world, each item has its importance based on their values. It is observed that constraint based FIM could be one of the better ways for significant information retrieval. Based on the importance/meaning/value of items, some researchers had proposed algorithms. In this paper, weight constraint is used and an enhanced algorithm for mining Weighted Frequent Itemsets (WFIs) from databases has been proposed. Based on the weight constraint, frequent itemsets FI's are mined efficiently. The datasets used for this research are realtime datasets which are available in the FIMI pattern mining repository. Comparative results indicate that the proposed algorithm outperforms the existing algorithms in an efficient way in terms of time, memory consumption and scalability.

Key words: Frequent itemset mining, weighted itemsets, pattern mining.

Cite this Article: Ramah Sivakumar, Weighted Frequent Itemset Mining in Transactional Databases using an Enhanced Weight Technique, *International Journal of Advanced Research in Engineering and Technology*, 11(12), 2020, pp. 3380-3397.
https://iaeme.com/Home/issue/IJARET?Volume=11&Issue=12

1. INTRODUCTION

Mining all the available frequent itemsets in the database normally leads to large number of FI's. A transactional database contains a number of transactions. Each transaction contains a set of items. Every item in the transaction has its importance based on their value. Hence, giving equal importance to all the itemsets could not be possible for proper information retrieval. For example, the items in a supermarket transaction may have importance based on their price and quality, some words in a text may have importance based on their meaning, and medical treatments have various levels of emergency and so on. For a meaningful mining process

constraint based FIM could be considered. Many constraint based techniques has been proposed by the researchers so far to make frequent itemset mining a meaningful one. Weight constraint is one among them where weights are calculated and assigned to the items based on their values. Weighted databases (WD's) are used in realtime applications such as sales, stock markets, intelligent systems and so on in which itemsets are classified with their weights. Studies have been an ongoing process using WD's in the mining process.

A transaction database, TDB, is a set of transactions in which each transaction, denoted as a tuple <tid, X>, contains a unique tid and a set of items. A pattern is considered as k-pattern if it contains k items. A pattern $\{x_1, x_2,.., x_n\}$ is also represented as $x_1, x_2,.., x_n$. The support of a pattern is that the number of transactions containing the pattern within the database. A weight of an item should be a non-negative real number that reflects the importance of the item within the transaction database. The term, weighted itemset is used to represent a set of weighted items. A weight is given to an item within a weight range, $W_{min} \leq$ Item Weight $\leq W_{max}$. The problem of weighted frequent pattern mining is to find the complete set of patterns satisfying a support constraint and a weight constraint in the database. Many researchers [19] [20] [21] [22] [23] [24] [25] [26] [27] [28] [29] done an analysis of the transactional databases in the recent years.

In this paper, a weight based frequent Itemset mining algorithm *WBFIM* (Weight Based Frequent Itemset Mining with weight range and minimum weight) has been proposed which is based on the pattern growth technique. The weight of the itemset is calculated using an improved technique. The rest of the paper is comprised of review of literature in section 2. In section-3, the proposed algorithm has been defined with example and in section -4 experimental results are discussed. Section – 5 is where the conclusion and future scope of the work is given.

2. REVIEW OF LITERATURE

The method of mining FWI's was first proposed by Ramkumar et.al in 1998[1]. Many approaches have been devised in the mining process. Based on the formula of calculating weighted support values the process was then divided into two distinct approaches. MINWAL(O) and MINWAL(W) algorithm was proposed by Chun Hing Cai et al In 1998[2].

The WARM algorithm proposed by Feng Tao et al [3], inherits G. D. Ramkumar's calculation method and extends the unbound weighted Itemset that belong to [0, 1]. Tao et.al used the arithmetic mean of the weights of an item in a transaction to calculate a transaction weight. The author used the Lattice structure which was widely used in mining data without weights for computational speed up. The limitation in the approach was time-consuming as it scan the database multiple times. Next, an average function was used to calculate the weight of an itemset. The support of an itemset is multiplied by its weight to calculate its weighted support. As the average function is used, WIS may vary according to the updation in processing. WFIM algorithm which inherits the calculation method of Chun Hing Cai was proposed by Unil Yun et al.[4] in 2005. Unil Yun's team has more than 20 research works related to weighted association rule mining. However, these works are all based on the WFIM algorithm.

An upper bound model was proposed by Yun and legget et.al (2006)[5] , in which maximum weight value was adopted as weight upper bound for each transaction. WHIUA algorithm was proposed by Zi-guo Huai et al. in 2011[6]. Still WHIUA algorithm does not satisfy the downward closure property, thereby occupying very large search space and hence not suitable for common pruning strategies. Satisfying the downward closure property, Guo-Cheng Lan et al.[7] proposed the PWA algorithm in 2013. This strategy effectively pruned with upper bound weights of the transactions along with the itemset.

Coenen et.al(2013)[8] proposed a technique called weighted itemset tidset tree(WIT-tree) structure for mining. In this method the database is scanned only once and a diffset strategy was incorporated in the mining process. It has a limitation that it consumes more time in sparse

datasets. Xuyang Wei et al.[9] proposed the IWFPM algorithm in 2015 which effectively pruned and satisfied the downward closure property with the itemset and the upper bound weights of the transactions. To tighten the upper bounds, a sequence maximum weight model was proposed by G.C.lan et.al (2015)[10]. It reduced the number of candidates in mining. A structure named as interval word segment (IWS), proposed by nguyen et al(2016)[11] using bit-vector representation of tidsets to remove unwanted bits in the dataset. The performance of the approach was more for sparse databases and less for dense databases. That is the structure is inefficient in dense databases. Lee et.al (2017) [12] proposed a novel approach using FP-tree without any TIDs. Prefix tree structure with two dimensional array is used in the first one. A new methodology was incorporated in the second one to mine the database efficiently. Time complexity was more in the algorithm as it took long time to scan the tree when the database was large. Hence, this approach consumed more memory and processing time also for the large databases, as the built tree is also large.

Preliminaries

Let I= {i_1, i2, i3,…….i_n} be a set of distinct items and W be the set of non-negative whole numbers. A pair (x, w) is named a weighted item where x∈ I is an item and w∈W is that the weight related to x. A transaction should be a set of weighted items, each of which can appear in multiple transactions with different weights.

Definitions:

Weighted database (WD): It is a tuple(T,I,W), where T={t_1,t_2,t_3,….,T_m} is a set of transactions. I={i_1,i_2,i_3,….i_n} is a set of items. W={w_1,w_2,w_3,….w_n} is a set of corresponding weights of item in I.

