International Journal of Engineering & Technology, 7 (3.28) (2018) 144-148

International Journal of Engineering & Technology

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

Research paper

Comparative Performance Analysis of Postdiffset in Frequent vs. Infrequent Itemset Mining

Wan Aezwani Wan Abu Bakar[1]*, Mustafa Man[2], Julaily Aida Jusoh[1]

[1]*Faculty of Informatics and Computing, Universiti Sultan Zainal Abidin, Besut Campus, 22200 Besut, Terengganu, Malaysia*
[2]*School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu, 21030 Kuala Terengganu, Terengganu, Malaysia*
**Corresponding Author E-mail: wanaezwani@unisza.edu.my*

Abstract

This paper presents comparative performance analysis of Postdiffset algorithm in mining of frequent and infrequent itemset via FIMI (Frequent Itemset Mining) benchmark case study. The Postdiffset is the Eclat-variant algorithm that mines the itemsets in tidsets (transaction id of items) format in the first looping and follows by diffset (difference set of itemsets) in the second looping onwards. We apply Postdiffset in mining of both frequent and infrequent itemset via dense datasets of chess and mushroom as well as for sparse datasets of retail and T10I4D100K. The overall results show postdiffset performs moderately between 21% to 40% towards tidset algorithm in frequent itemset mining in all datasets but loose performance towards diffset and sortdiffset. Contradictory, postdiffset gives promising results in terms of execution time with outperforming in all algorithms (diffset and sortdiffset) for all selected dense and sparse datasets between 23% to 99% outperformance percentage.

Keywords: *Postdiffset; performance analysis; frequent itemset; infrequent itemset.*

1. Introduction

Frequent and infrequent mining of itemset or pattern have been a spur of research in association rule mining (ARM). Originates from the finding of relationships among frequent items in examining customer purchasing behavior in supermarket transaction data, ARM seeks to examine combination of the items or set of items that may affects in the presence of new or specific items. The relationship is not only based on the intrinsic properties of data itself but rather based on the co-occurrence of the items within the transactional database [1]. Not only frequent, the recent real scenario in medical, forestry or security and defense department datasets show an increasing demand for infrequent or rare association mining. In both frequent and infrequent itemset or pattern mining, various algorithms have been proposed from the state-of-the-art base models of either Apriori [2], FP-Growth [3] or Eclat [4] approaches and methodologies. With regards to these 3 approaches, the design of database structure is based upon horizontal or vertical data formats. These two data formats have been widely discussed by showing few examples of algorithm of each data formats. Regardless of both database format, the issues of huge memory consumption due to highly candidate generation during association process are yet to be accomplished. Our approach is to have further investigation on frequent and infrequent itemsets that underlies on vertical data formats as our database layout structure. The objective is to reduce in memory consumption usage during candidate generation that is done through intersection process of each itemsets.
In this paper, we propose our Postdiffset as another Eclat-invariant algorithm. We conduct and test our algorithm in two (2) different itemsets category i.e. frequent and infrequent itemsets. The results of experimentation are compared in view of performance analysis. The results of both categories show a new finding to the filed of association rule mining.

2. Mining of Frequent and Infrequent Itemset

Frequent itenmset mining refers to the mining of the set of items that frequently occur, while infrequent (rare) itemset mining means the mining of the sets of items that are infrequently (rarely) occur in the transactional database [6, 7]. The determination of each items to be set as frequent or infrequent relies on the setting of minimum support (min_supp) threshold value. If the items are greater than min_supp value, then that items are considered frequent or if the items are less than min_supp value, then that items are to be considered as infrequent items.
The aim of association rule mining that was first introduced by [2] is to find or extract interesting relationships, correlations, associations or casual structures among sets of items either in transactional database or other data repository. These relationships are based upon the co-occurrences of each items within database instead of intrinsic properties of that items. The relationships or an association among these items are formed in terms of rule and typically known as association rule. There are 2 types of rules are frequent rule and infrequent rule. Both rules represent a different information in which domain of database that they are found. In most cases, events that are frequently occur may be less interesting than the events that are rarely occurring. This is based on the fact that frequent rules represent a previously known association, while the rare rules or patterns may carry the previously unknown (peculiar) association but might be useful to domain experts. An example in medical datasets, mining of frequent itemsets might discover the usual and normal pattern of diseases while mining of infrequent itemsets might disclose a relatively rare pattern of diseases such as Zika symptoms [5] instead of dengue or HFMD

disease instead of H5N1 disease. For a symptom such as vomiting, nausea, headache or blurry image of vision, the frequent and normal rule indicates a high blood pressure but the infrequent or rare rules might finds that patient is under anemia or could have been a low blood pressure.

