computing@computingonline.net
www.computingonline.net

Print ISSN 1727-6209
On-line ISSN 2312-5381
International Journal of Computing

A FRAMEWORK FOR INCREMENTAL PARALLEL MINING OF INTERESTING ASSOCIATION PATTERNS FOR BIG DATA

Ahmed Sultan Alhegami [1], Hussein Alkhader Alsaeedi [2]

[1] University of Sana'a, Yemen, e-mail: Alhegami@su.edu.ye
[2] University of Science and Technology, Yemen, e-mail: h.alkhadher@ust.edu

Paper history:
Received 12 August 2019
Received in revised form 13 December 2020
Accepted 13 February 2020
Available online 31 March 2020

Keywords:
Big Data Mining;
Association Pattern Mining;
Parallel Mining;
Incremental Mining;
Interesting;
Measure Novelty Measure;
KDD.

Abstract: Association rule mining plays a very important role in the distributed environment for Big Data analysis. The massive volume of data creates imminent needs to design novel, parallel and incremental algorithms for the association rule mining in order to handle Big Data. In this paper, a framework is proposed for incremental parallel interesting association rule mining algorithm for Big Data. The proposed framework incorporates interestingness measures during the process of mining. The proposed framework works to process the incremental data, which usually comes at different times, the user's important knowledge is explored by processing of new data only, without having to return from scratch. One of the main features of this framework is to consider the user domain knowledge, which is monotonically increased. The model that incorporates the users' belief during the extraction of patterns is attractive, effective and efficient. The proposed framework is implemented on public datasets as well as it is evaluated based on the interesting results that are found.

1. INTRODUCTION

Recent advances in digital data collection and data acquisition technologies have opened new avenues to acquire and store increasingly massive volumes of data. This rapid growth of data leads to several considerable issues such as storage, security, scalability, and extraction of interesting knowledge which are difficult to handle using conventional techniques, methods, and tools. Data is useful only if it can be interpreted, analyzed and if a conclusion can be drawn from them [1-3].

Big Data mining refers to finding extraction techniques that are performed on Big Data. Big Data extracts and retrieves interesting patterns from a massive volume of data [4]. Association rule mining plays a very important role in a distributed environment in Big Data analysis [5].

Although many efficient algorithms have been developed to extract association rules, traditional algorithms do not work well on Big Data [6-9]. The main drawbacks with such algorithms are that they don't consider the data size and the time when the data arrives and therefore build a model in batch manner. In contrast, incremental algorithm constructs and refines the model as long as new data arrives at different times [10-13].

The aim of this paper is to propose a framework for incremental parallel mining of interesting association rules for big data. One of the main advantages of the proposed framework is to handle the time changing big data and user domain knowledge. This is useful when many datasets arrive at different times or from a distributed environment. Certainly, it is desirable to update the discovered patterns each time new data arrives. The incremental and parallel nature of the proposed framework makes it valuable to extract interesting patterns at a current time with regard to the previously discovered patterns, more willingly than comprehensively extracting all patterns.

The parallel and incremental association rules algorithms that incorporate the users' domain knowledge during the extraction of patterns are attractive, effective and efficient for the knowledge discovery in database (KDD) process.

2. RELATED WORKS

Frequent itemsets mining algorithms are such as Apriori method [14] and Tree method [15]. Also Parallel frequent itemsets mining algorithms are based on Apriori methods [14] such as in [6-, 7, 16]. They are categorized as count distribution (e.g., parallel data mining (PDM) [6], fast parallel mining [7]), and data distribution (DD) [17]. The assumptions of these approaches are that each processor of a parallel system calculates the local support counts of all candidate itemsets. Then, all processors compute the total support counts of the candidates by exchanging the local support counts. Other parallel frequent itemsets mining algorithms are based on Tree methods [15]. For example, Parallel FP-Growth algorithm (PFP-Growth) which is based on the clustered system [18], load balanced parallel FP Growth algorithm [19], an efficient parallel algorithm using message passing interface on a shared-nothing multiprocessor system [9], and Parallel FP-Growth algorithm to mining frequent patterns [8]. PFP algorithm makes use of the MapReduce parallel programming model for the purpose of analysis and mining of data [8, 20]. It splits the database into small chunks and then uses the MapReduce in three phases to count values, group items, and build tree, and eventually integrates as well as combines the results of the previous phases. The main drawbacks of PFP-Growth are that it does not work on an incremental database and doesn't use any subjective measure of interestingness. Many works have been conducted for developing algorithms based on mining incremental association rules [21-25]. The main hypothesis of these algorithms is to update the discovered model when new data stream arrives. In [24], DEMON algorithm is proposed to handle the evolving data more effectively and efficiently. In [25], DELI algorithm is proposed for monitoring the environment changes of the data stream. It makes use of statistical methods for the updating process. DELI algorithm uses a sampling method to estimate the support counts using an approximate upper/lower bounds on the number of changes in the newly discovred association rules. As the low bound gets smaller, the changes of the association rules get smaller, therefore the model maintenance is not required. Although these algorithms are incremental, they don't reuse the previously discovered knowledge when new data arrive at new time instance. In [22, 23] a Fast UPdate (FUP) algorithm is proposed which is incremental in nature for mining association rules in huge databases. It works by scanning the database to verify whether there are large itemsets or not. FUP algorithm is proposed to compute the large itemsets in the updated database. The main purpose of this algorithm is to solve the efficient update issue of association patterns in the updated database. The algorithm is extended to FUP* and FUP2 that scan the database kth time. In [26] Paralle incremental FP-Growth (PIFP-Growth) is proposed for improvement PFP algorithm [8] to solve the problem of an incremental database. PIFP Growth is based on MapReduce [20] for parallelized incremental mining. The drawbacks of these algorithms are the following ones: they have many stages that are time consuming and perform MapReduce several times, for instance, PFP uses MapReduce in three stages out of seven stages while PIFP uses MapReduce in four stages out of seven stages. In addition, both algorithms don't use any subjective measure of interestingness. The novelty measure of discovered patterns is proposed in [10-12]. It is quantified with respect to known knowledge and it eliminates the patterns that are not interesting from the user's point of view. In our work, we take advantage of the novelty measure of interestingness proposed in [10-12]. Although PFP and PIFP are proposed to deal with parallel and incremental Big Data mining, both approaches are based on traditional FP Growth [15] and make use of MapReduce programming model. Our framework can use any frequent pattern mining algorithm which uses MapReduce. It is similar to PFP and PIFP as it uses MapReduce to achieve parallelism but it differs from PIFP in its incremental manner. The major differences between the proposed framework and PFP and PIFP are:

