International Journal of Electrical and Computer Engineering (IJECE)
Vol.8, No.6, December2018, pp. 4477~4485
ISSN: 2088-8708, DOI: 10.11591/ijece.v8i6.pp4477-4485

Postdiffset Algorithm in Rare Pattern: An Implementation via Benchmark Case Study

Mustafa Man[1], Wan Aezwani Wan Abu Bakar[2], Masita Masila Abd. Jalil[3], Julaily Aida Jusoh[4]
[1,3]School of Informatics & Applied Mathematics, Universiti Malaysia Terengganu, Malaysia
[2,4] Faculty Informatic and Computing, Universiti Sultan Zainal Abidin, Malaysia

Article Info

Article history:

Received Apr 10, 2018
Revised Jun 12, 2018
Accepted Jun 30, 2018

Keyword:

Association rule mining
Eclat algorithm
Frequent itemset
Infrequent itemset
Vertical databse

ABSTRACT

Frequent and infrequent itemset mining are trending in data mining techniques. The pattern of Association Rule (AR) generated will help decision maker or business policy maker to project for the next intended items across a wide variety of applications. While frequent itemsets are dealing with items that are most purchased or used, infrequent items are those items that are infrequently occur or also called rare items. The AR mining still remains as one of the most prominent areas in data mining that aims to extract interesting correlations, patterns, association or casual structures among set of items in the transaction databases or other data repositories. The design of database structure in association rules mining algorithms are 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. The efforts on horizontal format suffers in huge candidate generation and multiple database scans which resulting in higher memory consumptions. To overcome the issue, the solutions on vertical approaches are proposed. One of the established algorithms in vertical data format is Eclat.ECLAT or Equivalence Class Transformation algorithm is one example solution that lies in vertical database format. Because of its 'fast intersection', in this paper, we analyze the fundamental Eclat and Eclat-variants such asdiffsetand sortdiffset. In response to vertical data format and as a continuity to Eclat extension, we propose a postdiffset algorithm as a new member in Eclat variants that use tidset format in the first looping and diffset in the later looping. In this paper, we present the performance of Postdiffset algorithm prior to implementation in mining of infrequent or rare itemset. Postdiffset algorithm outperforms 23% and 84% to diffset and sortdiffset in mushroom and 94% and 99% to diffset and sortdiffset in retail dataset.

Corresponding Author:

Wan Aezwani Wan Abu Bakar,
Facultyof Informatic and Computing, Universiti Sultan Zainal Abidin,
Besut Campus, 22200 Besut, Terengganu, Malaysia.
Email: wanaezwani@unisza.edu.my

1. INTRODUCTION

The main objectives of association rules mining are to find the correlations, associations or casual structures among sets of items in the data repository. In other words, it allows non discovery of implicative and interesting tendencies in databases. Frequent itemset and infrequent itemset mining are critical fields in association rule mining. The fields are widely used across a variety of domains such as market basket analysis, remedial, biology, banking or retail services [1], [21]. Frequent or infrequent itemsets may contribute to big data generation. Undoubtedly, the critical issues regarding memory space consumption and data storage capacity will significantly effect prior to frequent or infrequent generation of itemsets [22], [23], [24]. The objective of frequent itemset is to find frequent grouping of items in database containing series of item transactions while the objective of infrequent itemset is contradict to frequent. All itemsets which has value that is greater than minimum support is called frequent itemsets.Infrequent itemset finds hidden association and correlation among rare itemsets. The rare consolidation of these itemsets may be interesting and gain more profit making. Rare cases have special concern since they represent significant difficulties for data mining algorithms. All itemsets which has the value that is lesser than minimum support is called infrequent itemsets. The idea of mining association rule originates from the analysis of market basket data [2]. Example of a 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.

