Indonesian Journal of Electrical Engineering and Computer Science
Vol. 3, No. 3, September 2016, pp. 546 ~ 553
DOI: 10.11591/ijeecs.v3.i2.pp546-553

Mining Association Rules: A Case Study on Benchmark Dense Data

Mustafa Man[1], Wan Aezwani Wan Abu Bakar[2], Zailani Abdullah[3], Masila Abd Jalil[4], Tutut Herawan[5]
[1,2,4]School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu,
21030 Kuala Terengganu, Terengganu, Malaysia
[3]Faculty of Entepreneur and Business, Universiti Malaysia Kelantan,
16100 Kota Bharu, Kelantan, Malaysia
[5]Department of Information Systems, Faculty of Computer Science and Information Technology,
University of Malaya, Lembah Pantai, 50603 Kuala Lumpur, Malaysia
Corresponding author, e-mail: mustafaman@umt.edu.my, beny2194@yahoo.com, zailania@umk.edu.my, masita@umt.edu.my, tutut@um.edu.my

Abstract

Data mining is the process of discovering knowledge and previously unknown pattern from large amount of data. The association rule mining (ARM) has been in trend where a new pattern analysis can be discovered to project for an important prediction about any issues. Since the first introduction of frequent itemset mining, it has received a major attention among researchers and various efficient and sophisticated algorithms have been proposed to do frequent itemset mining. Among the best-known algorithms are Apriori and FP-Growth. In this paper, we explore these algorithms and comparing their results in generating association rules based on benchmark dense datasets. The datasets are taken from frequent itemset mining data repository. The two algorithms are implemented in Rapid Miner 5.3.007 and the performance results are shown as comparison. FP-Growth is found to be better algorithm when encountering the support-confidence framework.

Keywords: *data mining, association rule mining (ARM), frequent pattern mining (FPM), rapid miner, apriori, fp growth*

1. Introduction

Data mining is the research area where the huge dataset in database and data repository are scoured and mined to find novel and useful pattern. Association analysis is one of the four (4) core data mining tasks besides cluster analysis, predictive modeling and anomaly detection [1]. The task of Association Rule Mining (ARM) is to discover if there exist the frequent itemset or pattern in database and if any, an interesting relationships between these frequent itemsets can reveal a new pattern analysis for the next step of decision making.

Finding frequent itemsets or patterns (as shown in Figure 1) is a big challenge and has a strong and long-standing tradition in data mining. It is a fundamental part of many data mining applications including market basket analysis, web link analysis, genome analysis and molecular fragment mining [2]. The idea of mining association rule originates from the analysis of market basket data [3]. Example of simple rule is “A customer who buys bread and butter will also tend to buy milk with probability s% and c%”. The applicability of such rule to business problems makes the association rule to become a popular mining method.

The ARM that relates to frequent pattern is called Frequent Pattern Mining (FPM). The state-of-the-art algorithms in FPM are based upon horizontal data format and vertical data format. Most of previous frequent mining techniques are dealing with horizontal format of their data repositories but suffer from the requirement of many database scans. However, current and emerging trend exists where some of the research works are focusing on dealing with vertical data format and the rule mining results are quite promising. Apriori [3, 4] that relies on horizontal format and FP-Growth [5] that relies on vertical format are among the best-known algorithms in FPM. Neither horizontal nor vertical data format, both are still suffering from the huge memory consumption [3-5] with higher datasets.

Received April 2, 2016; Revised July 25, 2016; Accepted August 10, 2016

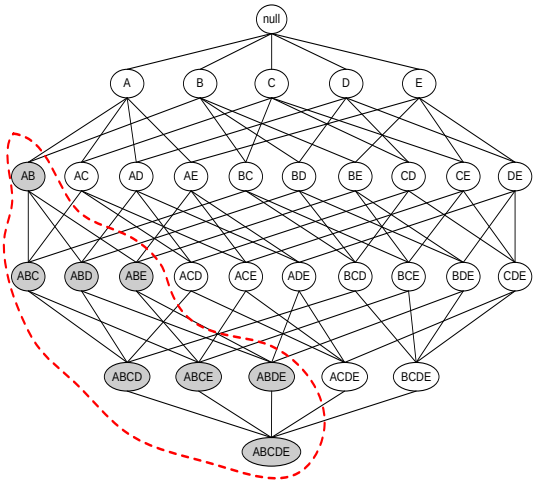

Figure 1. Frequent Itemset and Its Subset [2]

In this paper, we discover the performance and scalability measures of both algorithms that represent both formats and compare their results in generating association rules based on benchmark dense datasets. The datasets are taken from frequent itemset mining data repository.