- PFP uses MapReduce in three out of its five stages and PIFP uses MapReduce in four out of its seven stages while the proposed framework uses MapReduce only twice out of its four stages.
- Both PFP and PIFP don't consider the previous, discovered patterns when new data arrives while the framework updates the model with novel patterns as new data stream arrives.
- The PIFP resets the threshold value as new data arrives and updates the old local tree while our approach constructs different local tree as new data arrives and generates new frequent items.
- To achieve parallelism, PFP and PIFP divide up items into groups and perform Generating group dependent transactions to build trees and extract frequent items while the framework uses MapReduce to construct trees directly from transactions after pruning the infrequent items that don't meet the minimum support criterion.

Even though PFP and PIFP are based on FP-Growth which includes two steps, the framework adds extra steps in order to update the model as new data arrives and guarantees that the discovered patterns are interesting.

The rest of the paper is organized as follows. In section 3, we present the problem statement. A Framework for Incremental Parallel Mining of Interesting Association Patterns is presented in section 4. In section 5 a detailed example is illustrated. In section 6, the experimental results are presented and the conclusion is given in section 7.

3. PROBLEM STATEMENT

At time instance t_i, an incremental Big Data D_i, $i \in \{1,\ldots,n\}$, is collected. Suppose, D_i is partitioned into m parts, where $dp \subset D$. Each $dP_j\ where\ 1 \leq j \leq m$ is saved on processor P_j, and F-List is generated to construct local trees FP-Tree$_m$. Subsequently, frequent items are extracted and association rules are generated to form model T_i.

Let M_i and M_{i+1} be two models discovered at time instances t_i and t_{i+1} from datasets $\bigcup_{j=1}^{i} D_j$ and $\bigcup_{j=1}^{i+1} D_j$ respectively. The objective is to update M_i to M_{i+1} using D_{i+1} and M_i. M_i is the model discovered at time t_i now represents the previously discovered knowledge (PDK). M_{i+1} is the up-to-date model obtained by adding interesting patterns discovered from D_{i+1}. This is achieved by constructing a model $\check{T}_{i+1}$ from D_{i+1} such that association patterns in $\check{T}_{i+1}$ have user specified degree of interestingness with respect to the rules in M_i. Subsequently, $\check{T}_{i+1}$ is used to update M_i to M_{i+1}.

4. A FRAMEWORK FOR INCREMENTAL PARALLEL MINING OF INTEREST

In this paper, we present a framework that efficiently discovers interesting patterns from Big Data. It makes use of MapReduce [20] to deal with data in a parallel manner. Our proposed framework is similar to the PFP [8] algorithm except that each rule generated from frequent itemset list in PFP may not be interesting. At time T_i, our framework computes the novelty aspect of interestingness measure with respect to the existing model MT_i and pruning uninteresting patterns that are not significant in the current data set. The framework is shown in Fig. 1. It comprises 3 phases namely, building local tree, finding frequent itemset, and building incremental interesting model. These phases are explained in the following subsections:

4.1 BUILDING LOCAL TREES

In this phase, Big Data is divided into m small parts, where m can be set manually, among P processors using the MapReduce parallel programming model for the purpose of analyzing and mining data. Each P MapReduce first, reads each small part to achieve parallel count and the integrated count results into a frequent list called F-List, then, it sorts the items of F-List in descending order. Finally, MapReduce performs the second iteration to read each small part and build a local FP -Tree. The phase outputs are FP-Tree$_m$. The following steps are required to build local trees and the algorithm is presented in Algorithm 1.

1. **Set** m, Define and Clear *F – List*.
2. **Division** of D_i into m parts where $1 \leq j \leq m$, and save each dp_j on different processor called P.
3. **First scan** each Transaction T into p_j to compute supports for all items in parallel manner.
4. **Integrate** the count results into *F-List*.
5. **Sort** items of *F-List* in descending order.
6. **Second scan** each Transaction T into p_j:
   - **Sort** items in descending order of T based on *F- List*.
   - **Building** the local tree called $localTree_j$ as algorithm *FP-Tree* in [15].
7. **Return** $localTree_j$

Algorithm 1: Building Local Trees

Procedure: **Building LocalTrees** (D_i,σ)
Set of m;
Define and clear F-List : F [];
Division of D_i into $dP_j\ where\ 1 \leq j \leq m$
Send $dP_j$to $P_j$different
In each P_j
F []←**Mapper**(dP_j)
Sort F [];
Call Reduce(dP_j,F[],σ)
Procedure: Mapper(dP_j);
{
Define and clear $f - listLocal = fL_j$[]
foreach Transactions T in dP do
    foreach Item a in T do
    fL_j [a] ++;
    end
end
return fL_j;
}
Procedure: Reduce(dP_j,F[])
{
Define and clear the root of $localTree_j$**: r;**
foreach Transactions T in dP_jdo
Make T**ordered according to F ;**
Call Construct Tree(T**; r);**
end
Call **Finding Frequent itemset(r,**F[])
}