Weight settings: Based on the domain the item weight is calculated using the break-even point of the item.

The weight can be classified as Item weight and Itemset weight.

Item weight: The item weight is denoted as *w*(i). The item weight is considered as a function and can be written as *w(i)=f(a),* where a is the weight attribute of certain item.

Itemset Weight: Itemset weight is denoted as *w(is)*. It is calculated based on the weights of the items that constitute the Itemset.

3. METHODOLOGY

Framework of Weight Based Frequent Itemset Mining approach

The introduction of weight constraints in frequent pattern mining has made the process more interesting. Different weights are given to items consistent with their importance or intensity. In contrast to previous frequent pattern mining approaches which are mainly based on support constraints, weighted frequent item mining approaches consider not only the frequency on the other hand the importance of patterns with the concern of downward closure property.

The Proposed Weighted Frequent Itemset Mining algorithm (WBFIM) is an enhanced algorithm where the minimum support, item weights and Itemset weights are calculated and weight ranges are used in the mining process. As the transactional database is considered, the minimum support of each item is calculated from the item's break-even point.

Minimum support of item

Minimum support = BreakEvenPoint of item.

Example: For an online store dataset, the cost of items can be considered for calculating break even point.

BreakEvenPoint = FC / (SP–VC) -----→ (3.1)

Where FC is the fixed cost, SP is the Sales price of the item and VC is its variable cost. The weight of the item is its minimum support. If the frequency of the item is greater than or equal to its minimum support, then the item is considered as frequent.

Minimum support of Itemset

Minimum support of Itemset = sum of weights of the items in the Itemset

WBFIM provides the following contributions:

- In the *WBFIM*, individual items are assigned different weights using the break-even formula, for example it can be the marginal profit of an item and the weighted items within weight range reflect their importance. Weight constraints are pushed into the pattern growth algorithm keeping the downward closure property.
- The number of weighted frequent itemsets can be adjusted by changing parameters such as a weight range and a minimum weight although a *minimum support* is lower in the dense database or long databases.
- In *WBFIM*, the weight and support of each item are considered separately for pruning the search space. *WBFIM* allows the user to balance support and weight of itemsets.
- Extensive performance evaluation and analysis of relationship between a weight and a support are conducted.

WBFIM uses the bottom-up divide and conquer method in mining weighted frequent itemsets. In general, a descending ordered prefix tree and bottom up traversal or ascending ordered prefix tree and top down traversal are used together. However, the proposed algorithm adopts an ascending *weight ordered* prefix tree. The tree is traversed bottom-up since the previous matching cannot maintain the downward closure property. A support of each itemset is usually decreased as the length of an itemset is increased; however the weight has a different characteristic. An itemset which has a low weight sometimes can get a higher weight after adding another item with a higher weight; hence it is not guaranteed to keep the downward closure property. For instance, assume that the *minimum support* is 3, the support of item "A" is 2, and also the support of itemset "AB" is 2, a weight of item "A" is 1 and a weight of item "B" is 2. The weighted support of item "A" is 1 and the weighted support of itemset "AB" is 3. Item "A" cannot be pruned even if the weighted support (1) of item "A" is less than *minimum support* (3) since the weighted support of itemset "AB" is equal to the *minimum support* and itemset "AB" could be considered as weighted frequent itemset. To tackle this problem, frequent prefix trees are constructed by weight ascending order and these trees are traversed in bottom up.

The proposed algorithm is presented in detail with actual examples in order to illustrate the steps in the FP-tree construction and the mining of a weighted frequent itemset from the FP tree.

Definitions of WBFIM

WBFIM algorithm is based on the following definitions.

Definition 1: Item Weight

The assigned weight of items reflects the importance of each item in the transaction database. In this research work, for calculating the weight of each item the break-even point (BEP) formula is used considering the marginal profit of each item in online store dataset. The inputs are I (Item), FC (fixed cost), sales price (SP) and Variable cost (VC). The result derived is called as item weight.

Item Weight = FC / (SP–VC) -----→ (3.2)

The algorithm for finding item weight

Input: I(Item), FC(fixed cost), sales price (SP) and Variable cost (VC)

Output: the set of item weight

//weight calculation for each item

Procedure weight (I, FC, SP, VC)

Begin

BEP ← Ø;

For each $I_i \leq n$ do // (for each item in the transaction DB)

$TI_i \leftarrow$ FC / (SP–VC); // (Weight calculation for each item)

BEP ← BEP + TIi; // (weights storage)

end

Procedure min_max (BEP) // (Finding minimum and maximum weights)

For each item

Compare item weight with weights of other items

Find out the minimum and maximum weight from items' weights

Sort items according to item's weight in ascending order

End

End

For calculating the weighted support of an itemset, the transaction weight (tw) of a transaction t_k is calculated first and it is defined as follows:

$tw(t_k) = \sum i_j \in w_j / |t_k|$ ----------→ 3.3

Weighted support of an itemset *ws*:

$$ws(X) = \frac{\sum_{t_k \in t(X)} tw(t_k)}{\sum_{t_k \in T} tw(t_k)}$$ ----------→ 3.4

Table 1 A transaction database as a running example in *WBFIM*

TID	Items	Frequent Item List
1	A,B,C,E	B,C,E
2	B,C,D	B,C
3	A,B,C,D,E	B,C,E
4	A,B,E	B,E
5	B,C,E	B,C,E
6	B,C,D,E	B,C,E

Table 2 Calculated Item Weight

Item	Item Weight
A	0.5
B	0.2
C	0.8
D	0.4
E	0.3

Table 3 Calculated transaction weight

TID	tw
1	0.45
2	0.46
3	0.44
4	0.33
5	0.43
6	0.42
Sum	2.53

Example: Considering Tables 3.1 and 3.2, tw(t1) value can be computed as follows:

tw(t1) = (0.5+0.2+0.8+0.3)/4 =0.45

Table 7.3 shows all tw values of transactions in Table 3.1.

From Tables 3.1 and 3.3, compute the weighted support (ws) of itemset {BD} value as follows: As {BD} appears in transactions {2, 3, 6}, ws (BD) is computed as:

ws (BD) = ((0.46+0.44+0.42) / 2.53) ≈ 0.52

The mining of FWI requires the identification of all itemsets whose weighted support satisfies an user specified minimum weighted support threshold (minws), that is FWI = {X ⊆ I| ws(X) ≥ minws}.