The rest of this paper is organized as follow. Section 2 describes rudimentary of association rules. Section 3 describes Apriori and FP Growth algorithms. Section 4 describes experimental results. Finally, the conclusion of this work is described in section 5.

1.1. Association Rules

Following is the formal definition of the problem in [3]. Let $I = \{i_1, i_2,\ldots,i_m\}$ be the set of items. Let D is a set of transaction where each transaction T is a set of items such that $T \subseteq I$. An *association rule* is an implication of the form $X \subseteq Y$, where X represents the antecedent part of the rule and Y represents the consequent part of the rule where $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \emptyset$. The itemset that satisfies minimum support is called *frequent itemset*. The rule $X \Rightarrow Y$ holds in the transaction set D with *confidencec* if c% of transactions in D that contain X also contain Y. The rule $X \Rightarrow Y$ has *supports* in the transaction set D if s% of transaction in D contains $X \cup Y$. The illustration of support-confidence notions is given as below:

a) The support of rule $X \Rightarrow Y$ is the fraction of transactions in D containing both X and Y.

$$Support\ (X \Rightarrow Y) = \frac{X \cup Y}{|D|}.$$

Where $|D|$ is the total number of records in database.

b) The confidence of rule $X \Rightarrow Y$ is the fraction of transactions in D containing X that also contain Y.

$$Confidence\ (X \Rightarrow Y) = \frac{supp\ (X \cup Y)}{supp\ (X)}.$$

The rules which satisfy both a minimum support threshold (min_supp) and minimum confidence threshold (min_conf) are called *strong rule* where min_supp and min_conf are user specified values.

1.2. Apriori and FP Growth Algorithms

Since the introduction of frequent itemset mining by [3], it has received a major attention among researchers and various efficient and sophisticated algorithms have been proposed to do frequent itemset mining. Among the best-known algorithms are Apriori and FP-Growth.

1.2.1. Apriori Algorithm

The Apriori algorithm [3, 4] uses a breadth-first search and the downward closure property, in which any superset of an infrequent itemset is infrequent, to prune the search tree. Apriori usually adopts a horizontal layout to represent the transaction database and the frequency of an itemset is computed by counting its occurrence in each transaction. Apriori uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as *candidate generation*, and groups of candidates are tested against the data). The algorithm terminates when no further successful extensions are found. The key idea is such that the apriori property (downward closure property) states that any subsets of a frequent itemset are also frequent itemsets. The best known algorithm that involve two steps:

1) Step 1: Find all itemsets that have minimum support (frequent itemsets, also called large itemsets).
2) Step 2: Use frequent itemsets to generate rules.

1.2.2. Apriori Pseudo-code

```
Ck: Candidate itemset of size k
Lk : frequent itemset of size k
L1 = {frequent items};
for (k = 1; Lk !=∅; k++) do begin
Ck+1 = candidates generated from Lk ;
for each transaction t in database do
      increment the count of all candidates in Ck+1  that are contained in t
Lk+1  = candidates in Ck+1 with min_support
 end
return∪kLk;
```

1.2.3. Apriori principle

1) If an itemset is frequent, then all of its subsets must also be frequent. Apriori principle holds due to the following property of the support measure:

$$\forall X,Y:(X\subseteq Y)\Rightarrow s(X)\geq s(Y)$$

2) Support of an itemset never exceeds the support of its subsets

1.2.4. FP-Growth Algorithm

The FP-Growth [5-7] employs a divide-and-conquer strategy and a FP-tree data structure to achieve a condensed representation of the transaction database. It has become one of the fastest algorithms for frequent pattern mining. In large databases, it's not possible to hold the FP-tree in the main memory. A strategy to cope with this problem is to firstly partition the database into a set of smaller databases (called projected databases), and then construct an FP-tree from each of these smaller databases. The steps are as follows:

1. Scan DB once, find frequent 1-itemset (single item pattern)
2. Sort frequent items in frequency descending order, f-list
3. Scan DB again, construct FP-tree

The algorithm of FP-Growth is given as below and Figure 2, 3 and 4 show the steps in constructing FP Tree from a transaction database.