Previous efforts on ARM have manipulated the traditional horizontal database format [2], [3]. Because of the persistent issues in storage and memory, later efforts turn to utilize on the vertical association rules mining algorithms [4]-[7]. The three basic models in frequent itemset mining are Apriori [7] that lies on horizontal format whereasEclat and FP-Growth [9], [11] underlying database structure is on vertical format.

Several workshave been conducted on vertical data association rules mining [3]-[6], [8], [10]-[12]. Among those efforts, Eclat algorithm is known for its 'fast' intersection of its tidlist whereby the resulting number of tids is actually the support (frequency) of each itemsets [4], [8]. That is, we should break off each intersection as soon as the resulting number of tids is below minimum support threshold that we have set. Studies on Eclat algorithm has attracted many developmentefforts including [5], [7], [13]. Motivated to its 'fast intersection', this paper presents a critical review in Eclat as well as to its variants. Our proposed solution, postdiffset algorithm performs moderately in selected dense dataset and good in selected sparse datasets.

2. RELATED WORKS

The Eclat stands for Equivalence Class Transformation [9], [12] takes a depth-first search and represents database in vertical layout such that each item is represented by a set of transaction IDs (called a tidset) whose transactions contain the item. Tidset of an itemset is generated by intersecting tidsets of its items. Because of the depth-first search, it is difficult to utilize the downward closure property like in Apriori. However, using tidsets has an advantage that there is no need for counting support, the support of an itemset is the size of the tidset representing it. The main operation of Eclat is intersecting tidsets, thus the size of tidsets is one of main factors affecting the running time and memory usage of Eclat. The bigger tidsets are, the more time and memory are needed.

Based upon discovery in [4], a new vertical data representation, called Diffset is proposed [5]. The so-called dEclat, adiffset of Eclat algorithm. Instead of using tidsets, they use the difference of tidsets (called diffsets). Using diffsets has reduced the set size representing itemsets dramatically and thus operations on sets are much faster. The dEclat has shown to achieve significant improvements in performance as well as memory usage over Eclat, especially on dense databases. However, when the dataset is sparse, diffset loses its advantage over tidset. Therefore, the researchers suggested using tidset format at the start for sparse databases and then switching to diffset format later when a switching condition is met.

As a continuity in [4], [5], a novel approach for vertical representation where in the authors used the combination of tidset and diffset and sorted the diffset in descending order to represent databases [7]. The technique is claimed to eliminate the need of checking the switching condition and converting tidset to diffset format regardless of database condition either sparse or dense. Besides, the combination can fully exploit the advantages of both tidset and diffset format where the prelim results have shown a reduction in average diffset size and speed of database processing.

3. ASSOCIATION RULE THEORETICAL BACKGROUND

Following is the formal definition of the problem defined in [3]. Let I = $\{i_1, i_2, \ldots i_m\}$ for $|m| > 0$ be the set of items. D is a database of transactions where each transaction has a unique identifier called tid. 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 represent the antecedent part of the rule and Y represents the consequent part of the rule where $X \subseteq I,\ Y \subseteq I$ and $X \cap Y = \emptyset$.A set $X \subseteq I$ is called an *itemset*. The itemset that satisfies minimum support is called frequent itemset. 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.
The confidence of rule $X \Rightarrow Y$ is the fraction of transactions in D containing X that also contain Y.

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

A rule is *frequent* if its support is greater than minimum support (min_supp) threshold. The rules which satisfy minimum confidence (min_conf) threshold is called *strong rule* and both *min_supp* and *min_conf* are user specified values [4].

4. REPRESENTATION OF DATA

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.