3. Basic Principle of Rule

3.1. Association

The theoretical definition of the problem as illustrated by [2] is such that, I = {i_1, i_2,…i_k} for |k| > 0 be the set of items. A transaction database, D where each transaction is identified by its distinct and unique identifier, tid. Each T transaction contains a set of items, I such that $T \subseteq I$. An association rule is in the form of $X \subseteq Y$ where X is to indicate the antecedent part of the rule and Y is to indicate the consequent part of the rule where $X \subseteq I,\ Y \subseteq I$ and $X \cap Y = \emptyset$. A set of $X \subseteq I$ is called an itemset. An itemset that has *k*-items is called a k-itemset. The itemset that satisfies minimum support is called frequent itemset. The form of $X \Rightarrow Y$ holds in the transaction set D with confidence, c provided c% of D transactions contain X and Y. The form of $X \Rightarrow Y$ has support, s in the transaction set D provided s% of D transaction contains $X \cup Y$. The rule for support-confidence formula is given as:

a) The support of rule $X \Rightarrow Y$ is the division of both X and Y transactions with D.

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

where |*D*| is the total number of records in database

b) The confidence of rule $X \Rightarrow Y$ is the division of both X and Y support transactions with the support of X.

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

3.2. Definitions

Definition 3.1. (Mining of Frequent Itemset): A transaction database, Tn and a minimum support, s_{min}, the problem of finding all the frequent itemsets is called frequent itemset mining problem, derived by:
F(Tn, s_{min}*).*

Definition 3.2. (Frequent Itemset): A transaction database, Tn over an itemset, B and a minimum support, s_{min}. The set of all frequent itemsets is derived by:

$F(Tn, s_{min})=\{ X \subseteq B \mid sup\ (X) \geq s_{min}\}$

Definition 3.3. (Infrequent Itemset): A transaction database, Tn over an itemset, B and a minimum support, s_{min}. The set of all infrequent itemsets is derived by:

$F(Tn, s_{min})=\{ X \subseteq B \mid sup\ (X) \leq s_{min}\}$

Definition 3.4 (Tree Searching): A transaction database, Tn over an itemset, B, the set of all possible different generated itemsets over is derived by :
$B_{itemsets} = 2^{|B|}$

Definition 3.5 (Downward Closure Property/Support Monotonicity): Given a transaction database Tn over an itemset, B, let X, Y ⊆ B be two itemsets. Then,

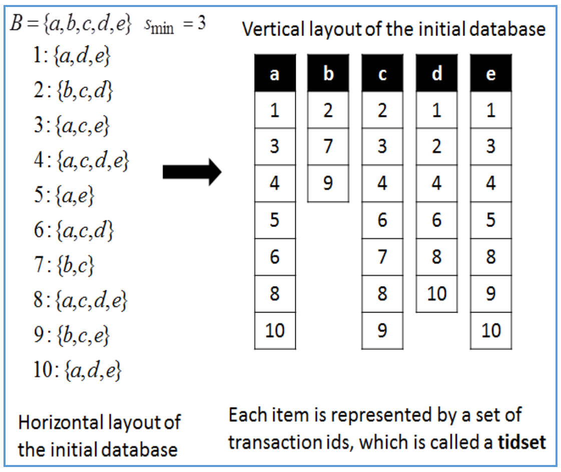
$X \subseteq Y \Rightarrow sup\ (Y) \leq sup\ (X)$

4. Data Representation

Data representation is critical in association rule mining. How data is stored in database, database layout and the searching strategy involved are all contribute to the performance of mining each itemsets.

4.1. Search Space in Database

The searching strategy is a crucial issue since it will affect the performance of itemset mining. The nature of how each items is stored (adjacency) is another issue. Most of Apriori-inspired version of algorithms such as [2, 6] works efficiently with sparse datasets because the frequency of patterns is short. But, when the frequency of patterns become long like in dense datasets that have huge different set of items i.e. live streaming dataset, the performance degrades drastically. The degradation is caused by many visits over database that increase in input and output overheads due to checking and pattern matching of large candidate sets. For x items, then proportionally 2^x – 2 additional frequent patterns to be examined by each algorithm. The crucial issue is to generate as few candidates as possible since the results of computing the supports is very much time consuming [9, 12]. Presumably, only frequent itemsets are generated and counted, but in real cases, the situation is on the other way around.