Definition 3.2 Weight Range (WR)

The calculated weight of an item must be within the weight range, that is, weight range = $W_{min} \leq IW \leq W_{max}$. where W_{min} represents minimum weight, W_{max} represents maximum weight and IW represents item weight.

Table 4 Weight Range and weight assignment

Item (minimum	A	B	C	D	E
Support	3	6	5	3	5
Weight	0.5	0.2	0.8	0.4	0.3

The algorithm for finding available weight ranges and weighted items within given maximum and minimum weights

Input: Weight range (maximum and minimum weights)

Output: available weight ranges and weighted items within weight range

//weight range calculation

Procedure weight_range (max_wt, min_wt)

Derive the available weight ranges within given weight range

Foreach weight range

Sort the weighted items within weight range in ascending order

end

End

Definition 3.3 Minimum weight threshold (min_weight)

In the *WBFIM*, a balance between the two measures of weight and support is maintained. Therefore, minimum weight constraint is defined like a *minimum support* in order to prune items which have lower weights.

Definition 3.4 Maximum Weight (MaxW)

The maximum weight (MaxW) is defined as the value of the maximum weight of item in a transaction database or a conditional database. In *WBFIM*, a MaxW is used in a transaction database.

Definition 3.5 Minimum Weight (MinW)

The minimum weight (MinW) is defined as the value of the minimum weight of item in a transaction database or a conditional database. In *WBFIM*, a MinW is used in a conditional database.

Definition 3.6 Infrequent itemset

An itemset X is called an infrequent itemset if the support of the itemset X is less than a *minimum support* and its weight is also less than a minimum weight.

Definition 3.7 Weighted Frequent Itemset (WFI)

An itemset X is a weighted frequent itemset if, following pruning conditions, condition 7.1 or condition 7.2 below is not satisfied. If the itemset X does not satisfy both of these, then the itemset X is called a weighted frequent itemset.

Pruning Condition 3.1 (support < min_sup && weight < min_weight) The support of an itemset is less than a *minimum support* and the weight of an itemset is less than a minimum weight constraint.

For finding infrequent items, pruning condition 7.1 is applied. When a weight and a support are considered separately, there are four cases for each item: a high support and a high weight, a high support and a low weight, a low support and a high weight and then a low support and a low weight. In definition 7.6, items which have a low support and a low weight are defined as infrequent items. The items which have a low support and a low weight can be pruned as these items have low frequencies and low importance. However, the items having other cases cannot be pruned since these items may have higher priority although the support of the itemset is low or the itemset may have higher frequency even if the weight of the itemset is low.

Pruning condition 3.2 (support * MaxW (MinW) < min_sup) In a transaction database, the value of multiplying itemset's support with a MaxW among items in the transaction database is less than a *minimum support*. In conditional databases, the value of multiplying the support of an itemset with a MinW of a conditional pattern in the FP-trees is less than a *minimum support*.

Lemma 3.1 When two conditions are applied to prune weighted infrequent itemsets, the case in which only pruning condition 3.1, but not pruning condition 3.2, is satisfied for pruning weighted infrequent itemset, is that a MaxW of a transaction database or a MinW of a conditional pattern in the FP-tree should be greater than one.

Proof: In this case, pruning condition 3.1, but not pruning condition 7.2 in the definition 7.7, should be satisfied in order to prune an itemset. That is, in condition 3.1, the support of an itemset is less than a *minimum support* and the weight of an itemset is less than a min_weight. However, the value of multiplying the itemset's support with a MaxW (MinW) of an itemset should be greater than or equal to a *minimum support*. It can be seen that the following two formulas should be satisfied.

Formula 1: support (i) < *minimum support*

Formula 2: support (i) * MaxW (MinW) ≥*minimum support*

It is already known that the a MaxW of a transaction database or a MinW of a conditional pattern in the FP-tree must be greater than or equal to one in order to satisfy both of the formulas. For example, assume that a min support is 5, a minimum weight threshold is 0.8, a support of an itemset is 4, the weight of an itemset is 0.7 and the MaxW of an itemset in TDB

is 1.3. In this case pruning condition 7.1 is satisfied but the pruning condition 7.2 is not satisfied. Therefore, this itemset is pruned by the condition 7.1 in definition 7.7.

Lemma 3.2 There is no limitation to use the pruning condition 3.2 of definition 3.7. That is, the pruning condition 3.2 (support (i) × MaxW (MinW) < minimum support) can be applied.

When only pruning condition 3.2 is satisfied , but not pruning condition 3.1 of definition 3.7 is satisfied to prune weighted infrequent itemsets, a MaxW (Maximum Weight) of a transaction database or a MinW (minimum weight) of a conditional pattern in the FP tree can be any value.

Proof: In this case, an itemset is pruned since pruning condition 7.2 is satisfied although condition 3.1 is not satisfied. We can see that the following two formulas should be satisfied

Formula 3: (support (i) ≥ minimum support || weight ≥ min_weight)

Formula 4: (support (i) × MaxW (MinW) < minimum support)

To satisfy Formula 4, MaxW (MinW) should be less than one if the support of an itemset is greater than or equal to a minimum support in the Formula. However, if a weight of an itemset is greater than or equal to a minimum weight threshold in the formula 3 and the support of an itemset is less than a *minimum support*, there is no relationship between Formula 3 and Formula 4. In other words, pruning condition 3.2 (support * MaxW (MinW) < *minimum support*) can be applied without any limitations.

Lemma 3.3 When two pruning conditions are applied to prune weighted infrequent itemsets, the method to use two pruning conditions, always prunes more than the approach to use only a *minimum support* when a MaxW of the transaction database or a MinW of the conditional pattern in the FP tree is less than one.

Proof: Every item has the same priority in normal frequent itemset mining. No consideration of weights. The weight can be considered as 1.0. If pruning condition 3.2 is only considered, we can understand that more items or itemsets will be pruned when weights of items are set as less than one. For example, assume that a *minimum support* is 4 and the support of an itemset is 5. Normally, the itemset would not be pruned since the weight of all the itemset is 1.0 and the support of the itemset is greater than a *minimum support*. However, the itemset is pruned when the weight of the itemset is 0.7 by condition 3.2 in definition 3.7.

Example 3.2: Table 4 is considered for calculations in this example. The *minimum support* is 3. The calculated weight range is 0.2 – 1.5 (minimum weight is 0.2 and maximum weight is 1.5). By varying the weight ranges and using the pruning techniques discussed the frequent itemsets in each transactions are shown in table 5. As this is shown for example, all the items are taken for consideration assigning different weights in different weight ranges. Actually, the weights of items remain the same. Only those items that come under the defined weight range will be taken for computation.