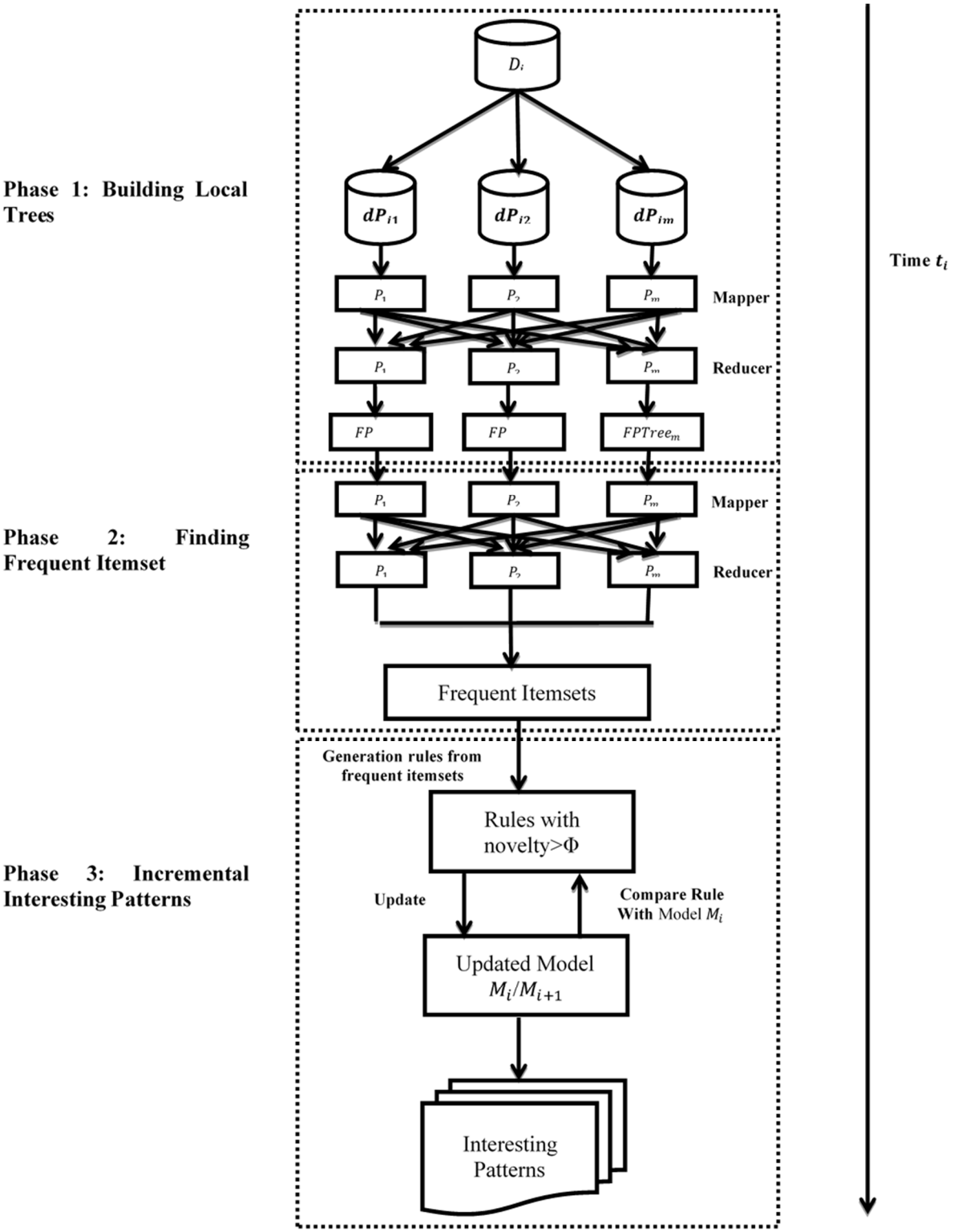


Figure 1 – The Framework for Incremental Parallel Mining of Interesting Association Patterns for Big Data

4.2 FINDING FREQUENT ITEMSETS

In this phase, the *FP-Tree*$_m$ generated in the 1st phase is taken by Mappers which connect trees with each other from different nodes. Subsequently, the Reducers extract the frequent itemset from trees, and save them in memory temporarily. The output of this phase is the list of frequent itemset. The following steps are required to find the frequent itemsets and the algorithm is presented in Algorithm 2.

1. Divide F-list to number of groups (mGroups) called G-list.
2. Each G-list is sent to different processors each of which has MapReduce.
3. For every processor, the items of descending order of F-list (from that last item to first

item) are examined to find out whether these items belong to G-list or not.

a. Mapper reads all paths of each item in different FP Trees and extract 1 temporary local F-list for each item.
b. Reduce constructs temporary local tree for each item based on their paths and temporary local F-list.
c. For every temporary local tree, Reduce extracts local frequent items for the items with unique paths, otherwise, the previous steps are repeated.

4. Merge all local frequent item list on different processor to form frequent items.

Algorithm 2: Finding Frequent Itemset

Procedure: **Finding Frequent itemset (Trees local, F-List,** σ)
Division of **F-List** in to *m* **Groups** G-List
Send G-List to P different
In each P different
For(i=**F-List.Size-1;i=0;i--)**
{
a: F-List[i]
If (a⊏G-List)
{
Mapper Read all paths in Local Trees to F- List local Temp of a Reduce building TreelocalTemp of item(a)
If (TreelocalTemp of item (a) is single path)
{
Output(Frequent Items list Local Temp)
}
}
}
Integration Frequent items local lists to Frequent Items List

4.3 BUILDING INCREMENTAL INTERESTING MODEL

In this phase, association patterns are extracted from the frequent itemsets. These patterns are evaluated using confidence measure and prune the patterns that do not satisfy this criterion resulting in a set of strong association patterns which are subjected to the novelty criterion [11] with the aim of deciding either these patterns are interesting or not. This phase takes into consideration the existing model M_i representing the known association rules and consequently resulting in discovering of M_{i+1}. For each frequent itemsets, only novel rules are extracted and used to update the model M_{i+1}. We compute novelty degree rule with the novelty measure (NM), (NM) presented in [10] as shown in equation 1:

$$NM = \frac{(|S1| + |S2| - 2 * K) + \sum_{i-1}^{k} \delta(C_1^i, C_2^i)}{|S1| + |S2|}, \quad (1)$$

where $S1$ and $S2$ are two conjunct sets with cardinalities $|S1|$ and $|S2|$ respectively. K = the pairs of compatible conjuncts between $S1$ and $S2$. (C_1^1, C_2^i) is the i^{th} pair of compatible conjuncts. The algorithm computes novelty measure (NM) at every stage of rules generation to determine whether a rule is likely to lead to an interesting rule, or not. A rule becomes a candidate for next stage rule generation if its novelty measure (NM) value is 1 or the relevance factor of the closest rule in M is less than the relevance factor threshold value. An interestingness value of 1 of the partial temporal rule indicates that this rule is unlikely to expand to any existing temporal association rule. The following steps are required to build the incremental interesting model and the algorithm is presented in Algorithm 3.