$B=\{a,b,c,d,e\}\quad s_{min}=3$

1:	$\{a,d,e\}$
2:	$\{b,c,d\}$
3:	$\{a,c,e\}$
4:	$\{a,c,d,e\}$
5:	$\{a,e\}$
6:	$\{a,c,d\}$
7:	$\{b,c\}$
8:	$\{a,c,d,e\}$
9:	$\{b,c,e\}$
10:	$\{a,d,e\}$

Horizontal layout of the initial database

Vertical layout of the initial database

a	b	c	d	e
1	2	2	1	1
3	7	3	2	3
4	9	4	4	4
5		6	6	5
6		7	8	8
8		8	10	9
10		9		10

Each item is represented by a set of transaction ids, which is called a **tidset**

Fig. 1: Database Representation

In Figure 1, from the horizontal layout view, the itemset in B that comprises of {a,b,c,d,e} and each itemsets is indexed with different and distinct (unique) transaction numbers (or transaction id or tids). When converting to column or vertical layout, each items {a,b,c,d,e} are then reordered in such a way where each items are grouped together with all corresponding tids (transaction number that contains the items) in vertical layout view. Then, the total number of tids implicitly disclose the support value of that particular items. Vertical point of layout speeds up the measuring of support or frequency occurrences of each items.

5. ECLAT and ECLAT-invariants

5.1. The Traditional Eclat (Eclat-tidset)

Equivalence Class Transformation or ECLAT is the first depth-first search algorithm for its searching strategy and the first vertical database format for its database layout structure [4, 11]. The algorithm minimizes input and output (I/O) cost by reducing the number of database scans to generate candidate as well as the efficiency of searching scheme. Its main operation is only simple intersection of transaction id set (tidsets). In fact, this is a contributing factor consuming the running time and memory usage of Eclat.

Eclat deploys prefix-based equivalence relation, θ_1 along with bottom up search that enumerates all frequent itemsets [10]. There are two main steps: *candidate generation* and *pruning*. Candidate generation is the enumeration process from {a} to relation of {ab} to {abc} to {abcd} until the relation of {abcde}, that is from itemset-1 until itemset-5, for {a,b,c,d,e}. Pruning is done through cutting or discarding the itemsets that is less that minimum support that depends upon user who specify the value. To the best of our knowledge, the vertical database structure with depth first searching (DFS) strategy conforms to efficient choices for achieving fast association rule mining results.

A few efforts in the literature which demonstrate vertical database layout [4, 11-13] dictates distinct advantages over horizontal database layout [2]. The first advantage is computing of supports of itemsets is much simpler since it involves only intersections of tids. Measuring of support also becomes faster since totak number of tids is an indicator of support. In contrast, multiple scan of database as well as tree data structures and functions are required for horizontal layouts [1-3]. The second advantage is to the reduced of database visits where only those selected candidates to the following visits of the mining process are written to text file. In vertical structure, each item i_k in the itemset B is represented as $i_k\colon \{i_k, t(i_k)\}$ and the initial transaction database consists of all items in the itemset. Either vertical or horizontal layouts or the combination of both layouts, it is possible to use the bit format to encode tids [6].

5.2. The Declat (Eclat-Diffset)

The dEclat [4, 14] or diffset is counting the difference set between 2 tidsets (i.e. tidset of the itemsets and its prefix as illustrated in Figure 2). Using diffset, the cardinality of sets representing itemsets is reduced significantly and this results in faster intersection and less memory usage. The prefix P is considered to contain the itemsets X and Y. Let *t(X)* denotes the tidset of X and *d(X)* denotes the diffset of X. When using tidset format, the *t(PX)* and *t(PY)* are available in the equivalence class and to obtain *t(PXY)*, the cardinality of *t(PX)* ∩ *t(PY)* = *t(PXY)* can be checked.

