International Journal of Engineering & Technology, 7 (2.28) (2018) 197-199

International Journal of Engineering & Technology

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

Research paper

Postdiffset: an Eclat-like algorithm for frequent itemset mining

W.A.W.A. Bakar[1]*, M.A. Jalil[2], M. Man[2] , Z. Abdullah[3], F. Mohd[2]

[1]*Faculty Informatic and Computing, Universiti Sultan Zainal Abidin, Besut Campus, 22200 Besut, Terengganu, Malaysia*
[2] *School of Informatics and Applied Mathematics, Universiti Malaysia Terengganu, Kuala Terengganu, Terengganu, Malaysia*
[3]*Centre of Computing & Informatics (CCI), UMK, City Campus, Kota Bharu, Kelantan*
**Corresponding author E-mail: wanaezwani@unisza.edu.my*

Abstract

Frequent itemset mining is a major field in data mining techniques. This is because it deals with usual and normal occurrences of set of items in a database transaction. Originated from market basket analysis, frequent itemset generation may lead to the formulation of association rule as to derive correlation or patterns. Association rule mining still remains as one of the most prominent areas in data mining that aims to extract interesting correlations, frequent patterns, association or casual structures among set of items in the transaction databases. Underlying structure of 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 works on horizontal approaches suffer in many candidate generation and multiple database scans that contributes to higher memory consumptions. In response to improve on horizontal approach, the works on vertical approaches are established. Eclat algorithm is one example of algorithm in vertical approach database format. Motivated to its 'fast intersection', in this paper, we review and analyze the fundamental Eclat and Eclat-variants such as tidset, diffset, and 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. We present the performance of postdiffset results in time execution as to indicate some improvements has been achieved in frequent itemset mining.

Keywords: *Association rule mining, data mining, eclat algorithm, frequent itemset, vertical data format.*

1. Introduction

Association rules (AR) mining is one of the important and advanced techniques in data mining. Originates from market analysis, 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 AR mining, frequent itemset is the field dealing with normal frequent occurrences of data items. The objective is to find frequent grouping of items in database containing series of item transactions. The database composes series of basket that are analogous to orders placed by customers. These orders are actually individual baskets made up of some number of items. Giant companies such as Trivago, Amazon and Netflix and other online distributors make use of frequent itemsets to project for an additional item that customer might want to purchase based on their purchasing history. The projection for additional items to suggest for customers is based on the association rule generated. 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 [1,2].
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 [4,5]. The three basic models in frequent itemset mining are Apriori [1,2] that lies on horizontal format whereas Eclat [4,5] and FP-Growth [6] underlying database structure is on vertical format.
Several efforts on vertical data association rules mining have been conducted [3,4]. Among them, 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]. 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 development efforts including [5,7,8].

2. Related works

The Eclat is the abbreviation of equivalence class transformation and the acronym for equivalence class clustering and bottom up Lattice Traversal [7,8]. It takes a depth-first search for its searching strategy and adopts a vertical layout to represent databases, in which 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.
Prior to findings in [4], the author then proposed a new vertical data representation, called Diffset [5]. The so-called dEclat, a

diffset of Eclat algorithm. Instead of using tidsets, they use the difference of tidsets (called diffsets). 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* [6]. 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 $t(PX) \cap t(PY) = t(PXY)$. 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.
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 wherein the authors used the combination of tidset and diffset and sorted the diffset in descending order to represent databases [6]. The technique (com-Eclat) 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. 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.

3. Basic principles

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. An itemset with *k*-items is called a k-itemset. The itemset that satisfies minimum support is called frequent itemset. The rule $X \Rightarrow Y$ holds in the transaction set *D* with confidence *c* if c% of transactions in *D* that contain *X* also contain *Y*. The rule $X \Rightarrow Y$ has support s in the transaction set *D* if s% of transaction in *D* contains $X \cup Y$. Illustration of support-confidence framework is given as below:

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

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

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*.

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

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]. An association rule is considered interesting if it satisfies both min_supp and min_conf thresholds [7..6]. If an itemset is infrequent, then all of its supersets are infrequent as well. Therefore, in algorithms that exploit this property, only candidates where all of its subsets are frequent are generated and counted for supports. Similarly, the set of infrequent itemsets is upward closed.