1. Generate association rules R from frequent item list.
2. Compute the Confidence of the rule (R)
3. If *Confidence* (R)>= θ Go step 4 else Go step 1
4. Compute the novelty measure (NM) of R with respect to Model M_i
5. If $NM(R)$> Φ Go step 6 else Go step1
6. Update Model M_i/M_{i+1}

5. A DETAILED EXAMPLE

For better understanding of our framework, consider a Big Data D arrived at time T_1, denoted by D_0. It contains 6 transactions as shown in Table 1. Suppose, D_0 is partitioned into 3 parts for the sake of parallel mining, i.e., m = 3, each of which is called dP_i, i = 1; 2; 3 whereas $dp \subset D$. Table 2 shows the data in every partition which has to be sent to different computers P_i. The computers P_i in turn computes support of its items by using Mapper and store the counts into f-list local. The following F-list local are generated from $P1$, $P2$ and $P3$ respectively: $f-listL_1$= {A=1,B=2,C=1,D=2,E=1}, $f-listL_2$= {A=2,B=2,C=1,D=2,E=1,F=1,G=1}, and $f-listL_3$= {A=1,B=2,E=2,C=2,D=1,G=1}.

Table 1. The transactions of D_0

ID	Items
1	A B C D
2	B E D
3	A B C D F
4	A B E D G
5	A B E C D
6	B E G C

Table 2. D_0 into 3 parts

ID	Items		Ordered frequent items
1	A B C D	dP_1	B D A C
2	BE D		B D E
3	A B C D F	dP_2	B D A C
4	A B E D G		B D A E
5	A B E C D	dP_3	B D C E
6	B E G C		B C E

Subsequently, the f-local lists are merged into cumulative list called F-List as follows: F-list = {A=4, B=6, C=4, D=5, E=4, F=1, G=2}. Now if we consider that the minimum, support σ =50%, the items G and F will be eliminated from F-list. Then, F-list is sorted on the basis of support in descending order as follows: F-list= {B=6,D=5,A=4,C=4,E=4}. The final F-list represents the reference for every *Pi* where local trees are constructed using Reduce. During construction of local trees, the items in every transaction are sorted in descending order according to their position in F-list and ignore items which are not in F-list as shown in the third column of Table 2. The ordered frequent items are used to construct local trees in which the roots are set to null. The local trees are constructed using FP-growth algorithm in every computer *Pi* as shown in Fig. 2. These local trees and F-list, which are maintained in the memory of *Pi* by using MapReduce, are the outcome of the first stage of the proposed framework. As the *FP-Growth* makes use of bottom-up strategy, the last item in *F-list* is considered first which is E in our example. All paths of E are examined in all local trees resulting the following: dP1:[D,B:1], dP2:[A,D,B:1], dP3[C,D,B:1]. These paths are used to generate temporary F-list using Mapper as follows: F-Listnew-E=[D=3,B=3,A=1,C=1]. Then, the items C and A are removed as they don't meet the support criterion and F-Listnew-E are reordered in descending order and also reorder the paths according to F-Listnew-E. Finally, the frequent items for item E are extracted using Reduce as the path is unique. The next item which is considered is C in which all paths of this item are examined resulting in the following: dP1:[A,D,B:1], dP2:[A,D,B:1], dP3[D,B:1], [B:1]. Then, Mapper is used to generate a temporary F-list for the item C contains F-listnew-C=[D=3,B=4,A=2]. Note that the item A will be removed due to minimum support criterion. Finally, the frequent items are generated as the path of the item C is unique. Similarly, the same process is executed for the remaining items and all frequent items are merged together which form the outcome of this stage. In our example, the frequent items list is = {[B,D,E:3] and [A,B,C,D:3]}. The next stage is to generate association rules from frequent item sets generated in the previous stage. Table 3 shows the corresponding set of discovered association rules assuming that the confidence threshold value is 0.6 for the frequent items {B,D,E:3}.

Table 3. The Association Rules Discovered at Time T_1

Rule No.	Association rules	Confidence	Accept	Compare with Rule	Novel φ	Novelty>50 Add to PDK
R1	B→D	3/6=.50	No	-------	-------	-------
R2	D→B	3/5=.60	Yes	Null	$\frac{0+1-2*0+0}{0+1}=1$	Yes
R3	B→E	3/6=.50	No	-------	-------	-------
R4	E→B	3/4=.75	Yes	R2	$\frac{2+2-2*1}{2+2}=.5$	Yes
R5	D→E	3/5=.60	Yes	R2,R4	$\frac{2+2-2*1+0}{2+2}=.5$	Yes
R6	E→D	3/4=.75	Yes	R2,R4,R5	$\frac{2+2-2*1}{2+2}=.5$	Yes
R7	B→DE	3/6=.50	No	-------	-------	--
R8	BD→E	3/5=.60	Yes	R5	$\frac{2+3-2*2}{2+3}=.2$	No
R9	BE→D	3/4=.75	Yes	R6	$\frac{2+3-2*2}{2+3}=.2$	No
R10	D→BE	3/5=.60	Yes	R5	$\frac{2+3-2*2}{2+3}=.2$	No
R11	DE→B	3/4=.75	Yes	R2	$\frac{2+3-2*2}{2+3}=.2$	No
R12	E→BD	3/4=.75	Yes	R2	$\frac{2+3-2*2}{2+3}=.2$	No

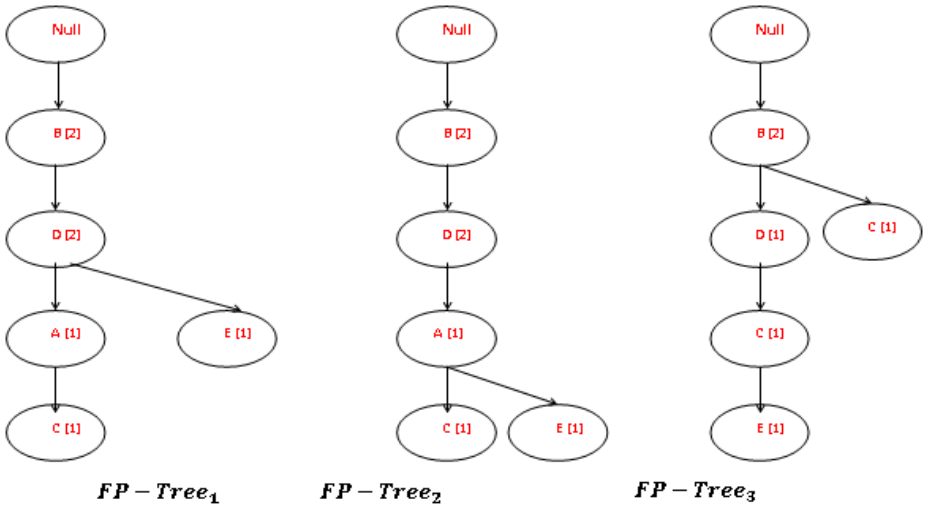


Figure 2 – The Local Trees on Different P_i

Notice that in Table 3, the rules R1, R3, and R7 are eliminated due to confidence criterion which is set to be 60%.