Table 5 Example set of Items with support, weight ranges and weights

Item (min_sup = 3)	**a**	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**
Support	2	2	4	5	6	5	3	4	1
(0.2 ≤ WR1 ≤0.7)	0.	0.3	0.6	0.4	0.5	0.7	0.2	0.7	0.3
(0.7 ≤ WR2 ≤0.9)	0.	0.75	0.8	0.9	0.85	0.75	0.7	0.9	0.7
(0.7 ≤ WR3 ≤1.3)	1.	1.0	0.9	1.0	1.2	0.7	0.8	1.3	0.9
(1.0 ≤ WR4 ≤1.5)	1.	1.1	1.4	1.2	1.3	1.5	1.0	1.5	1.1

Table 6 Weighted frequent itemsets for different weight ranges

TID	WFI list (0.2 ≤ WR1 ≤0.7) MinW = 0.2	WFI list (0.7 ≤ WR2 ≤0.9) MinW = 0.7	WFI list (0.7 ≤ WR3 ≤1.3) MinW = 0.7	WFI list (1.0 ≤ WR4 ≤1.5) MinW = 1.0
101	d, e, f	c, d, e, f	c, d, e, f	a, c, d, e, f
102	d, e, f	c, d, e, f, h	c, d, e, f, h	a, c, d, e, f, h
103	d, e, f	d, e, f, h	d, e, f, g, h	b, d, e, f, g, h
104	e, f	c, e, f	c, d, e, f, g, h	b, c, e, f, g
105	d, e, f	c, d, e, f, h	d, e, h	c, d, e, f, g, h
106	d, e	d, e, h	c, e, f, g	d, e, h

Example 3.3 This example is to show the difference in FWI by changing a WR and a min_weight. In this example, pruning condition 3.1 in definition 3.7 is applied using WR3 as a weight range. If the min_weight is 1.2, items "a", "b" and "i" are pruned as the support of these items is less than a minimum support and the weight of these items is also less than a minimum weight. In a similar way, the following results are obtained by changing min_weight. If min_weight is 1.1, items "i" and "b" are pruned and if min_weight is 1.0, item "i" is pruned. As a result, the number of weighted frequent items can be changed according to different min_weights. When pruning condition 3.2 of definition 3.7 is considered, if a weight range is 1.0 ≤WR4 ≤1.5, only item "i" is pruned because the value of multiplying item i's support (1) with a MaxW (1.5) is less than a minimum support (3). However, the items "a" and "b" are not pruned because the value of multiplying the support (2) of item "a" and "b" with a MaxW (1.5) is equal to a minimum support (3). In a similar fashion, we can get the following results by changing WRs. If the weight range is 0.7 ≤WR3 ≤1.3, item "a", "b", and "i" are pruned. If the weight range is 0.7 ≤WR2 ≤ 0.9, item "a", "b", "i" and "g" are pruned and if the weight range is 0.2 ≤ WR1 ≤ 0.7, item "a", "b", "c", "i", "g" and "h" are pruned. Thus, the number of weighted frequent items can be adjusted by using the different WRs.

While applying WR2, item g's support is 3, MaxW is 0.9 and the value (2.7) of multiplying item's support (3) with a MaxW (0.9) of an itemset in the TDB is less than minimum support (3) so item "g" can be removed. Meanwhile, the number of WFI can be increased when WR4 is used as the weight range. The support of item "a" in the transaction database is 2. However, the value (3) of multiplying the support (2) of item "a" with a MaxW (1.5) of an itemset is equal to a minimum support (3) so this item is added in the WFI list. In the similar fashion the results in Table 5 can be obtained by changing the WRs.

The complete weighted frequent itemset tree structure

Table 7 WFI after pruning and sorting with a WR: 0.7–0.9 in *WBFIM*

TID	weight ascending order item list (item: weight)
100	(f: 0.75) (c: 0.8) (e: 0.85) (d: 0.9)
200	(f: 0.75) (c: 0.8) (e: 0.85) (d: 0.9) (h: 0.9)
300	(f: 0.75) (e:0.85) (d: 0.9) (h: 0.9)
400	(f: 0.75) (c: 0.8) (e: 0.85)
500	(f: 0.75) (c: 0.8) (e: 0.85) (d: 0.9) (h: 0.9)
600	(e: 0.85) (d: 0.9) (h: 0.9)

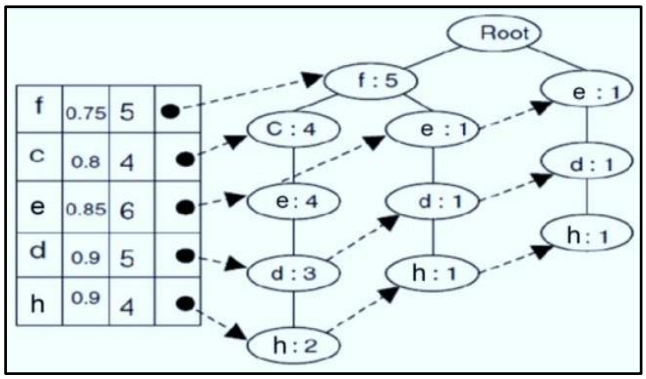


Figure 1 The complete FP tree in *WBFIM*.

The FP-trees in the proposed algorithm are constructed as follows. Scan the transaction database one time and count the support of each item and calculate the weight of each item. After this, sort the items in weight ascending order. Although supports of items may be lower than the *minimum support* and infrequent, the items cannot be deleted since infrequent items may become weighted itemsets in the next step. The weighted infrequent items are removed according to pruning conditions 3.1 and 3.2 in definition 3.7. For instance, assume that WR2 is used as a WR, *minimum support* is 3 and min_weight is 0.7. Then, items "a", "b", "g", and "i" are removed. Table 3.7 shows the result of removing weighted infrequent itemsets and sorting them by weight ascending order. When an item is inserted in the FP-tree, as already discussed, a weighted infrequent item is removed and the rest, weighted frequent items and itemsets, are sorted by weight ascending order. Each node in the FP-tree has item-id, a weight, count and node link. Separate header tables exist for each FP tree and there is an entry for each item in the header table. Figure 3.1 presents the complete FP-tree and corresponding header table for this example in *WBFIM*.

Bottom up divide and Conquer Approach

After a complete FP-tree is constructed from the transaction database, *WBFIM* mines frequent itemsets from the FP-tree. The weighted frequent itemsets are generated by adding items one by one. The proposed algorithm adapts divide and conquer approach for mining weighted frequent itemsets. It divides mining the FP-tree into mining smaller FP trees.