1.2.5. FP Growth (Tree, α) Pseudocode

```
if Tree contains a single path P then
for each combination (denoted as β) of the nodes in the path P do
        generate pattern β ∪ α with support = min_supp of nodes in β;
else for each αi in the header of Tree do {
        generate pattern β = αi ∪ α with support = αi.support;
        construct β's conditional pattern base and then
          β's conditional FP tree Treeβ;
        if Treeβ ≠ ∅then
          call FP-Growth (Treeβ, β) }
```

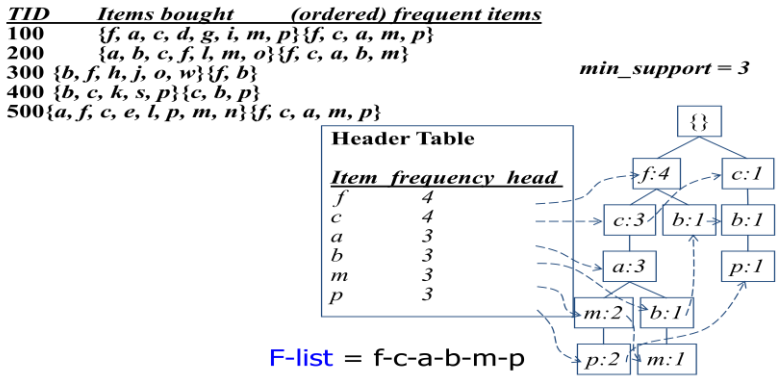

Figure 2. Construct FP-tree from a Transaction Database [5]

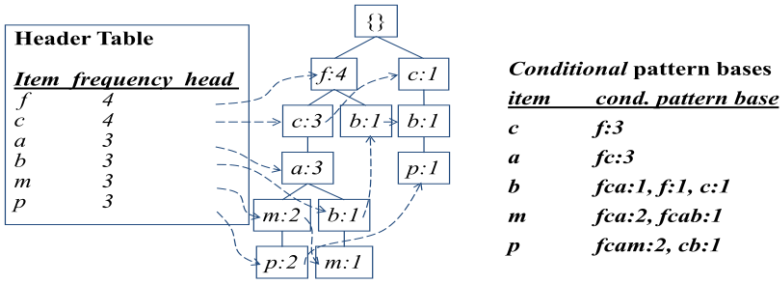

Figure 3. Finding Patterns Having P from P-conditional Database [5]

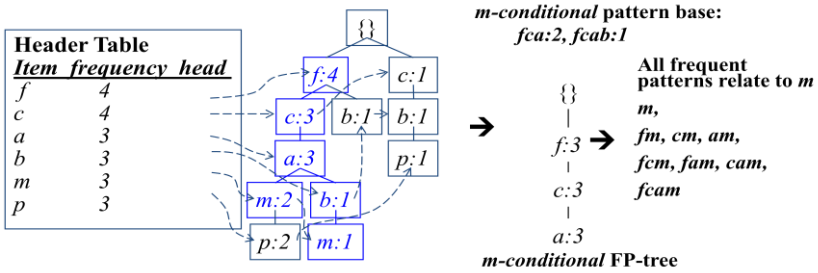

Figure 4. From Conditional Pattern-bases to Conditional FP-trees [5]

2. Experiment Results

2.1. Experimentation Platform and Datasets

All experiments are performed on a DELL Inspiron 620, Intel ® Pentium ® CPU G630 @ 2.70 GHz with 4GB RAM in a Win 7 64-bit platform. The tool used is Rapid Miner (RM) 5.3.007. The raw benchmark data are retrieved from http://fimi.ua.ac.be/data/ in a *.dat file format. For the ease of use in RM, we convert to Comma Separated Value (CSV) format. For experimentation purposes, some datasets have been modified by removing instances that have incomplete data and removing attributes that have only one categorical value. In RM itself, we have to perform data transformation in order to be processed through the specified algorithm. When importing data into RM, we have to specify what parameter to be set as ID, label, or attributes. There are five (5) datasets include chess, connect, mushroom, pumb_star and T40I10D100K. All selected datasets are different from one another in terms of size, either horizontally or vertically aimed to analyze the performance of selected algorithms when involving a huge number of records as well as very high number of attributes. Table 1 shows the characteristics of datasets.

Table 1. Database Characteristic

Datasets	Size (KB)	Average Length (Attribute)	Records (Transaction)
Chess	335	37	3196
Connect	9039	43	67558
Mushroom	558	23	8125
Pumb_star	11028	50	49047
T40I10D100K	15116	32	100001

2.2. RM Development and Results

The results of the experiments are summarized in three (3) tables and six (6) figures. Figure 5 illustrates the processes involved in deploying Apriori algorithm.