The remaining rules are subjected to novelty measure which is set in the example to be %50. As the data arrived at time T_1, no comparison will be made against Model M_1 because there are no novel rules discovered so far at time T_1. The last two columns of Table 3 show the computation of novelty degree of the rules and therefore the rules which are not novel are eliminated as they are uninteresting. Subsequently, the model M_1 is updated as shown in Fig. 3. Now suppose another data D_1 arrives at time T_2 as shown in Table 4. The same stages are repeated taking into consideration the novel rules in the model M_1. Table 5 shows the corresponding set of the discovered association rules assuming that the confidence threshold value is 0.6 for the frequent items {B,C,E : 2}. Notice that in Table 5, the rules {R13,R17,R18} are found to be novel and hence the model M_1 is updated incrementally to form model M_2 as shown in Fig. 4.

Table 4. The transactions of D_1

ID	Items
1	A C D
2	B C E
3	A B C E
4	B E

Table 5. The Association Rules Discovered at Time T_2

Rule No.	Association rules	Confidence	Accept	Compare with Rule	Novel φ	Novelty>50 Add to PDK
R13	B→C	2/3=.67	Yes	R2,R4	$\frac{2+2-2*1}{2+2}=.5$	Yes
R14	C→B	2/3=.67	Yes	R13	$\frac{2+2-2*2}{2+2}=.0$	No
R15	B→E	2/2=1	Yes	R4	$\frac{2+2-2*2}{2+2}=.0$	No
R16	E→B	2/3=1	Yes	R4	$\frac{2+2-2*2}{2+2}=.0$	No
R17	C→E	2/3=.67	Yes	R5,R13	$\frac{2+2-2*1}{2+2}=.5$	Yes
R18	E→C	2/3=.67	Yes	R13	$\frac{2+2-2*1}{2+2}=.5$	Yes
R19	B→CE	2/3=.67	Yes	R4	$\frac{2+3-2*2}{2+3}=.2$	No
R20	CE→B	2/2=1	Yes	R4	$\frac{2+3-2*2}{2+3}=.2$	No
R21	C→BE	2/3=.67	Yes	R4	$\frac{2+3-2*2}{2+3}=.2$	No
R22	BE→C	2/3=.67	Yes	R13	$\frac{2+3-2*2}{2+3}=.2$	No
R23	E→BC	2/3=.67	Yes	R13	$\frac{2+3-2*2}{2+3}=.2$	No
R24	BC→E	2/2=1	Yes	R17	$\frac{2+3-2*2}{2+3}=.2$	No

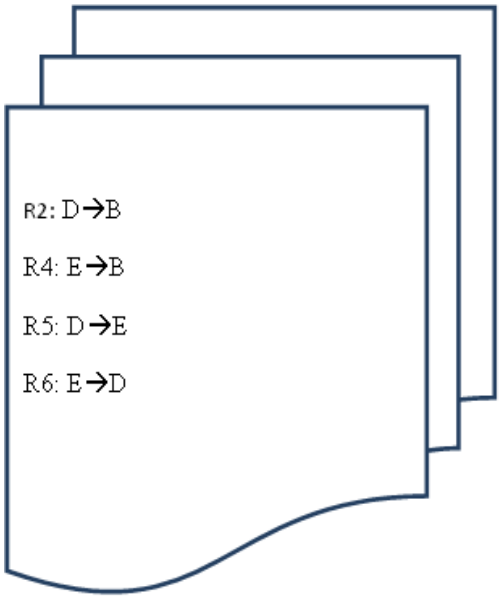


Figure 3 – Model M_1

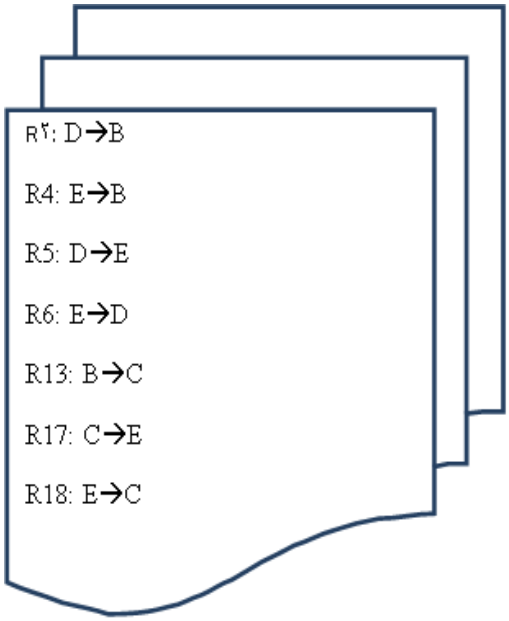


Figure 4 – Model M_2

6. EXPERIMENTAL STUDIES

In this section, experimental results are presented, in particular those related to the proposed framework performance. We conducted two experiments, the first experiment is shown in section 6.1 and the second experiment is in section 6.2. The proposed framework and other algorithms are written in Java and implemented on Hadoop. All experiments were conducted on a PC with Intel Core i5 2.6 GHz and 4G main memory, running on Microsoft Windows 10 64-bit. The experiments are conducted using real-life datasets available at http://kdd.ics.uci.edu. The datasets are considered as evolving with time, and divided up into three increments: D_1, D_2, and D_3 assumed that they have arrived at times T_1, T_2, and T_3 respectively. Table 6 shows the characteristics of these datasets.