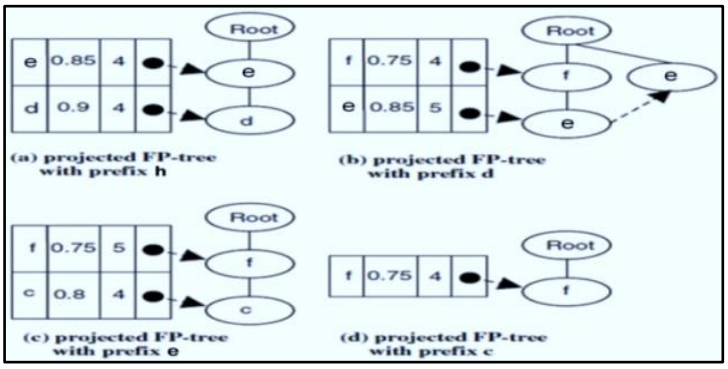


Figure 2 Conditional FP trees in *WBFIM*

WBFIM Algorithm

WBFIM can push weight constraints into the pattern growth algorithm and show how to keep the downward closure property. The items' weights are calculated by using the weight procedure. A weight range and a minimum weight are defined and the items that falls within

the range are considered for the mining process which leads to constraint based technique. Here, the weighted frequent itemset mining process is summarized and presented below.

ALGORITHM [WBFIM]: Weight Based Frequent Itemset Mining with a weight range and a minimum weight constraint in a large transaction database.

Input:

- A transaction database: TDB,
- A minimum support threshold: *minimum support*
- Weights of the items within weight range: w_i
- A minimum weight threshold: min_weight

Output: The complete set of weighted frequent itemsets.

Begin

1. Scan TDB once and calculate the weight and weight range of items using the proposed *weight* procedure and find the complete weighted frequent items satisfying the following definition: An itemset X is a weighted frequent itemset if the following pruning conditions 1.1 and 1.2 are not satisfied.

Condition 1.1: (support (i) < *minimum support* && weight (i) < min_weight)

Condition 1.2: (support (i) * MaxW < *minimum support*)

2. Sort the items in weight ascending order. The sorted weighted frequent item list forms the weight_order and header table of FP tree.

3. Scan the TDB again and build a complete FP-tree using weight_order.

4. Mine a complete FP-tree for weighted frequent itemset mining in a bottom up manner.

Build conditional databases for all remaining items in weight_list and complete local weighted frequent itemsets for the conditional databases. (An itemset X is a weighted frequent itemset if the following pruning conditions 4.1 and 4.2 are not satisfied).

Condition 1.3: (support (i) < *minimum support* && weight (i) < min_weight)

Condition 1.4: (support (i)* MinW < *minimum support*)

5. When all the items in the overall header table have been mined, WBFIM Stops.

End;

Performance Evaluation

In this section, the performance study over a real-time dataset and a synthetic dataset is presented. The existing weight based mining algorithms are considered for evaluation process. However, WAR [Wang et.al 2000] does not consider a weight measure in mining frequent patterns. After getting these patterns, WAR considers a weight measure to generate weighted association rules. WARM [Tao et.al 2003] uses a weight measure but does not use a support measure. The main two improvement of *WBFIM* are:

- New weight calculation procedure, a weight range and a minimum weight.
- Use of a pattern growth method for weighted frequent pattern mining. In order to evaluate performance, WBFIM is compared with MINWAL and FP-growth algorithms. MINWAL is weighted frequent pattern mining algorithm which uses k-support upper-bound to maintain downward closure property and FP-growth is the first algorithm based on pattern growth approach in mining frequent patterns.

Subsequently, the efficiency of the weight range and the minimum weight is analyzed. The impact of weight range and the minimum weight over the number of weighted frequent itemset and runtime are discussed. Finally, the scalability of *WBFIM* is analyzed against the number of transactions in the datasets.

Experimental Setup

In the experimental study, two real datasets and one synthetic dataset is used. T10I4Dx is a synthetic dataset. The synthetic datasets were generated from the IBM dataset generator which is available in FIMI repository. T10I4Dx dataset is very sparse and contains 10,00,000 transactions. The real-time dataset is a transaction set collected from a shopping mall. It is a very dense dataset and includes transaction ids' and purchased products. It is preprocessed and each item is assigned a number which favored the evaluation procedure. The Mushroom and chess datasets are dense and they can be obtained from the UCI machine learning repository (http://www.ics.uci.edu/~mlearn) and the frequent itemset mining dataset repository (http://fimi.cs.helsinki.fi/data/).

The implementation of the algorithm is done in java and executed using NetBeans IDE. Weight range, *minimum weight* and *minimum support* are set up as the cut off values for weighted frequent itemsets.

Dataset Characteristics

Table 8 Characteristics of real and synthetic datasets

Data sets	Number of Transactions	Number of Items	Average number of items per transaction
Real-time	60K	170	45
Mushroom	8124	120	23
Chess	3196	75	37
T10I4Dx	1000K	1000	10

Measuring the performance based on Processing time and number of generated itemsets

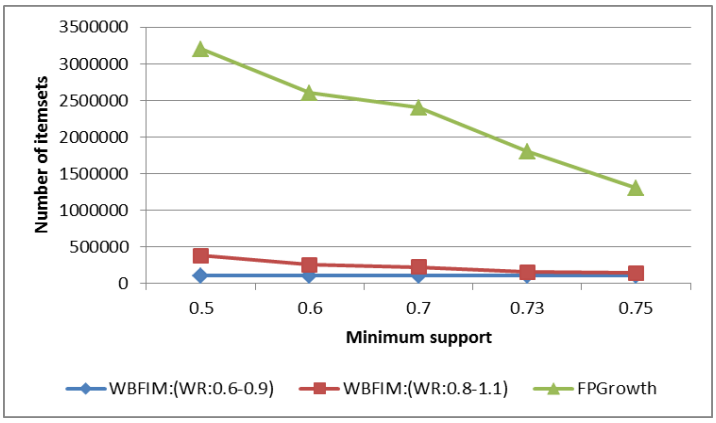


Figure 3 Number of itemsets generated in *WBFIM* (Real-Time dataset).