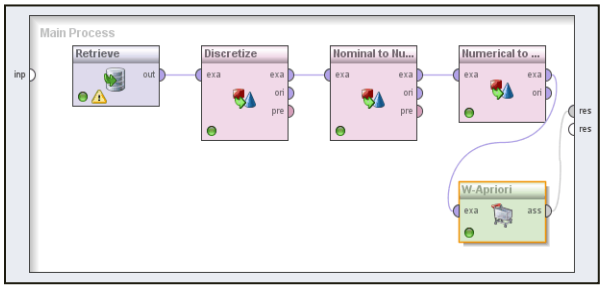

Figure 5. Rapid Miner Processes by W-Apriori algorithm

The W-Apriori process is an extension of Weka-Apriori into the RM tool. First, the benchmark data (in csv) is retrieved by calling *retrieve()* process. Then data transformation has to be constructed through *descretizebyfrequency()* process. This operator converts the selected numerical attributes into nominal attributes by discretizing the numerical attribute into a user-specified number of bins. Bins of equal frequency are automatically generated, the range of different bins may vary. Then data is converted from *nominaltonumerical()* to *numericaltopolynominal()*. The process *nominaltonumerical()* is to change the nominal attributes to numerical attributes while the process *numericaltopolynominal()* is to change the numerical attributes to polynominal attributes, that is allowed in Apriori algorithm. Then we call the Weka extension, *W-Apriori()* to generate the best rules. The parameter is set to be a default value.

Figure 6 depicts on the processes involved in deploying the Weka extension W-FP-Growth algorithm.

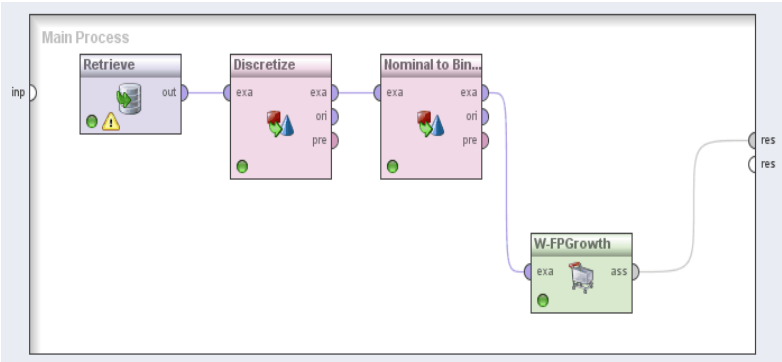

Figure 6. Rapid Miner Processes by W-FPGrowth Algorithm

The root process starts with retrieving the csv dataset. Then the *discretizeby frequency()* is selected to change the real attributes to nominal. Next, the *NominaltoBinominal()*

process is called to change the nominal attributes to binominal attributes which is allowed in FPGrowth algorithm and lastly *W-FPGrowth()* process is called to find frequent pattern and generate the rules.

The performance of the Apriori and FP-Growth algorithms are measured in terms of total execution time and total generated rules. The running time is subjected to factors such as different search method in both algorithms and also the size of dataset itself.

Figure 7–12 illustrate the graphs of the results obtained. From these figures which representing 3 different values of min_conf (i.e. 0.9, 0.5 and 0.1), it can be seen that the patterns plotted are almost similar. The graphs show that W-FPGrowth outperforms W-Apriori where more number of rules generated within lesser execution time. From the detailed result in RM, between W-Apriori and W-FPGrowth, there are almost similar attributes interpreted to be the antecedent and consequent. With W-FPGrowth, there are more attributes found to be the interesting rules as compared to W-Apriori. For any mining algorithm, it should find the same set of rules although their computational efficiencies and memory requirements may be different [5].