Table 6. Dataset Characteristics

Dataset	Time	Size of Dataset	No of items	Items density
Kosarak	T_1	330001	31783	8
	T_2	330001	32098	8
	T_3	330000	32218	8
Accidents	T_1	113395	395	33
	T_2	113394	398	33
	T_3	113394	385	33
T40I10D100K	T_1	33334	942	39
	T_2	33333	941	39
	T_3	33333	942	39
T10I4D100K	T_1	33334	868	10
	T_2	33333	870	10
	T_3	33333	869	10

6.1 FIRST EXPERIMENT

In this experiment, the performance of the proposed framework is compared to the FP-Growth algorithm, PFP-Growth, and PIFP. Since the number of discovered rules of the FP-Growth algorithm, PFP-Growth and PIFP are similar, we perform the comparison to the FP-Growth algorithm only.

Table 7. The discovered rules on Kosarak dataset using FP-Growth and T_1; T_2; T_3 in our Framework

Minimum support σ	Minimum confidence θ	FP-Growth Algorithm	Our framework with Novelty threshold Φ=0.50		
		Discovered rules	Novel rules		
		$T_1+T_2+T_3$	T_1	T_1+T_2	$T_1+T_2+T_3$
0.004	0.60	5789	1200	1478	1567
	0.65	5384	1168	1392	1452
	0.70	5061	1034	1203	1237
0.006	0.60	2303	478	566	582
	0.65	2180	453	531	548
	0.70	2034	423	487	502
0.008	0.60	981	213	253	264
	0.65	923	200	238	246
	0.70	843	185	218	225
0.010	0.60	577	133	152	158
	0.65	537	125	142	146
	0.70	492	115	130	133
0.012	0.60	372	87	98	99
	0.65	344	81	91	91
	0.70	309	74	82	82

The performance is measured in term of the number of discovered rules with various thresholds of minimum support and confidence and fixed novelty threshold Φ = 0.50. The dataset used is (Kosarak) and it is considered evolving with time and partitioned into three parts representing times T_1; T_2; T_3 respectively as shown in Table 6. As we can notice in Table 7, the number of discovered rules is reduced in the proposed framework compared to FP-Growth in all various minimum support and confidence.

Fig. 5, Fig. 6, and Fig. 7 show the reduction of the discovered rules using (Kosarak) dataset at T_1,T_2, and T_3 times.

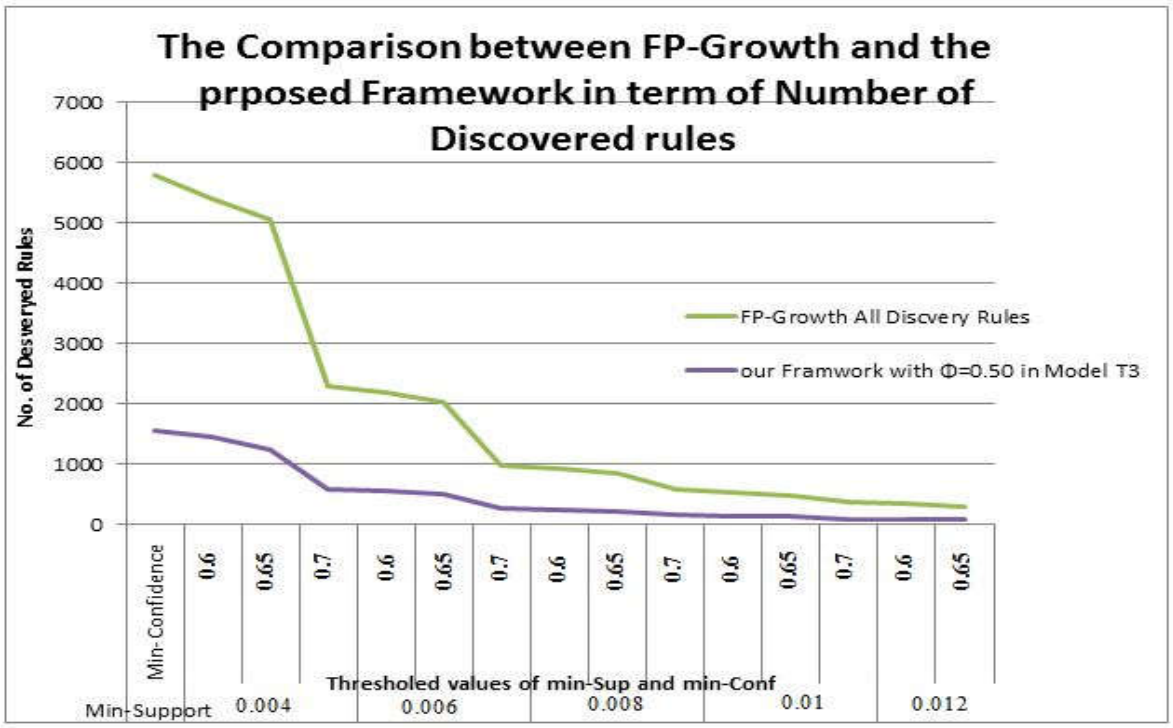


Figure 5 – The Comparison between FP-Growth and the Proposed Framework in term of number of discovered rules using (Kosarak) dataset

