A NUMERICAL METHOD FOR FREQUENT PATTERNS MINING

NORWATI MUSTAPHA, MOHAMMAD-HOSSEIN NADIMI-SHAHRAKI, ALI B MAMAT, MD. NASIR B SULAIMAN

Department of Computer Science, Faculty of Computer Science and Information Technology,

University of Putra Malaysia, Selangor, 43400, Malaysia

Email:{norwati,ali,nasir}@fsktm.upm.edu.my , nadimi@ieee.org

ABSTRACT

Frequent pattern mining is one of the active research themes in data mining. It plays an important role in all data mining tasks such as clustering, classification, prediction, and association analysis. Identifying all frequent patterns is the most time consuming process due to a massive number of patterns generated. A reasonable solution is identifying maximal frequent patterns which form the smallest representative set of patterns to generate all frequent patterns. In this paper, an efficient numerical method for mining frequent patterns is proposed. This method is based on prime number characteristics to generate all frequent patterns by using maximal frequent ones. There are two new properties introduced in this method; a novel tree structure called PC_Tree and PC_Miner algorithm. The PC_Tree is a simple tree structure but yet capable to capture the whole of transactions information with an efficient data transformation technique that utilizes the prime number theory. The PC_Miner algorithm traverses the PC_Tree by using an efficient pruning technique. The experimental results verify the compactness and the efficiency of mining shown by the proposed method.

Keywords: *Data Mining, Frequent Pattern, Maximal frequent pattern, Data Transformation.*

1. INTRODUCTION

The explosive growth of many business, government and scientific databases has far outpaced human ability to interpret and digest this data. Data mining therefore appears as a tool to address the need of sifting useful information such as hidden patterns from databases.

As shown in Fig 1, data mining is an essential step in the process of knowledge discovery in databases (KDD) to extract data patterns. It is a composite process of multiple disciplines including statistics, database systems, machine learning, intelligent computing and information technology.

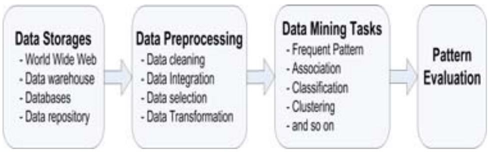


Fig1. KDD Process

Since the introduction of the Apriori algorithms [2], frequent pattern mining is one of the active research themes in data mining. It covers a broad spectrum of data mining tasks including clustering, classification, prediction, and association analysis. Frequent patterns are itemsets or substructures that exist in a data set with frequency no less than a user specified threshold. Many algorithms have been introduced to solve the problem of frequent pattern mining more efficiently. They are almost based on three fundamental frequent patterns mining methodologies: Apriori, FP-growth and Eclat [8].

The Apriori-based algorithms almost suffer from multiple database scan and candidate generation problem. The FP-growth method needs twice database scan. There have been introduced some efficient FP-growth extensions which need only once data base scan [12]. However, they almost need a large amount of memory to fit the tree structure which used in their method. The Eclat uses a Boolean power set lattice that needs much space to store the labels and tid-lists as well.

Identifying all frequent patterns is the most time consuming process due to a massive number of patterns generated and this is the problem that faced by the above mentioned algorithms. For n items in the domain of a transaction database, there are $O(2^n)$ candidate pattern which should be computed their frequency to find frequent patterns. A reasonable solution is identifying maximal frequent patterns which form the smallest representative set of patterns to generate all frequent patterns [10, 13]. In this paper, an efficient numerical method for mining frequent patterns is proposed. Particularly, our method introduces a new method based on prime number characteristics to find completed frequent patterns by using maximal frequent patterns. However, this numerical approach can be extended for all data mining tasks. It is an improvement of previous method that has been proposed by us for maximal frequent pattern mining [13]. However the tree structure, search space pruning technique, and traversing technique have mostly been changed in mining algorithm. The proposed method includes an efficient data transformation technique, a novel tree structure called Prime-based encoded and Compressed Tree or PC_Tree and also PC_Miner algorithm. Our method needs only once database scan. The data transformation technique utilizes prime number theory and transforms all items existing in each transaction into only a positive integer called Transaction Value (TV). The experimental results show that the size of data set can be reduced by using this technique dramatically. The PC_Tree is a novel and simple tree structure but yet efficient to capture whole of transactions information by keeping only their transaction values. The PC_Miner algorithm traverses the PC_Tree by using an efficient pruning technique to find the maximal frequent patterns which form completed set of the frequent patterns. An experimental analysis has comprehensively been conducted on the performance of the proposed method. The experimental results verify the accuracy and efficiency of the proposed method.

The rest of this paper is organized as follows. Section 2 introduces the problem and reviews some efficient related works. The proposed method is described in section 3. The experimental results and evaluation show in section 4. Finally, section 5 contains the conclusions and future works.

2. PROBLEM AND RELATED WORK

Frequent patterns are itemsets or substructures that exist in a dataset with frequency no less than a user specified threshold.

2. 1 Problem Description

Let $L= \{i_1, i_2 \dots i_n\}$ be a set of items. Let D be a set of database transactions where each transaction T is a set of items and |D| be the number of transactions in D. Given $P= \{i_j \dots i_k\}$ be a subset of L ($j \le k$ and $1 \le j, k \le n$) is called a pattern. The support of a pattern P or S (P) in D is the number of transactions in D that contains P. Pattern P will be called frequent if its support is no less than a user specified support threshold min_sup σ ($0 \le \sigma \le |D|$). The problem of frequent pattern mining is finding all frequent patterns (FP) from dataset D with respect to specified min_sup σ.