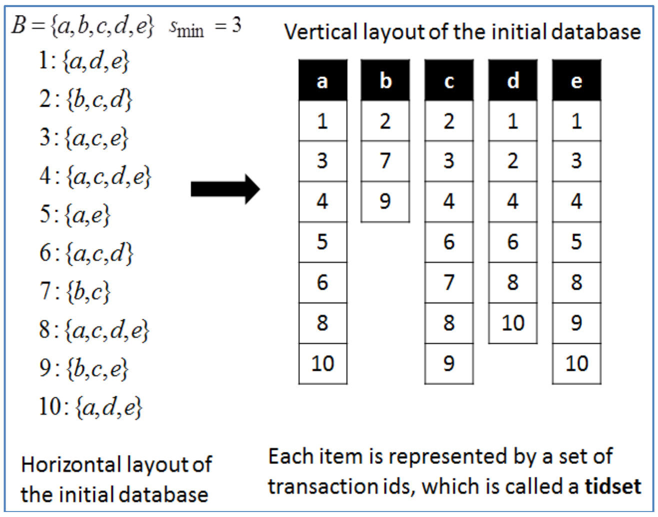
4.1. Search Space and Database Issues

Either with horizontal data format or vertical data format, one must take into account on the search space strategy employment regardless the database condition of whether it is sparse database or dense database. The Apriori-inspired algorithms [5] perform well with sparse datasets such as market basket data when the frequent patterns are short. But, when the frequent patterns are long with dense datasets such as bioinformatics and telecommunication, the performance degrades drastically.

The degradation is caused by many passes over the database that automatically incurs I/O overheads and it is computationally expensive in checking large sets of candidates by pattern matching. For m items, there could imply $2^m–2$ additional frequent patterns that will explicitly examined by each algorithms. It is important to generate as few candidates as possible since computing the supports is time consuming [14]. As the best case, only frequent itemsets are generated and counted, unfortunately, the idea is impossible in general.

4.2. Horizontal Verses Vertical Layouts

In the horizontal layout, each transaction T_i is represented as T_i: (tid, I) where tid is the transaction identifier and I is an itemset containing items occurring in the transaction. The initial transaction consists of all transactions T_i.In the vertical layout, each item i_k in the item base B is represented as i_k: $\{i_k, t(i_k)\}$ and the initial transaction database consists of all items in the item base. For both layouts, it is possible to use the bit format to encode tids and also a combination of both layouts can be used [7], [8]. Figure 1 illustrates horizontal and vertical layout of data representation by [7].The items in B consist of {a,b,c,d,e} and each itemsets are allocated with unique identifiers (tids) for each transactions. This is clearly visualized in horizontal format. To switch to vertical format, every items {a,b,c,d,e} are then organized where all items are allocated with their corresponding tids. When this is done, it is clearly visualized the support of each items through the counting number of every item's tids.


$B = \{a,b,c,d,e\}$ $s_{min} = 3$
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
Each item is represented by a set of transaction ids, which is called a **tidset**

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

Figure 1. Horizontal and vertical layout

5. DESIGN OF ECLAT AND ECLAT-LIKE ALGORITHMS

There are two main steps: candidate generation and pruning. In candidate generation, each *k-itemset* candidate is generated from two frequent (k-1)-itemsets and its support is counted, if its support is lower than the threshold, then it will be discarded, otherwise it is frequent itemsets and used to generate (k+1)-itemsets. Since Eclat uses the vertical layout, counting support is trivial. Depth-first searching strategy is done where it starts with frequent items in the item base and then 2-itemsets from 1-itemsets, 3-itemsets from 2-itemsets and so on.

5.1. Traditional Eclat (Tidset)

A*k*-itemset is generated by taking union of two *(k-1)*-itemsets which have *(k-2)* items in common, the two (*k-1*)-itemsets are called parent itemsets of the k-itemset. Fox example, {, {ab} and {ac} are parent of {abc}. To avoid generating duplicate itemsets, (*k-1*)-itemsets are sorted in some order. To generate all possible k-itemsetsfrom a set of (k-1)-itemsets sharing*(k-2)*-items, union operation is conducted of a *(k-1)*-itemsetswith the itemsets that stand behind it in the sorted order, and this process takes place for all (k-1)-itemsets except the last one. For example, from a set of $\{a, b, c, d, e\}$, which share 0 item, then this could be sorted into alphabet order. To generate all 2-itemsets, the union of {*a*} with *{b,c,d,e}* will result into 2-itemsets *{ab,ac,ad,ae}*, then for the union of *{b}* with *{c,d,e}* will result in *{bc,bd,be}*, similarly for *{c}* and *{d}*. Finally, all possible 2-itemsets *{ab,ac,ad,ae,bc,bd,be,cd,ce,de}* is generated to get all possible 3-itemsets until the rest of the number of possible itemsets.