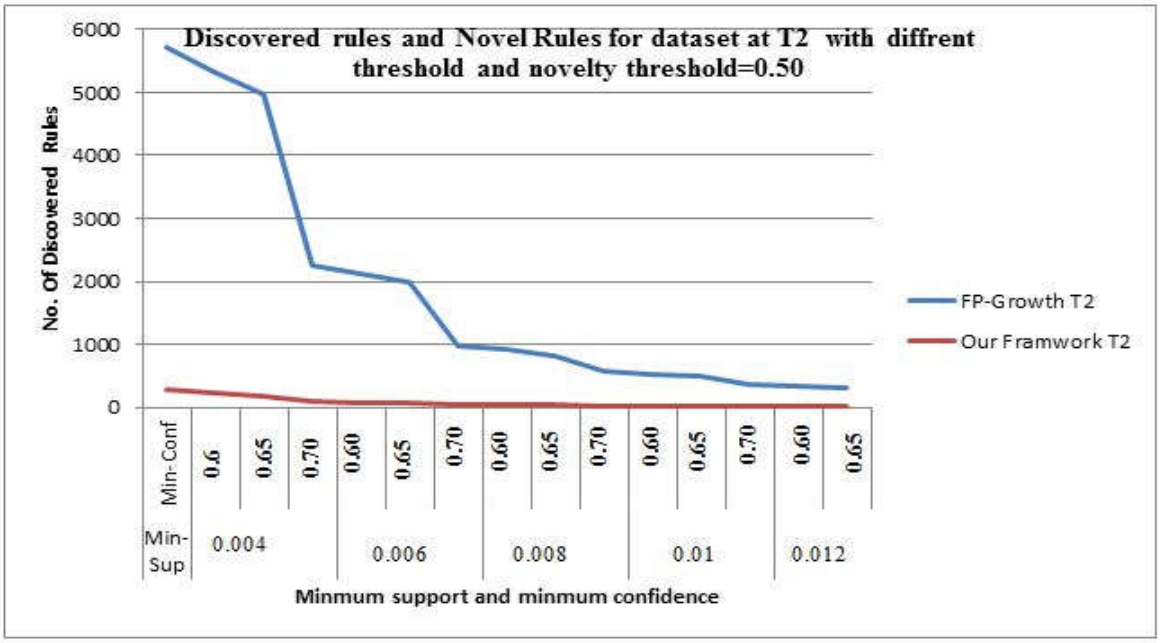


Figure 6 – Discovered rules and Novel Rules for dataset at T_2

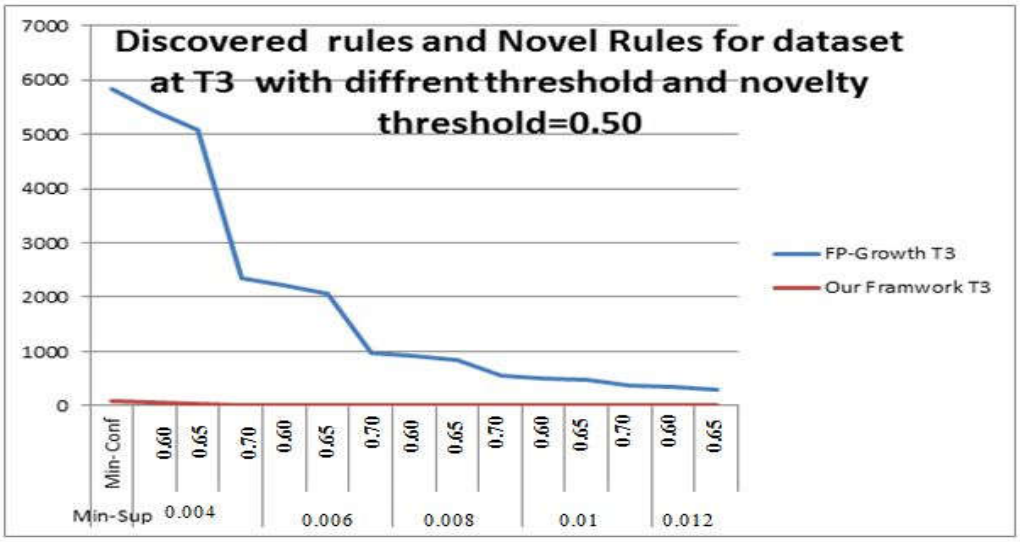


Figure 7 – Discovered rules and Novel Rules for dataset at T_3

6.2 SECOND EXPERIMENT

The objective of the second experiment is to show the effectiveness of our framework in reducing the number of discovered rules against PIFP algorithm. It is expected that the number of discovered rules keeps on decreasing over the time. We work with four datasets and considered these datasets as evolving with time, and partitioned them into 3 increments: D_1, D_2 and D_3 mined at times T_1, T_2 and T_3 respectively. For each dataset used, the minimum support σ and minimum confidence θ are fixed and Novelty threshold Φ varies. It is observed that the number of interesting rules decreases in our framework in contrast to the number of rules discovered by PIFP algorithm at T_1,T_2, and T_3. Intuitively, the interesting rules discovered by our framework at time T_1 is no more interesting at time T_2 and the interesting rules discovered at time T_2 is no more interesting at time T_3. Consequently, as the value of Novelty threshold Φ increases, the number of discovered interesting rules decreases at each time as per our expectations. The results are demonstrated in Table 8.

Table 8. The Comparison between PIFP and the Proposed Framework in term of number of discovered rules using many datasets with various novelty threshold Φ

Dataset	minimum support σ	minimum confidence θ	PIFP-Growth			Novelty threshold Φ	Our framework		
			T_1	T_2	T_3		T_1	T_2	T_3
Kosarak	0.004	0.5	7049	6808	6906	0.5	1396	656	256
						0.6	1055	496	193
						0.7	983	462	180
						0.8	932	438	171
						0.9	831	391	152
Accidents	0.3	0.8	4422352	4413405	4491707	0.5	884504	318422	89158
						0.6	751834	270660	75785
						0.7	530712	191056	53496
						0.8	309612	111460	31209
						0.9	221132	79607	22290
T40I10D100K	0.025	0.2	625	625	630	0.5	207	74	21
						0.6	194	70	20
						0.7	157	56	16
						0.8	125	45	13
						0.9	107	38	11
T10I4D100K	0.0005	0.3	1480	1295	1481	0.5	508	183	51
						0.6	370	133	37
						0.7	276	99	28
						0.8	235	85	24
						0.9	156	56	16

7. CONCLUSION AND FUTURE WORK

In this research, we proposed a framework for incremental parallel interesting association rule mining for Big Data. The proposed approach incorporates interestingness measure during the process of mining. It makes a self-upgrading model that utilizes novelty criterion to reflect the user subjectivity and extract patterns, incrementally, from datasets arrive at different points in time. Our future work includes enhancing the framework to create an association system in which the model can adapt to a data stream environment.

8. REFERENCES

[1] J. Manyika, M. Chui, B. Brown, J. Bughin, R. Dobbs, C. Roxburgh, et al., *Big data: The Next Frontier for Innovation, Competition, and Productivity*, McKinsey Global Institute, June 2011, pp. 156.

[2] A. Kejariwal, "Big data challenges: a program optimization perspective," *Proceedings of the 2012 Second International Conference on Cloud and Green Computing*, 2012, pp. 702-707.