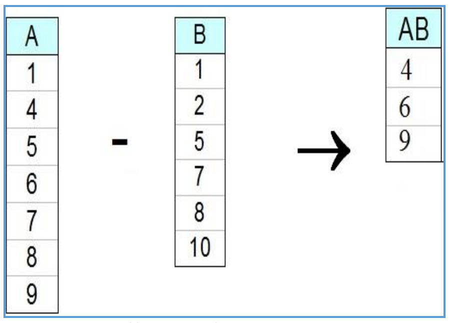


Fig. 2: Difference of itemset A and B

5.3. The Sorting Eclat (Eclat-Sortdiffset)

The sortdiffset (combination of tidsets + diffsets and sort) is introduced by [14] to enhance dEclat during switching condition. When switching process takes place, there exist tidsets which do not satisfy the switching condition, thus these tidsets remain as tidsets instead of diffset format. The situation results in both tidsets and diffsets format of itemsets in particular equivalence class and the next intersection process will involve both formats.

Conceptually, by given equivalence class with prefix *P* consisting of itemsets X_i in some order, intersection of PX_i with all PX_j with $j>i$ is to be performed in order to obtain a new equivalence class with prefix PX_i and frequent itemsets X_iX_j. PX_i and PX_j could be in either tidset or diffset format. If PX_i is in diffset format and PX_j is in tidset format, $d(PX_i) \cap t(PX_j) = d(PX_jX_i)$ which belongs to the equivalence class of prefix PX_j, not PX_i as expected. In other words, in order to do intersection between itemsets in diffset format and itemsets in tidset format to produce new equivalence classes properly, itemsets in tidset format must stand before itemsets in diffset format in the order of their equivalence class. That can be achieved by swapping (sorting) itemsets in diffset and tidset format, a process which has the complexity *O(n)* where *n* is the number of itemsets of the equivalence class.

6. Proposed Algorithm (Eclat-Postdiffset)

In postdiffset algorithm [8, 9], the association of itemsets is done according to tidset in the first loop and diffset in the later looping. In other words, the first level of looping is based on tidsets process, follows by the second level onwards of looping, the association are getting the result of diffset (difference intersection set) between ith column and i+1th column and save to db. Referring to Figure 3, the min_support threshold value is determined in terms of percentage where the user-specified min_support value will be divided by 100 and multiply with total rows (records) of each dataset. Then, starting with the first loop, the support value if mining of frequent itemset will be higher and/or equal to (>=) min_support (refer to line 5). In contrast, for mining of infrequent itemsets, the value of support must be less than (<) min_support (i.e. change line 5 to be < instead of >=).

```
Input: E((i_1, t_1), ... (i_n, t_n))|P), s_min
Output: F(E, s_min)
1.  start
    2.  //get min_support
    3.  min_supp=number_of_rows*percentage_min_supp
        ort;
    4.  run tidset for first loop;
    5.  if(support<=min_support){
    6.  add data to the next process;
    7.  add data into db
    8.  }
    9.  end tidset
    10. //for next loop
    11. start looping ;
    12. run diffset;
    13. if(support<=min_support){
    14. add data to the next process;
    15. add data into db
    16. }
    17. end looping.
    18. end diffset;
    19. write to file the value for current/last transaction da-
        ta;
20. end
```

Fig. 3: Postdiffset Pseudocode

7. Database Characteristics

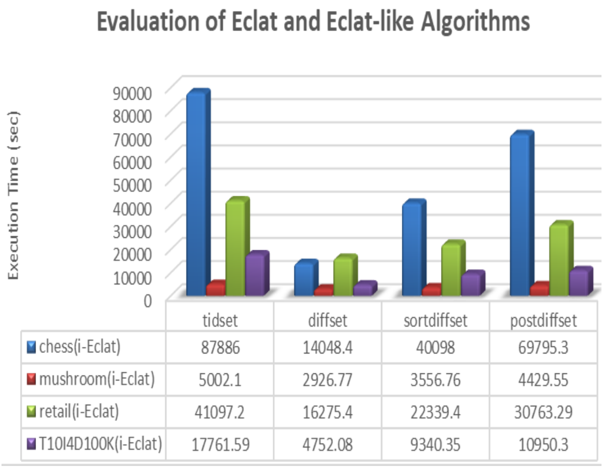
The datasets are selected in 2 different category i.e. dense and sparse datasets. Chess and mushroom are in dense category, while retail and T10I4D100K are in sparse category. Dataset chess is extracted from the chess end game positions for King and King Rook. The mushroom comprises of different attributes between 23 gilled mushroom species in the Agaricus and Lepiota family. The retail is obtained from anonymous Belgian retail store while the T10I4D100K dataset is derived from IBM Almaden Quest research group generator. All datasets are retrieved from [14]. The detail results and discussions prior to frequent and infrequent experimentation and testing are tabulated in section 8.