Eclat starts with prefix {} and the search tree is actually the initial search tree. To divide the initial search tree, it picks the prefix {a}, generate the corresponding equivalence class and does frequent itemset mining in the sub tree of all itemsets containing {a}, in this sub tree it divides further into two sub trees by picking the prefix {ab}: the first sub tree consists of all itemset containing {ab}, the other consists of all itemsets containing {a} but not {b}, and this process is recursive until all itemsets in the initial search tree are visited. The search tree of an item base {a,b,c,d,e} is represented by the tree as shown in Figure 2.

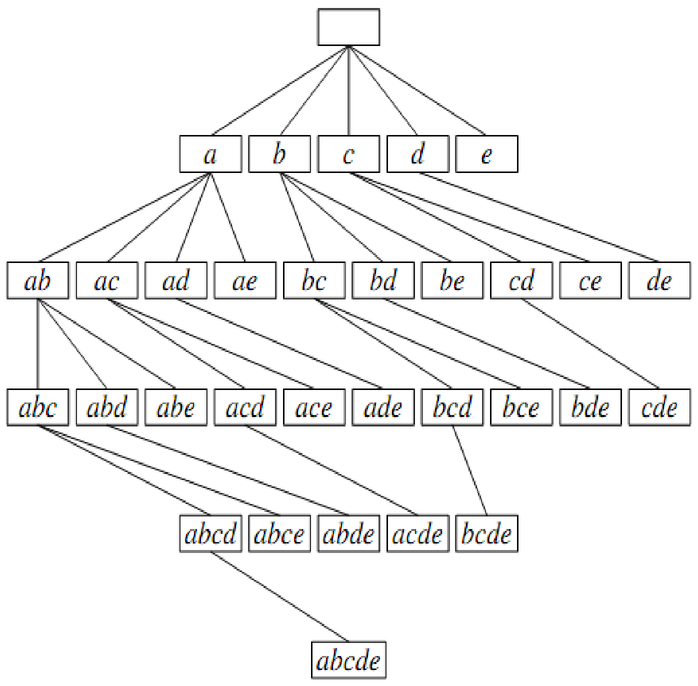


Figure 2. Search tree for {a,b,c,d,e} with null set

Figure 3 illustrates of detail steps taken in Eclat algorithm when assuming that the initial transaction database is in vertical layout and represented by an equivalence class E with prefix {}. It uses prefix-based equivalence class along with bottom-up search. Frequent itemsets are generated by intersecting tidlist of all distinct pairs of atoms (i.e. $i_{1..}i_n$) and checking the cardinality of the tidlist. This process is repeated until all frequent itemsets are enumerated.

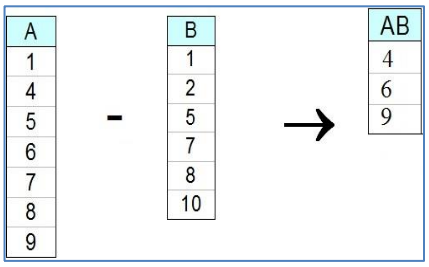
```
$Input: E((i_1, t_1), \dots (i_n, t_n))|P), s_{min}$
$Output: F(E, s_{min})$
1: $for\ all\ i_j\ occuring\ in\ E\ do$
2:     $P := P \cup i_j$ // add $i_j$ to create a new prefix
3:     $init(E')$ // initialize a new equivalence class with the new prefix P
4:     $for\ all\ i_k\ occuring\ in\ E\ such\ that\ k > j\ do$
5:         $t_{tmp} = t_j \cap t_k$
6:         $if\ |t_{tmp}| \geq s_{min}\ then$
7:             $E' := E \cup (i_k, t_{tmp})$
8:             $F = F \cup (i_k \cup P)$
9:         $end\ if$
10:    $end\ for$
11:    $if\ E' \neq \{\}\ then$
12:        $Eclat(E', s_{min})$
13:    $end\ if$
14: $end\ for$
```