[3] S. Kaisler, F. Armour, J. A. Espinosa, and W. Money, "Big data: Issues and challenges moving forward," *Proceedings of the 2013 46th Hawaii International Conference on System Sciences*, 2013, pp. 995-1004.

[4] S. Moens, E. Aksehirli, and B. Goethals, "Frequent itemset mining for big data," *Proceedings of the 2013 IEEE International Conference on Big Data*, 2013, pp. 111-118.

[5] R. Agrawal, T. Imieliski, and A. Swami, "Mining association rules between sets of items in large databases," *Proceedings of the 1993 ACM SIGMOD Conference*, 1993, pp. 207-216.

[6] J. Park, M. Chen, and P. Yu, "Efficient parallel data mining for association rules," *Proceedings of the fourth International Conference on Information and Knowledge Management CIKM'95*, 1995, pp. 31-36

[7] O.R. Zaane, M. El-Hajj, and P. Lu, "Fast parallel association rule mining without candidacy generation," *Proceedings of the 2001 IEEE International Conference on Data Mining*, 2001, pp. 665-668.

[8] H. Li, Y. Wang, D. Zhang, M. Zhang, and E.Y. Chang, "Pfp: parallel fp-growth for query recommendation," *Proceedings of the 2008 ACM Conference on Recommender Systems*, 2008, pp. 107-114.

[9] L. Liu, E. Li, Y. Zhang, and Z. Tang, "Optimization of frequent itemset mining on multiple-core processor," *Proceedings of the 33rd International Conference on Very Large Data Bases*, Vienna, Austria, 2007, pp. 1275–1285.

[10] V. Bhatnagar, A. S. Al-Hegami, and N. Kumar, "A hybrid approach for quantification of novelty in rule discovery," *Proceedings of the WEC*, vol. 2, 2005, pp. 39-42.

[11] V. Bhatnagar, A.S. Al-Hegami, and N. Kumar, "Novelty as a measure of interestingness in knowledge discovery," *International Journal of Information Technology*, vol. 2, no. 1, pp. 36-41, 2005.

[12] A.S. Al-Hegami, V. Bhatnagar, and N. Kumar, "Novelty framework for knowledge discovery in databases," *Proceedings of the International Conference on Data Warehousing and Knowledge Discovery*, 2004, pp. 48-57.

[13] E. Yafi, A.S. Al-Hegami, M.A. Alam, and R. Biswas, "YAMI: incremental mining of interesting association patterns," *Int. Arab J. Inf. Technol*., vol. 9, pp. 504-510, 2012.

[14] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proceedings of the 20th Int. Conf. on Very Large Data Bases*, VLDB, 1994, pp. 487-499.

[15] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *Proceedings of the ACM SIGMOD Conference*, 2000, pp. 1-12.

[16] A. Pradeepa and A. Thanamani, "Parallelized comprising for apriori algorithm using mapreduce framework," *International Journal of Advanced Research in Computer and Communication Engineering*, vol. 2, pp. 4365-4368, 2013.

[17] Y. Xun, J. Zhang, and X. Qin, "Fidoop: Parallel mining of frequent itemsets using mapreduce," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 46, pp. 313-325, 2016.

[18] J. M. Kunkel, "Simulating parallel programs on application and system level," *Computer Science-Research and Development*, vol. 28, pp. 167-174, 2013.

[19] Z. Zeng, C. Yang, and Y. Tao, "Research of load balance FP-growth algorithm in parallel," *Computer Engineering and Applications*, vol. 46, pp. 125-126, 2010.

[20] J. Dean and S. Ghemawat, "MapReduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, pp. 107-113, 2008.

[21] D.W. Cheung, J. Han, V.T. Ng, and C. Wong, "Maintenance of discovered association rules in large databases: An incremental updating technique," *Proceedings of the Twelfth*

International Conference on Data Engineering, 1996, pp. 106-114.

[22] D.W.-L. Cheung, V.T. Ng, and B.W. Tam, "Maintenance of discovered knowledge: A case in multi-level association rules," *Proceedings of the KDD*, 1996, pp. 307-310.

[23] D.W. Cheung, S.D. Lee, and B. Kao, "A general incremental technique for maintaining discovered association rules," *Proceedings of the Conference on Database Systems for Advanced Applications'97*, ed: World Scientific, 1997, pp. 185-194.

[24] V. Ganti and R. Ramakrishnan, "Mining and monitoring evolving data," in *Handbook of Massive Data Sets*, ed: Springer, 2002, pp. 593-642.

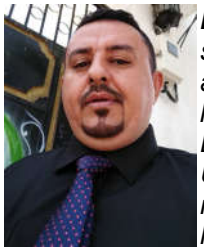
[25] S.D. Lee and D.W.-L. Cheung, "Maintenance of discovered association rules: When to update?," *Proceedings of the DMKD*, 1997, pp. 1-14.

[26] X. Wei, Y. Ma, F. Zhang, M. Liu and W. Shen, "Incremental FP-Growth mining strategy for dynamic threshold value and database based on MapReduce," *Proceedings of the 2014 IEEE 18th International Conference on Computer Supported Cooperative Work in Design (CSCWD)*, Hsinchu, 2014, pp. 271-276. DOI: 10.1109/CSCWD.2014.6846854

[27] M.J. Zaki and C.-J. Hsiao, "CHARM: An efficient algorithm for closed itemset mining," *Proceedings of the 2002 SIAM International Conference on Data Mining*, 2002, pp. 457-473.

[28] M. Riondato, J.A. DeBrabant, R. Fonseca, and E. Upfal, "PARMA: a parallel randomized algorithm for approximate association rules mining in MapReduce," *Proceedings of the 21st ACM International Conference on Information and Knowledge Management*, 2012, pp. 85-94.

***Ahmed Sultan Al-Hegami**, a Professor at the Faculty of Computers and Information Technology, Sana'a University, Yemen. His research interest includes artificial intelligence, machine learning, big data, temporal databases, real time systems, data mining, and knowledge discovery in databases.*



***Hussein Alkhader Alsaeedi**, PhD scholar at the University of Science and Technology of Yemen. He is a lecturer in Data Mining at the Department of Computer Science, University of Science and Technology,Yemen, Sana'a. He received his Master's Degree in Computer Information Systems from the Arab Academy, Sana'a.*