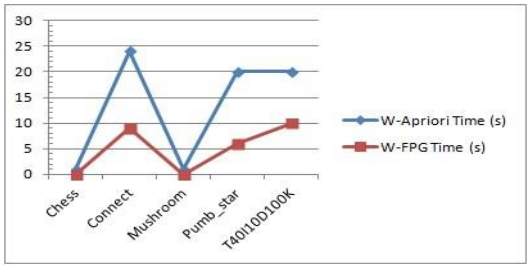

Figure 7. W-Apriori vs. W-FPGrowth: Execution time (in seconds) when min_conf = 0.9

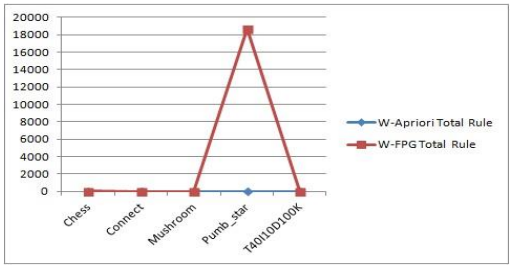

Figure 8. W-Apriori vs. W-FPGrowth: Rules generated when min_conf = 0.9

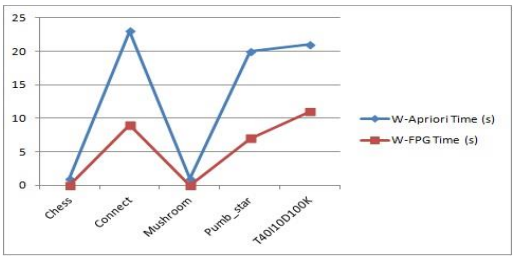

Figure 9. W-Apriori vs. W-FPGrowth: Execution time (in seconds) when min_conf = 0.5

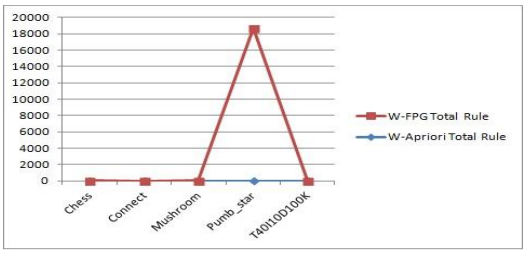

Figure 10. W-Apriori vs. W-FPGrowth: Rules generated when min_conf = 0.5

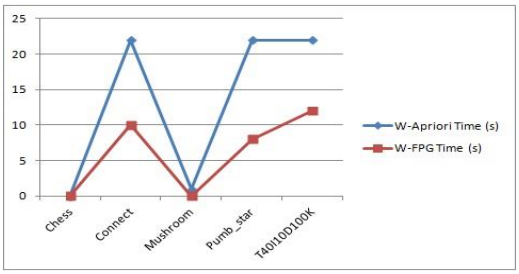

Figure 11. W-Apriori vs. W-FPGrowth: Execution time (in seconds) when min_conf = 0.1

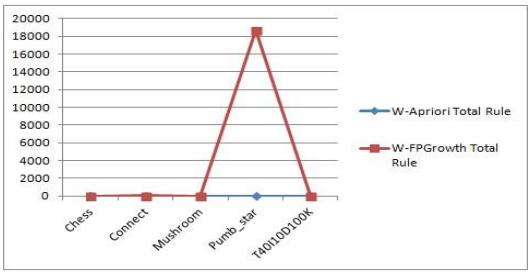

Figure 12. W-Apriori vs. W-FPGrowth: Rules generated when min_conf = 0.1

The results of scalability measures on Apriori and FP Growth algorithm are depicted in Figure 13–16 by taking min_conf as 0.9. The scalability is measured in two different types. The first is the scalability of both algorithms in dataset size on execution time and the second is the scalability of both algorithms in dataset size on rules generated. From the plotted graphs, it can be observed that the scalability of both algorithms in dataset size on execution time is almost similar and tends to non-linear, but there is a slightly different of pattern plotted on the number of rules generated. In Apriori, the pattern plotted on rules generated is similar with the pattern on execution time, but in FP Growth, the pattern plotted on rules generated is slightly different where in pumb_star with total of 15116 size (in KB), the number of rules generated shows quite a huge number which is 18860 rules.

In reviewing the scalability of Apriori and FP Growth on five (5) datasets, taking the value in dataset size, there is only small variation on execution time or number of rules generated. However on the whole, these algorithms have a good scalability to data size. This can be seen through Figure 13–16 where the good scalability of algorithm to the data size is highly desirable in real data mining applications [8].