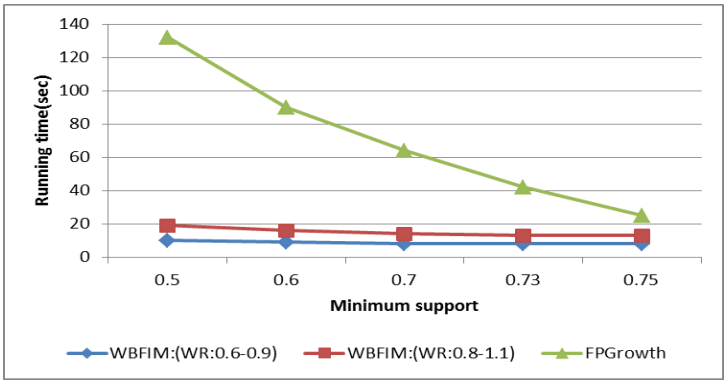


Figure 4 Processing time of *WBFIM* (Real-time dataset).

The experiment results show that in most cases, *WBFIM* outperforms, FP-growth algorithm. First, from the performance results of the Real-time dataset, it is observed *WBFIM* generate fewer itemsets and run faster than FP-growth. Specifically, the number of itemsets generated is less as the weight range is increased. The number of itemsets discovered by *WBFIM* is several orders of magnitude fewer than the number of patterns found by FP-growth.

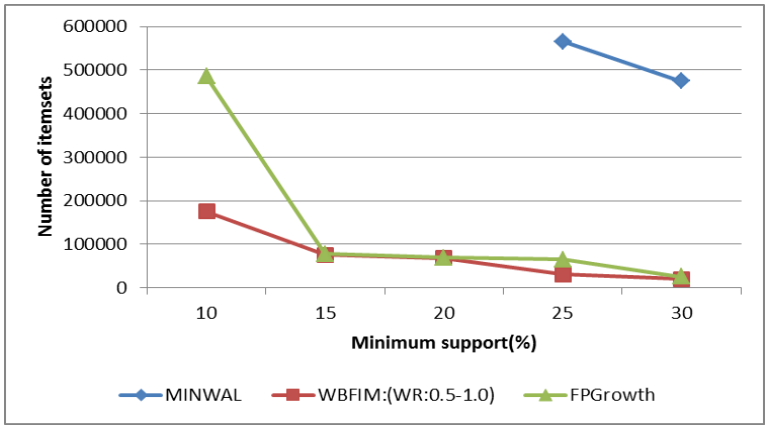
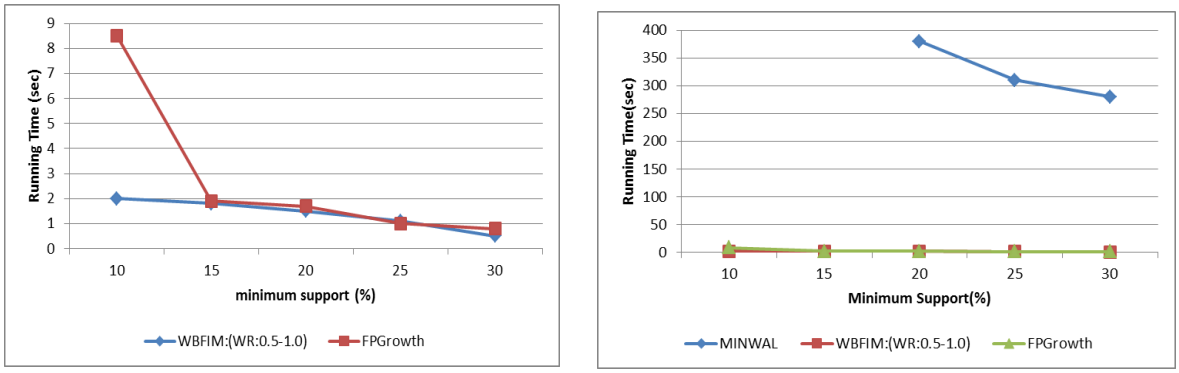


Figure 5 Number of itemsets generated in *WBFIM* (Mushroom dataset)



(a) (b)

Figure 6 Processing time of *WBFIM* (Mushroom dataset)

Figure 3.6a and Figure 3.6b demonstrate the results of performance test using Mushroom dataset by setting weight range from 0.5 to 1.0. *WBFIM* outperforms MINWAL, and FP-growth. When the support threshold is lowered, the performance difference becomes bigger. Note that in Figure 3.5, the number of itemsets in *WBFIM* increases as the *minimum support* is

decreased. However, the number of patterns in MINWAL and FP-growth is substantially increased.

Performance comparison based on memory consumption

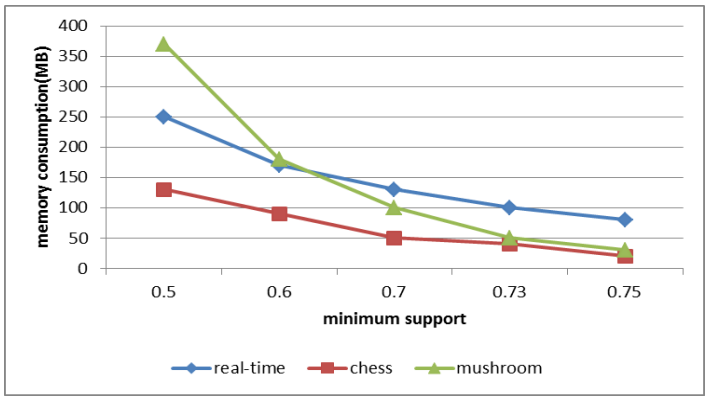


Figure 7 Memory consumption in *WBFIM* (*Weight range* = 0.5-1.0).

Figure 3.7 show the memory usage of the *WBFIM* algorithm when running on the experimental databases. It could be noticed that *WBFIM* used the least memory on most databases such as Chess, Mushroom, and real-time. However, on very sparse and large databases, *WBFIM* used more memory when the threshold was set small. *WBFIM* adopted a good data compression strategy based on the FP-tree structure. Only the space needed for mining is taken for consideration. Hence the amount of memory consumed is reduced according to the minimum support values. Therefore, *WBFIM* used memory efficiently on most experimental databases, and was only inferior to some others on very sparse and large databases like foodmart.

Scalability of WBFIM

To test the scalability with the number of transactions, the T10I4DxK dataset is used. WBFIM is compared with MINWAL, and FP-growth. In Figure 3.7 and Figure 3.8, both WBFIM and FP-growth show linear scalability with the number of transactions from 100k to 1000k. However, WBFIM is much more scalable than the others.