8. Experimentation Results and Discussion

Results of experimentation are depicted in two (2) different angle of pattern i.e. frequent (normal) pattern and infrequent (rare) pattern of results. Our proposed algorithm shows a contradiction of results in both patterns.

8.1. Frequent Itemsets of Chess, Mushroom, Retail and T10I4D100K

Figure 4 illustrates the results from frequent itemset mining of chess, mushroom, retail and T10I4D100K. In Figure 4, for chess, postdiffset with 69795.3 secs in execution time turns to be the third after diffset (20704.2 secs) and sortdiffset (20932.14 secs). Tidset is the last to perform with 87886 secs. Postdiffset performance poses a similar trend in mushroom, retail and T10I4D100K. The postdiffset has resulted in 4429.55 secs after diffset (2926.77 secs) and sortdiffset (3556.76 secs) in mushroom. The last falls to tidset with 5002.1 secs. Meanwhile, for sparse category, diffset performs with 16275.4 secs, followed with sortdiffset, postdiffset and tidset with 22339.4 secs, 30763.29 secs and 41097.2 secs respectively in retail. Sparse dataset of T10I4D100K also shows the same result's indicator with the best performance in execution time goes to diffset (4752.08 secs) and followed by sortdiffset (9340.35 secs), postdiffset (10950.3 secs) and tidset (17761.59 secs). In other words, postdiffset outperforms only in tidset with 21% in chess, 11% in mushroom, 25% I n retail and 38% in T10I4D100K. It loose its performance towards diffset and sortdiffset.

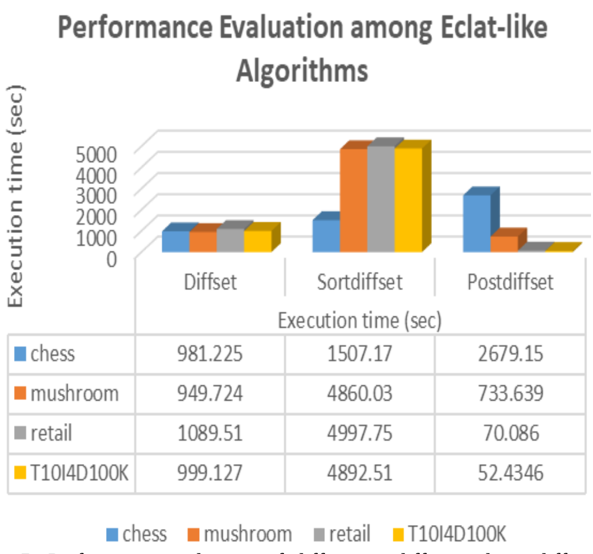


	tidset	diffset	sortdiffset	postdiffset
chess(i-Eclat)	87886	14048.4	40098	69795.3
mushroom(i-Eclat)	5002.1	2926.77	3556.76	4429.55
retail(i-Eclat)	41097.2	16275.4	22339.4	30763.29
T10I4D100K(i-Eclat)	17761.59	4752.08	9340.35	10950.3

Fig. 4: Performance evaluation of diffset, sortdiffset and postdiffset in frequent itemset of chess, mushroom, retail and T10I4D100K

8.2. Infrequent Itemsets of Chess, Mushroom, Retail and T10I4D100K

For infrequent itemset experimentation, we compare between diffset, sortdiffset and postdiffset since tidset always shows a highest ratio in execution time. Referring to Figure 5, in dense dataset, postdiffset lose its performance by 63% to diffset and 44% to sortdiffset in chess. But, in mushroom, postdiffset outperform with 23% in diffset and 84% in sortdiffset. For sparse dataset category, postdiffset tremendously outperform with 94% and 95% to diffset in retail and T10I4D100K. The algorithm continues to outperform tremendously in sortdiffset with 99% both in retail and T10I4D100K dataset.