4. Proposed algorithm

Diffset is shown to achieve significant improvements in performance and memory usage over traditional Eclat (tidset) especially in dense database. When database is sparse, diffset 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. 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 1, the min_support threshold value is determined in terms of percentage where the user-specified min_support value is divided by 100 and times the total rows (records) of each dataset. Then, starting with the first loop, if the support is greater than or equal (>=) to min_support, then, the first level of looping is based on tidsets process, whereas 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: $E((i_1, t_1), \ldots (i_n, t_n))|P), s_{min}$
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;
```

Fig. 1: Pseudocode for Postdiffset Algorithm

5. Experimental settings

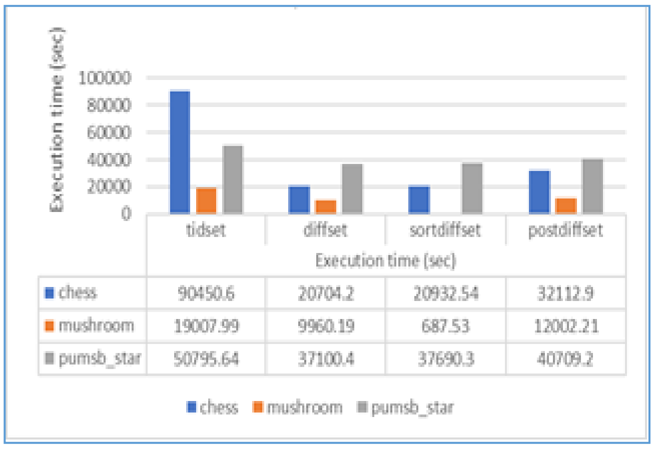
All experiments are performed on a Dell N5050, Intel ® Pentium ® CPU B960 @ 2.20 GHz with 8GB RAM in a Win 7 64-bit platform. The software specification for algorithm development is deployed using open source software i.e. MySQL version 5.6.20 – MySQL community server (GPL) for our database server, Apache/2.4.10 (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 database of chess, mushroom and pumsb_star is extracted from online frequent itemset mining dataset repository (http://fimi.ua.ac.be/data/). We initiate our experimentation with testing in dense dataset category. 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
Pumsb_star	49047	50	11028	Dense

5.1. Empirical results

The experimentation is done with regards to Eclat-tidset, Eclat-diffset, Eclat-sortdiffset and Eclat-postdiffset algorithm. Figure 2 shows the graph of performance result in execution time (in second) within three datasets i.e. chess, mushroom and pumb_star.



	tidset	diffset	sortdiffset	postdiffset
	Execution time (sec)			
🟦 chess	90450.6	20704.2	20932.54	32112.9
🟧 mushroom	19007.99	9960.19	687.53	12002.21
▨ pumsb_star	50795.64	37100.4	37690.3	40709.2

Fig. 2: Performance on tidset, diffset, sortdiffset and postdiffset in chess, mushroom and pumsb_star

In chess, postdiffset turns to be the third after diffset (20704.2 secs) and sortdiffset (20932.14 secs) that results in 32112.9 secs. Tidset is the last to perform with 90450.6 secs. In mushroom and pumb_star datasets, quite similar trends happen to postdiffset. For mushroom, postdiffset has resulted in 12002.21 secs after sortdiffset (687.53 secs) and diffset (9960.19 secs). The last falls to tidset with 19007.99 secs. Meanwhile for pumsb_star, the first in execution time falls to diffset with 37100.4 secs. Then it follows by sortdiffset with 37690.3 secs. Then postdiffset turns to be the third with 40709.2 secs while the last is tidset with 50795.64 secs.

6. Conclusion and future direction

Even though our proposed algorithm is found to be moderate in performance of time execution, but it shows better trends than Eclat-tidset. The probability of itemset occurrences in each database would be the contributing factor to the algorithm performance. In our next discovery, we would see our postdiffset performance in benchmark sparse dataset such as T10I4D100K and retail. We may also test our proposed algorithm in infrequent itemsets, to discover either showing a similar results trends or the results of infrequent itemsets may contradict with our recent frequent itemsets. The research and future direction of ARM especially in frequent pattern mining is discovered and disclosed by [6] on whether we can derive a compact but high quality set of patterns which can be valuable and useful in applications. The issue is yet to being explored and solved [9] before frequent pattern mining can become a cornerstone approach in data mining applications.

Acknowledgement

We wish to thank MyBrain15 for financial support and all faculty members for supporting our work in reviewing, proof reading and providing constructive comments and suggestions. This study has been also supported in part of the Fundamental Research Grant Scheme (FRGS) with vot number 59394, under the Malaysia Ministry of Higher Education (MOHE) and Research Development Grant (GPP), under Universiti Malaysia Terengganu (UMT).

References

[1] Agrawal R, & Srikant R. “Fast algorithms for mining association rules”, *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*, Vol.12, No.15, (1994), pp:487–499.
[2] Agrawal R, Imielinski T, & Swami A (1993), Mining association rules between sets of items in large databases. *ACM SIGMOD Record*, 22(2), 207–216.
[3] Han J, Pei J, & Yin Y (2000), Mining frequent patterns without candidate generation. *ACM SIGMOD Record*, 29(2), 1–12.
[4] Zaki MJ, Parthasarathy S, Ogihara M, Li W, et al. “New algorithms for fast discovery of association rules”, *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD’97)*, (1997), pp:283–286.
[5] Zaki MJ, & Gouda K. “Fast vertical mining using diffsets”, *Proceedings of The Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, (2003), pp:326–335.
[6] Han J, Cheng H, Xin D, & Yan X (2007), Frequent pattern mining: current status and future directions. *Data Mining and Knowledge Discovery*, 15(1), 55–86.
[7] Trieu TA, & Kunieda Y. “An improvement for declat algorithm”, *Proceedings of The 6th International Conference on Ubiquitous Information Management and Communication (ICUIMC’12)*, Vol. 54, (2012), pp: 1–6.
[8] Yu X, & Wang H (2014), Improvement of Eclat algorithm based on support in frequent itemset mining. *Journal of Computers*, 9(9), 2116–2123.
[9] Man M, Rahim MSM, Zakaria MZ, & Bakar WAWA (2011), Spatial information databases integration model. *Informatics Engineering and Information Science. Springer*, 77–90.