First, scalability of WBFIM is tested with regard to the number of transactions from 100K to 1000K. Different *minimum support*s ranging from 0.2% to 0.5%. and weight range as 0.2 to 0.8 on the T10I4Dx synthetic dataset.

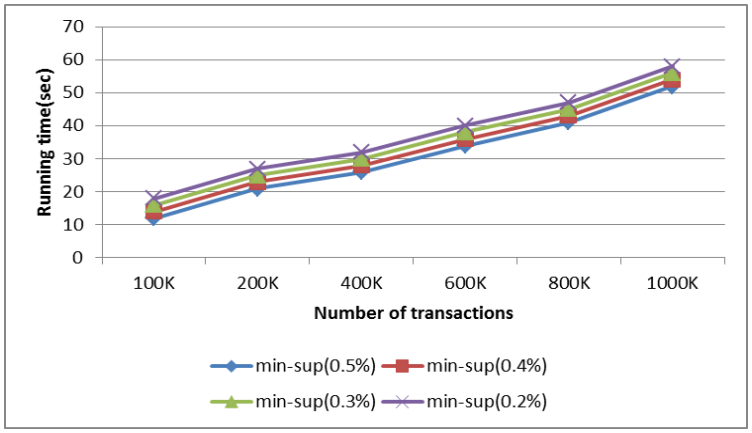
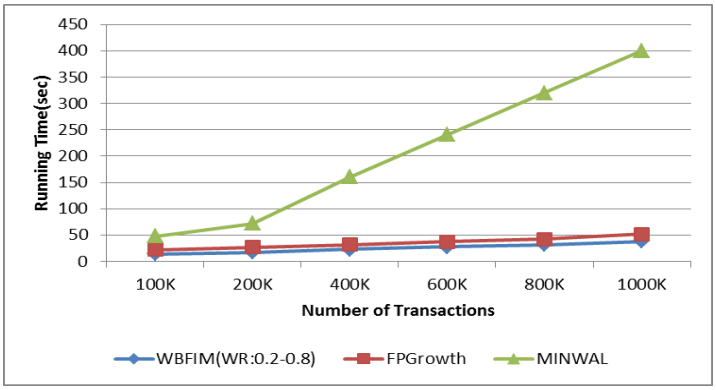
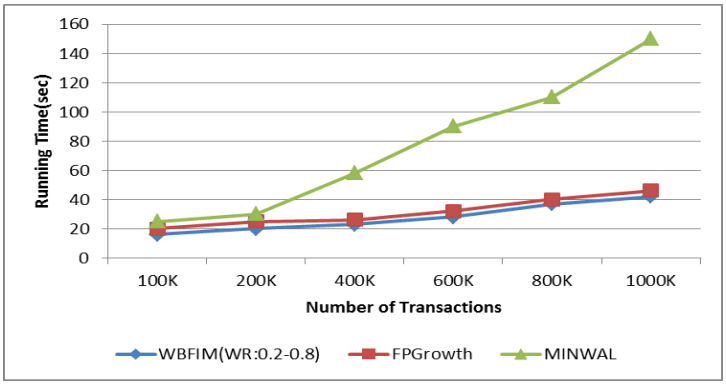


Figure 8 Scalability test in *WBFIM* (T10I4DxK dataset, WR: 0.2-0.8)



(a)(T10I4DxK dataset, *Minimum support* = 0.1%).



(b)(T10I4DxK dataset, *Minimum support* = 0.4%).

Figure 9 Scalability of *WBFIM* compared with other algorithms and different supports.

In Figure 8, it is inferred that *WBFIM* has good scalability in terms of number of transactions and becomes better as the *minimum support* is increased. Subsequently, *WBFIM* is compared with MINWAL, and FP-growth. The *minimum support* is set as 0.1% in Figure 9a and 0.4% in Figure 9b, and a weight range from 0.2 to 0.8. It is also observed that *WBFIM* has much better scalability in terms of base size. The slope ratio of *WBFIM* in both *minimum supports* is lower than other algorithms. In comparison with other algorithms, *WBFIM* not only runs faster, it also has much better scalability in terms of base size.

Impact of minimum weight in WBFIM

Table 8 Impact of *minimum weight* in WBFIM

Minimum Support of Real-time dataset	**Number of WFI WR: 0.5 –1.5 MW: 1.5**	**Number of WFI WR: 0.5 – 1.5 MW: 1.0**	**Number of WFI WR: 0.5 – 1.5 MW: 0.5**	**Number of FI**
57000(95%)	115	714	1271	2015
54000(90%)	630	2046	5112	25126
48000(80%)	2260	2680	2944	513972
42000(70%)	3497	3982	3993	4029811

Table 8 lists the number of Weighted Frequent Itemsets (WFI) with various minimum weights, Frequent Itemset (FI). It is understood from Table 7 that, *WBFIM* can generate smaller WFI by using different Minimum Weight (MW) thresholds. For example, in Table 3.7, the

number of WFI at a minimum support: 90%, WR: 0.5 – 1.5 and a min_weight: 0.5, is 5112. However, the number of WFI can be reduced to 2046 with a min_weight: 1.0 and can be further reduced to a 630 with a min_weight: 1.5. The numbers of frequent itemsets are 25126. In this way, the proper number of weighted frequent itemsets can be found by adjusting the minimum weight.

4. SUMMARY AND CONCLUSION

Many studies on frequent pattern mining have been conducted in the last decade. One of the main limitations of the traditional method for mining frequent patterns is that all items are treated uniformly, while real items have different characteristics. For this reason, weighted frequent pattern mining algorithms have been studied. Mining the complete set of patterns often suffers from generating a very large number of patterns and association rules. Although frequent closed/maximal patterns mining algorithms are suggested, they, in large databases, still generate too many patterns when support is low or the pattern becomes long.

An efficient and scalable frequent pattern mining algorithm is suggested with weight constraints. The main approach is to push the weight constraints into the pattern growth algorithm while maintaining the downward closure property. *WBFIM* focuses on weighted frequent pattern mining based on a pattern growth algorithm. A weight range and a minimum weight constraint are defined and items are given different weights within the weight range. Most of the existing weighted frequent itemset mining algorithms are based on Apriori approach. However, *WBFIM* uses divide and conquer approach and maintains downward closure property. From the experimental study it is observed that *WBFIM* is faster than MINWAL, and FPgrowth algorithms. Additionally, it generates fewer frequent itemsets for larger databases even with a very low *minimum support*. The extensive performance analysis shows that *WBFIM* is efficient and scalable in weighted frequent itemset mining.