	Diffset	Sortdiffset	Postdiffset
chess	981.225	1507.17	2679.15
mushroom	949.724	4860.03	733.639
retail	1089.51	4997.75	70.086
T10I4D100K	999.127	4892.51	52.4346

Fig. 5: Performance evaluation of diffset, sortdiffset and postdiffset in infrequent itemset of chess, mushroom, retail and T10I4D100K

9. Conclusion

The results of proposed algorithm (postdiffset) seems to contradict between mining in frequent and infrequent itemsets of dense dataset category (i.e. chess and mushroom) and sparse dataset category (i.e. retail and T10I4D100K). Postdiffset algorithm performs moderately when mining in frequent itemsets either in dense datasets of chess and mushroom, as well as in sparse datasets of retail and T10I4D100K. The performance of postdiffset turns to be in the third ranking out of 4 total algorithms (i.e. diffset, sortdiffset, postdiffset and tidset) in time execution measurement.
But, the performance of our proposed algorithm is contradicts when mining in infrequent itemsets of the same selected datasets. It turns to give the best execution time towards mushroom, retail and T10I4D100K datasets. Only it did not perform in chess since it executes with 2679.15 secs that depicts the longest time to execute. The reason could be the frequency of occurrences of each items in chess gives diffset and sortdiffset the better execution time as compared to postdiffset itself.
The results may conclude that there would be none best algorithm to suit for all experimented datasets due to the different nature of each dataset itself. Since our experimentation is dealing with only support measure as our interesting measure of itemsets, then in the future, we would focus on applying hybrid interestingness measure instead of only support and min_support such as Critical Relative Support (CRS) [14] to see the effects on execution time. We would also produce the association rules instead of only execution time.

Acknowledgement

We wish to thank internal grant of UNISZA (UniSZA/2018/DPU/11) for financial supporting our work. Also thanks to all team members in reviewing for spelling errors and synchronization consistencies and also for the constructive comments and suggestions.

References

[1] Koh, Y. S., & Ravana, S. D. (2016). Unsupervised rare pattern mining: A survey. *ACM Transactions on Knowledge Discovery from Data*, *10*(4), 45.

[2] Agrawal, R., & Srikant, R. (1994). Fast algorithms for mining association rules. *Proceedings of the 20th Int. Conf. Very Large Data Bases*, 1215, 487-499.

[3] Han, J., Pei, J., & Yin, Y. (2000). Mining frequent patterns without candidate generation. In *ACM Sigmod Record*, 29(2), 1-12.

[4] Zaki, M. J., & Gouda, K. (2003). Fast vertical mining using diffsets. *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 326-335.

[5] Sareen, S., Gupta, S. K., & Sood, S. K. (2017). An intelligent and secure system for predicting and preventing Zika virus outbreak using Fog computing. *Enterprise Information Systems*, *11*(9), 1436-1456.

[6] Shrivastava, S., & Johari, P. K. (2016, May). Analysis on high utility infrequent ItemSets mining over transactional database. *Proceedings of the IEEE International Conference on Recent Trends in Electronics, Information and Communication Technology*, pp. 897-902.

[7] Keste, P. A., & Shaikh, N. F. (2016). Improved approach for infrequent weighted itemsets in data mining. *Proceedings of the* IEEE *International Conference on Electrical, Electronics, and Optimization Techniques*, pp. 2663-2667.

[8] Jusoh, J. A., & Man, M. (2018). Modifying iEclat algorithm for infrequent patterns mining. *Advanced Science Letters*, *24*(3), 1876-1880.

[9] Zaki, M. J. (2000). Scalable algorithms for association mining. *IEEE Transactions on Knowledge and Data Engineering*, *12*(3), 372-390.

[10] Kotiyal, B., Kumar, A., Pant, B., Goudar, R. H., Chauhan, S., & Junee, S. (2013). User behavior analysis in web log through comparative study of Eclat and Apriori. *Proceedings of the* IEEE *7th International Conference on Intelligent Systems and Control*, pp. 421-426.

[11] Zheng, X., & Wang, S. (2014). Study on the method of road transport management information data mining based on pruning eclat algorithm and mapreduce. *Procedia Social and Behavioral Sciences*, *138*, 757-766.

[12] Trieu, T. A., & Kunieda, Y. (2012). An improvement for declat algorithm. *Proceedings of the* ACM *6th International Conference on Ubiquitous Information Management and Communication*, pp. 54.

[13] Abdullah, Z., Herawan, T., Ahmad, N., & Deris, M. M. (2011). Mining significant association rules from educational data using critical relative support approach. *Procedia Social and Behavioral Sciences*, *28*, 97-101.

[14] Frequent Itemset Mining Repository at http://fimi.ua.ac.be/.