Figure 3. Pseudocode for Eclat algorithm

5.2. dEclat (Diffset)

The dEclat (different set or diffset) is proposed by [5] where the authors represent an itemset by tids that appear in the tidset of its prefix but do not appear in its tidsets. In abbreviation, diffset is the difference between two (2) tidsets (i.e. tidset of the itemsets and its prefix). Through diffset, the cardinality of sets representing itemsets is reduced rigorously and that contributes in faster intersection and less memory usage. Consider an equivalence class with prefix P contains the itemsets X and Y [7]. Let t(X) denotes the tidset of X and d(X) denotes the diffset of X. When using tidset format, we will have t(PX) and t(PY) available in the equivalence class and to obtain t(PXY) we check the cardinality of $t(PX) \cap t(PY) = t(PXY)$.

When using diffset format, we will have *d(PX)* instead of *t(PX)* and $d(PX) = t(P) - t(X)$, the set of tids in t(P) but not in t(X). Similarly, we have d(PY) = t(P) – t(Y). So the support of PX is not the size of its diffset. By the definition of *d(PX)*, it can be seen that $|t(PX)| = |t(P)| - |t(P) - t(X)| = |t(P)| - |d(PX)|$. In other word, $\sup(PX) = sup(P) - |d(PX)|$. Refer to the illustration in Figure 4.



A
1
4
5
6
7
8
9

-

B
1
2
5
7
8
10

→

AB
4
6
9

Figure 4. Difference of itemset A and B

To use diffset format, the initial transaction database in vertical layout is firstly converted to diffset format in which diffset of items are sets of tids whose transactions do not contain items. This is deduced from the definition of diffset, the initial transaction database in vertical layout is an equivalence with the prefix *P={}*, so the tidset of *P* includes all tids, all transactions contain *P*, and the diffset of an item *i*is $d(i) = t(P) - t(i)$, this is a set of tids whose transactions do not contain *i*. From this initial equivalence class, we could generate all itemsets with their diffsets and supports. The dEclat is different from Eclat in step 5, instead of generating a new tidset, a new diffset is generated. The performance and memory usage of dEclat has shown to achieve significant improvements over traditional Eclat (tidset) especially in dense database. But when database is sparse, it loses its advantages over tidsets. Then in [5] the authors suggested to use tidset format at starting for sparse database and later switch to diffset format when switching condition is met. From this starting point, postdiffset is proposed.

5.3. Com-Eclat (Sortdiffset)

The com-Eclat (combination of tidsets + diffsets and sort) is introduced by [7] 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.

6. POSTDIFFSET ALGORITHM

Postdiffset is designed prior to suggestion that is made in [5] to use tidset format at starting for sparse database and later switch to diffset format when switching condition is met. Conceptually, by given equivalence class with prefix P consisting of itemsetsX_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 prefixPX_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.