REFERENCES

[1] G. D. Ramkumar, S. Ranka, S. Tsur, "Weighted Association Rules: Model and Algorithm", *Proc. ACM SIGKDD*, 1998, 1-13.

[2] C. H. Cai, A. W. C. Fu, C. H. Cheng, W. W. Kwong, "Mining association rules with weighted items", *Proceedings. IDEAS'98. International Database Engineering and Applications Symposium (Cat. No.98EX156)*, Cardiff, UK, 68-77, 1998.

[3] F. Tao, F. Murtagh, M. M. Farid, "Weighted association rule mining using weighted support and significance framework", SIGKDD003, 661-666, 2003.

[4] U. Yun, J. J. Leggett, "WFIM: Weighted Frequent Itemset Mining with a weight range and a minimum weight", *SDM 2005*, 636-640, 2005.

[5] U. Yun, J. J. Leggett, "WIP: Mining Weighted Interesting Patterns with a strong weight and/or support affinity", Sixth SIAM International Conference on Data Mining, April 20-22, 2006, Bethesda, MD, USA

[6] Z. G. Huai, M. H. Huang, "A weighted frequent itemsets Incremental Updating Algorithm base on hash table", *2011 IEEE 3rd International Conference on Communication Software and Networks*, Xi'an, 201-204, 2011.

[7] G. C. Lan, T. P. Hong, H. Y. Lee, and C. W. Lin, "Mining Weighted Frequent Itemsets", *in Proceedings of the 30th workshop on Combinatorial Mathematics and Computation Theory (Alg*030*)*, 85-89, 2013.

[8] B. Vo, F. Coenen, B. Le, "A new method for mining Frequent Weighted Itemsets based on WIT-trees", *Expert Systems with Applications*, 40(4), 2013, 1256-1264.

[9] X. Wei, Z. Li, T. Zhou, H. Zhang, G. Yang, "IWFPM: Interested Weighted Frequent Pattern Mining with Multiple Supports", *Journal of Software*, 10, 2015, 9-19.

[10] G. C. Lan, T. P. Hong, H. Y. Lee, C. W. Lin, "Tightening upper bounds for mining weighted frequent itemsets", *Intell.Data Anal*, 19(2), 2015, 413-429.

[11] H. Nguyen, B. Vo, M. Nguyen, W. Pedrycz, "An efficient algorithm for mining frequent weighted itemsets using interval word segments", *Appl. Intell*, 45(4), 2016, 1008-1020.

[12] G. Lee, U. Yun, K. H. Ryu, "Mining Frequent Weighted Itemsets without Storing Transaction IDs and Generating Candidates", *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 25(1), 2017, 111-144.

[13] B. Vo, "An Efficient Method for Mining Frequent Weighted Closed Itemsets from Weighted Item Transaction Databases", J. Inf. Sci. Eng., 33(1), 2017, 199-216.

[14] H. Bui, B. Vo, H. Nguyen, T. A. N. Hoang, T. P. Hong, "A weighted N-list-based method for mining frequent weighted itemsets", Expert Syst. Appl., 96, 2018, 388-405.

[15] A. B. Cengiz, K. U. Birant, D. Birant, "Analysis of Pre-Weighted and Post-Weighted association Rule Mining", Innovations in Intelligent Systems and Applications Conference, Izmir, Turkey, 1-5, 2019.

[16] U. Dewan, C. F. Ahmed, C. K. Leung, R. A. Rizvee, D.Deng, J. Souza, "An efficient approach for mining weighted frequent patterns with dynamic weights", ICDM 2019, 13-27, 2019.

[17] B. Vo, H. Bui, T. Vo, T. Le, "Mining top-rank-k frequent weighted itemsets using WN-list structures and an early pruning strategy", Knowledge-Based Systems, 201-202, 2020, 106064-106075.

[18] H. Bui, B. Vo, T. A. N. Hoang, U. Yun, "Mining frequent weighted closed itemsets using the WN-list structure and an early pruning strategy", Appl Intell, 51(3), 2021, 1439-1459.

[19] Kalaiarasi, K., and R. Gopinath. "Stochastic Lead Time Reduction For Replenishment Python-Based Fuzzy Inventory Order Eoq Model With Machine Learning Support." *Technology (IJARET)* 11.10 (2020): 1982-1991.

[20] Kalaiarasi, K., and R. Gopinath. " Fuzzy Inventory Eoq Optimization Mathematical Model." *International Journal of Electrical Engineering and Technology (IJARET)* 11.8 (2020): 169-174.

[21] Priyadharshini, D., R. Gopinath, and T. S. Poornappriya. "A Fuzzy Mcdm Approach For Measuring The Business Impact Of Employee Selection." *International Journal of Management (IJM)* 11.7 (2020): 1769-1775.

[22] V Upendran, and R. Gopinath, "Feature Selection Based On Multicriteria Decision Making For Intrusion Detection System", *International Journal of Electrical Engineering and Technology*, 11.5 (2020): 217-226.

[23] R. Gopinath, A. Chitra, R. Kalpana. "Emotional Intelligence And Knowledge Management- A Relationship Study", *International Journal of Advanced Research in Engineering and Technology*, 11.11 (2020): 2363-2372.

[24] M. Subhashini, and R. Gopinath. "Employee Attrition Prediction In Industry Using Machine Learning Techniques", *International Journal of Advanced Research in Engineering and Technology*, 11.12 (2020): 3329-3341.

[25] S. Rethinavalli, and R. Gopinath. "Botnet Attack Detection In Internet Of Things Using Optimization Techniques", *International Journal of Electrical Engineering and Technology*, 11.10 (2020): 412-420.

[26] S. Rethinavalli, and R. Gopinath. "Classification Approach-Based Sybil Node Detection In Mobile Ad Hoc Networks", *International Journal of Advanced Research in Engineering and Technology*, 11.12 (2020): 3348-3356.

[27] T.S. Poornappriya., and R. Gopinath. "Application Of Machine Learning Techniques For Improving Learning Disabilities", *International Journal of Electrical Engineering and Technology,* 11.10 (2020): 403-411.

[28] T.S. Poornappriya., and R. Gopinath. "Rice Plant Disease Identification Using Artificial Intelligence Approaches", *International Journal of Electrical Engineering and Technology,* 11.10 (2020): 392-402.

[29] R. Gopinath., and T.S. Poornappriya. "An Analysis Of Human Resource Development Practices In Small Scale Startups", *International Journal of Advanced Research in Engineering and Technology,* 11.11 (2020): 2475-2483.