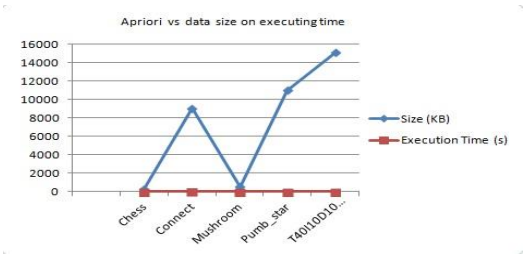

Figure 13. Scalability of Apriori vs data size on executing time when min_conf=0.9

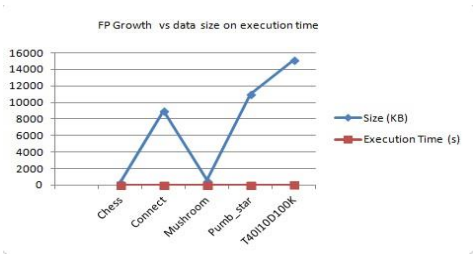

Figure 14. Scalability of FP Growth vs data size on executing time when min_conf=0.9

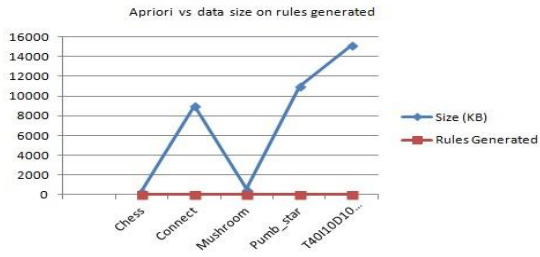

Figure 15. Scalability of Apriori vs data size on rules generated when min_conf=0.9

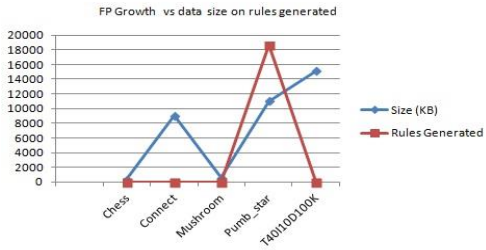

Figure 16. Scalability of FP Growth vs data size on rules generate when min_conf=0.9

3. Conclusion

In this paper, we have explored two well-known benchmark association rules mining algorithms. The experiment conducted in this paper has shown a comparison results between Apriori and FPGrowth algorithms using the benchmark dense datasets. It is clearly imitated from the graphs that FP Growth outperforms W-Apriori in terms of lesser execution time with more rules generated. The W-FPGrowth is found to be better algorithm when encountering the support-confidence framework. Originally, W-FPGrowth only performs 2 passes over datasets and, the datasets are already “compressed” with the generation of the FP-Tree by reducing irrelevant information. This is done through removing infrequent items. While for W-Apriori, it requires multiple database scans and a candidate generation approach by self-joining (by generating all possible candidate itemsets) before pruning (by removing those candidates that is not frequent). Therefore, it results in more execution time and generated rules are always 10 because the size of largest itemset is bound to 10. The more execution time generated, the higher of memory consumption of the machine.

There are many other interestingness measure that can be imposed to the algorithm and see whether the performance result between Apriori and FPGrowth are still the same or otherwise. For the future analysis, there are few alternatives we might want to tackle either in the same interestingness measure with vertical data format approach of Eclat algorithm [2] or with different interestingness measure but with similar ARM rule and compare the outcome.

Acknowledgements

We wish to thank Prof. Dr. Mohd Yazid Mohd Saman for his insightful comments and suggestions and a credit also to MyPhD scholarship under MyBrain15 of Kementerian Pendidikan Malaysia (KPM) for the financial foundation of this work.

References

[1] Tan PN, Steinbach M, Kumar V. Introduction to Data Mining. First Edition. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. 2005.
[2] Trieu TA, Kunieda Y. *An improvement for declat algorithm*. Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication (ICUIMC'12). 2012; 54: 1-06.
[3] Agrawal R, Srikant R, et al. *Fast algorithms for mining association rules*. In Proc. 20th int. conf. very large data bases, VLDB. 1994; 1215: 487–499.
[4] Agrawal R, Imieli´nski T, Swami A. *Mining association rules between sets of items in large databases*. In SIGMOD Rec. 1993; 22: 207–216.
[5] Han J, Kamber M, Pei J. Data mining: concepts and techniques. Morgan Kaufmann. 2006.
[6] Han J, Pei J, Yin Y. *Mining frequent patterns without candidate generation*. In ACM SIGMOD Record. ACM. 2000; 29(2): 1-12.
[7] Han J, Pei J, Yin Y, Mao R. *Mining frequent patterns without candidate generation: A frequent-pattern tree approach*. In Data mining and knowledge discovery. 2004; 8(1): 53–87.
[8] Mamat R, Herawan T, Deris MM. *MAR: Maximum attribute relative of soft set for clustering attribute selection*. In Knowledge-Based Systems. 2013; 52: 11–20.