In postdiffset algorithm, the first level of looping is based on tidsets process, follows by the second level onwards of looping are getting the result of diffset (difference intersection set) between ith column and i+1th column and save to db. Referring to Figure 5, 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 in each loop, starting with the first loop, if the support is greater than or equal (>=) to min_support, then, in postdiffset, the first level of looping is based on tidsets process, follows by the second level onwards of looping are getting the result of diffset (difference intersection set) between ith column and i+1th column and save to database.

```
Input: $E((i_1, t_1), \dots (i_n, t_n))|P), s_{min}$
Output: $F(E, s_{min})$
1.  start
    2.  //get min_support
    3.  min_supp=number_of_rows*percentage_min_support;
    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. flush value for current/last transaction data;
20. end
```

Figure 5. Postdiffset pseudocode

In Figure 5, the min_support is measured basedon the multiplication of the number of rows of itemsets in database with the user specified percentage of min_support. Then, each itemset is intersected with its transaction id (tid) for the first looping. If the support of each itemset*is less than or equal to* the min_support (itemset in this condition is very rare as to indicate the itemset of abnormal and peculiar cases), then that itemset is passed to the second level of looping. Starting from secondlooping onwards, each tids is intersected with its difference set (diffset) until finish. The experimentation of postdiffset algorithm is presented in the next section.

7. EXPERIMENTATION

All experiments are performed on a Dell N5050, Intel ® Pentium ® CPU B960 @ 2.20 GHz with 8GB RAM in a Win 7 64-bit platform. The software specification for algorithm development is deployed using open source software i.e. MySQL version 5.6.20 – MySQL community server (GPL) for our database server, Apache/2.4.10 (Win32) OpenSSL/1.0.1i PHP/5.5.15 for our web server, php as a programming language and phpMyAdmin with version 4.2.7.1, the latest stable version as to handle the administration of MySQL over the Web. The phpMyAdmin[91] is a free software tool written in PHP, that supports a wide range of operations on MySQL, MariaDB and Drizzle. The database characteristics is shown in Table 1.

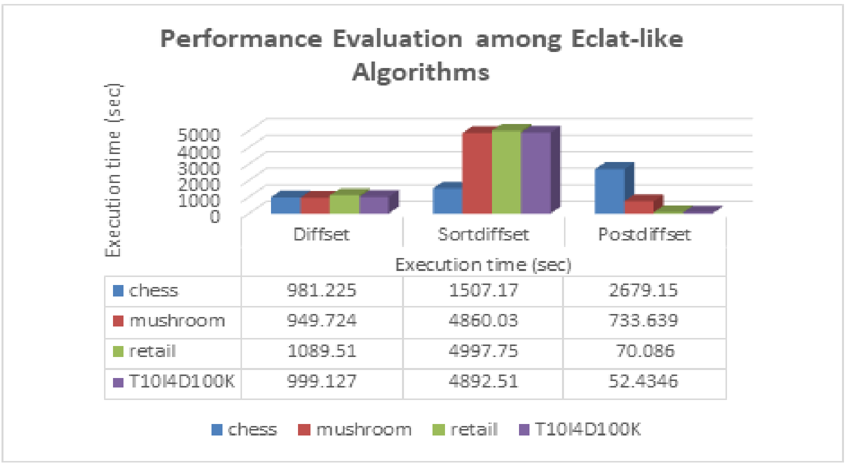
Table 1. Database Characteristics

Datasets	Num. of Transactions	Length (Attribute)	Size (KB)	Category
Chess	3196	37	335	Dense
Mushroom	8125	43	558	Dense
Retail	88162	68	5143	Sparse
T40I10D100K	100001	32	15116	Sparse

7.1. Empirical Results

For the ease and fast experimentation purposes, we have modified datasets to be only thousand rows of item sets that are randomly processed for mining purposes. Our experimentation is with regards to dEclat (diffset), com-Eclat (sortdiffset) and postdiffset algorithm because from our past experimentation on postdiffset implementation in frequent itemset mining, the results of traditional-Eclat (tidset) will always be the last in performance and memory usage among those three (3) algorithms. Figure 6 shows the graph of performance evaluationwith regards to execution time (in second) within four (4) datasets i.e. chess, mushroom, retail and T10I4D100K.

Referring to Figure 6, in dense dataset, postdiffset lose its performance by 63% to diffset and 44% to sortdiffset in chess. In mushroom, postdiffset outperform with 23% in diffset and 84% in sortdiffset. For sparse dataet category, postdiffset tremendously outperform with 94% and 95% to diffset in retail and T10I4D100K. The algorithm continues to outperform dramatically 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

Figure 6. Performance on diffset, sortdiffset and postdiffset in chess, mushroom, retail and T10I4D100K

8. CONCLUSION AND FUTURE DIRECTION

The performance of postdiffset varies depending upon datasets. It is best executed in sparse datasets i.e. retail and T10I4D100K while in dense datasets i.e. chess, it loses its reputation towards diffset and sortdiffset. But in mushroom, postdiffset did well among the other two (2) algorithms. The simple conclusion can be made where the nature of datasets in terms of how many time the occurrence of itemsets could me one of the contributing factor to the overall performance of certain association rule infrequent mining algorithms. Our next focus could be the enforcement of confidence level or other interenstingness measure towards itemsets rather than just focusing on minimum support value.

ACKNOWLEDGEMENTS

We wish to thank all faculty members for supporting our work in reviewing for spelling errors and synchronization consistencies and also for the meaningful comments and suggestions.

REFERENCES

[1] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules", in *Proceedings of 20th International Conference on Very Large Data Bases (VLDB)*, 1215, 487–499, 1994.
[2] R. Agrawal, et al., "Mining association rules between sets of items in large databases", *ACM SIGMOD Record*, 22(2), 207–216, 1993.
[3] J. Han, et al., "Mining frequent patterns without candidate generation", *ACM SIGMOD Record*, 29(2), 1–12, 2000.
[4] M.J. Zaki, et al.,"New algorithms for fast discovery of association rules", *In Proceedings of the ACM SIGKDD international conference on Knowledge Discovery and Data Mining (KDD'97)*, 283–286, 1997.
[5] M.J. Zaki and K. Gouda, "Fast vertical mining using diffsets", *In Proceedings of the ninth ACM SIGKDD international conference on Knowledge Discovery and Data Mining*. 326–335, 2003.
[6] P. Shenoy, et al., "Turbo-charging vertical mining of large databases", *ACM SIGMOD Record*, 29(2), 22–33, 2000.
[7] T.A, Trieu and Y. Kunieda, "An improvement for declat algorithm", In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication* (ICUIMC'12), 54, 1–6, 2012.
[8] J. Hipp, et al., "Algorithms for association rule mininga general survey and comparison", *ACM SIGKDD Explorations Newsletter*, 2(1), 58–64, 2000.
[9] J. Han, et al., "Frequent pattern mining: current status and future directions", *Data Mining and Knowledge Discovery*, 15(1), 55–86, 2007.
[10] C. Borgelt, "Efficient implementations of apriori and eclat", In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations* (FIMI03), 2003.
[11] L. Schmidt-Thieme, "Algorithmic features of eclat", In *Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations* (FIMI04), 2004.
[12] M.J. Zaki, "Scalable algorithms for association mining", *IEEE Transactions on Knowledge and Data Engineering*, 12(3), 372–390, 2000.
[13] X.Yu and H. Wang, "Improvement of eclat algorithm based on support in frequent itemset mining", *Journal of Computers*, 9(9), 2116–2123, 2014.
[14] B. Goethals, "Frequent set mining", In *Data Mining and Knowledge Discovery Handbook*, Springer, 321–338, 2010
[15] Borgelt, C.; and Kruse, R.; "Induction of association rules: Apriori implementation", In *Compstat*, Springer, 395–400, 2002.
[16] A. Savasere, et al., "An efficient algorithm for mining association rules in large databases", In *Proceeding of the 21th International Conference on Very Large Data Bases* (VLDB '95), 432–444, 1995.
[17] H. Toivonen, "Sampling large databases for association rules", In *Proceeding of the 22nd International Conference on Very Large Data Bases* (VLDB '96), 134–145, 1996.
[18] J. Han, et al., "Mining frequent patterns without candidate generation: A frequent-pattern tree approach", *Data Mining and Knowledge Discovery*, 8(1), 53–87, 2004.
[19] T. Slimani and A. Lazzez, "Efficient analysis of pattern and association rule mining approaches", *International Journal of Information Technology and Computer Science*, 6(3), 70–81, 2014.
[20] M. Man, et al., "Spatial information databases integration model", In A.A. Manaf*et a*l. (Eds.): ICIEIS 2011, *Informatics Engineering and Information Science*, Springer, 77–90, 2011.
[21] S. Shrivastava and P.K. Johari, "Analysis on high utility infrequent ItemSets mining over transactional database", *In Recent Trends in Electronics, Information & Communication Technology (RTEICT), IEEE International Conference* on pp. 897-902, 2016.
[22] M.A. Thalor and S. Patil, "Incremental Learning on Non-stationary Data Stream using Ensemble Approach", *International Journal of Electrical and Computer Engineering*, Aug 1; 6(4): 1811, 2016.
[23] G. Bathla, et al., "A Novel Approach for clustering Big Data based on Map Reduce", *International Journal of Electrical and Computer Engineering (IJECE)*, Jun 1; 8(3), 2018.
[24] M.B. Man, et al., "Mining Association Rules: A Case Study on Benchmark Dense Data", *Indonesian Journal of Electrical Engineering and Computer Science* on pp. 546-553, Sep 1; 3(3), 2016.

BIOGRAPHIES OF AUTHORS

Wan Aezwani Bt Wan Abu Bakar received her PhD in Computer Science at Universiti Malaysia Terengganu (UMT) Terengganu in Nov, 2016. Her focus area is in association rule in frequent itemset mining. She received her master's degree in Master of Science (Computer Science) from Universiti Teknologi Malaysia (UTM) Skudai, Johor in 2000 prior to finishing her study in Bachelor's degree also in the same stream from Universiti Putra Malaysia (UPM) Serdang, Selangor in 1998. Her master's research was formerly on Fingerprint Image Segmentation in the stream of Image Processing. Now she's pursuing her research towards association relationship in infrequent itemset mining which is more downstream to educational data settings.

Mustafa Man is an Associate Professor in School of Informatics and Applied Mathematics and also as a Deputy Director at Research Management Innovation Centre (RMIC), UMT. He started his PhD studies in July 2009 and finished his studies in Computer Science from UTM in 2012. He has received Computer Science Diploma, Computer Science Degree, Masters Degree from UPM. In 2012, he has been awarded a "MIMOS Prestigious Awards" for his PhD by MIMOS Berhad. His research is focused on the development of multiple types of databases integration model and also in Augmented Reality (AR), android based, and IT related into across domain platform.

Masita @ Masila Abdul Jalil received her B.Eng (Hons) in Computer System Engineering from the University of Warwick, UK in 1997. After graduated, she joined CELCOM (M), one of the leading telecommunication providers in Malaysia as a system engineer. She later pursued her Master study in Engineering Business Management at the same university before joining Universiti Malaysia Terengganu (UMT) as a lecturer in 2001. In 2012, she obtained her PhD in Information Technology from Universiti Kebangsaan Malaysia (UKM). Her current research interests include software reuse, computer science education and computer applications in forensics.

Julaily Aida Jusoh received her B.Eng (Hons) in Software Engineering from the Universiti Putra Malaysia (UPM), Selangor in 2004. After graduated, she furthered her Master study in Software Engineering in Universiti Malaysia Terengganu (UMT) in 2005. In 2009, she joined Universiti Sultan Zainal Abidin (UNISZA) as a lecturer. Now, she furthered her PhD studies in Universiti Malaysia Terengganu (UMT) since September 2016. She currently works in infrequent itemset mining using Eclat Algorithm for her PhD research. Her current research interests include software engineering, formal methods and pattern mining.