In many real applications especially in dense data with long frequent patterns enumerating all possible $2^L - 2$ subsets of an L length pattern is infeasible [5]. A reasonable solution is identifying a smaller representative set of patterns from which all other frequent patterns can be derived. Maximal frequent patterns (MFP) form the smallest representative set of patterns to generate all frequent patterns. In particular, the MFP are those patterns that are frequent but none of their supersets are frequent. The problem of maximal frequent patterns mining is finding all MFP in D with respect to σ.

The complexity of frequent patterns mining from a large amount of data is generating a huge number of patterns satisfying the minimum support threshold, especially when min_sup σ is specified low. This is because, all sub-pattern of a frequent pattern are frequent as well. Therefore a long pattern contains a number of shorter frequent sub-patterns. Various kinds of frequent patterns can be mined from different kinds of data sets. In this research, we use itemsets (sets of items) as a data set and the proposed method is for frequent itemset mining, that is, the mining of frequent from transactional data sets. However, it can be extended for other kinds of frequent patterns.

2.2 Related work

The Apriori is a basic algorithm for finding frequent patterns. It has been followed by several variations for improving efficiency and scalability. They almost suffer inherently from two problems; multiple database scans that are costly and generating lots of candidates [6].

Han et al. [9] proposed frequent pattern tree or FP-Tree as a prefix-based tree structure, and an algorithm called FP-growth. The FP-Tree stores only the frequent items in a frequency-descending order. The highly compact nature of FP-tree enhances the performance of the FP-growth. The FP-Tree construction requires two data scans. The FP-growth unlike the Apriori algorithm mines the complete set of frequent patterns without candidate generation. The experimental results showed that FP-Tree and almost all its extensions have a high compactness rate for dense data set. However, they need a large amount of memory for sparse data set where probability for sharing common paths is low [6, 12].

The presentation of data which will be mined is an essential consideration in almost all algorithms. The mining algorithms can be classified according to two horizontal and vertical database layouts. Both the Apriori and FP-growth methods use horizontal data format (i.e., {TID: itemset}) to mine frequent patterns. Zaki [15] proposed Eclat algorithm or Equivalence CLASS Transformation by using the vertical data format (i.e., {item: TID_set}). The Eclat uses the lattice theory to represent the database items. The results showed that the Eclat outperforms Apriori significantly. However, it needs an additional conversion step. This is because most databases use a horizontal format. Moreover, it uses a Boolean power set lattice that needs to much space to store the labels and tid-lists.

As said, to cope with the complexity of frequent patterns mining problem, our method generates FP from MFP. Many efficient algorithms have been introduced to solve the problem of maximal frequent pattern mining [10, 13]. Mostly, they traverse a search space to find MFP. The key to be an efficient traversing is the pruning techniques which can remove some branches in the search space. The pruning techniques used in efficient algorithms can be categorized into two groups:

Subset frequency pruning: the all subsets of any frequent pattern are pruned because they can not be maximal frequent pattern.

Superset infrequency pruning: the all supersets of any infrequent pattern are pruned because they can not be frequent pattern.

The Pincer-Search algorithm [10] uses horizontal data layout. It combines a bottom-up and a top down techniques to mine the MFP. However search space is traversed without an efficient pruning technique. The MaxMiner algorithm [3] uses a breadth-first technique to traverse of the search space and mine the MFP. It makes use of a look ahead pruning strategy to reduce database scanning. It prunes the search space by both subsets frequency and supersets infrequency pruning. The DepthProject [1] finds MFP using a depth first search of a lexicographic tree of patterns, and uses a counting method based on transaction projections. The DepthProject demonstrated an efficient improvement over previous algorithms for mining MFP. The Mafia [5] extends the idea in DepthProject. It uses a search strategy that has been improved by an effective pruning mechanism. The MaxMiner, DepthProject and Mafia use Rymon's set enumeration [14] to enumerate all the patterns. Thus these algorithms avoid having to compute the support of all the frequent patterns.

The Flex [11] is a lexicographic tree designed in vertical layout to store pattern P and list of transaction identifier where pattern P appears. Its structure is restricted test-and-generation instead of Apriori-like is restricted generation-and-test. Thus nodes generated are certainly frequent. The Flex tree is constructed in depth-first fashion. The experimental results showed the Flex is an efficient algorithm to find long and representative patterns MFP. However, it needs a large amount of memory especially to store the list of transaction identifier. This makes Flex is impossible to be fit in memory by one database scan as for huge number of frequent patterns generated.

3. PROPOSED METHOD

3.1. Data Transformation Technique

The presentation of database is an efficient consideration in almost all algorithms. The most commonly database layout is the horizontal and vertical layout [15]. In both layouts, the size of database is very large. As shown in Fig 1 the data transformation is an essential process in data preprocessing step which can reduce the size of database. Obviously, reducing of the size of database can enhance performance of mining algorithms. Our method uses a prime-based data transformation technique to reduce the size of transaction database. It transforms each transaction into a positive integer called Transaction Value (TV) during of the PC_Tree construction as follows: Given transaction T = (tid, X) where tid is the transaction-id and X = $\{i_j \ldots i_k\}$ is the transaction-items or pattern X. While the PC_Tree algorithm reads transaction T, the transformer procedure considers a prime number p_r for each item i_r in pattern X, and then TV_{tid} is computed by

Eq. (1) where T= (tid, X), X = {i_j … i_k} and i_r is presented by p_r.

$$TV_{tid} = \prod_{j}^{k} p_r \qquad (1)$$

Therefore, all transactions can be represented in a compacted layout using this transformation technique. In fact the transformer is a numerical encoder which hides transaction information. The transformation technique utilizes Eq. (1) based on simple following definition.

"A positive integer N can be expressed by unique product N = $p_1^{m_1}\, p_2^{m_2} \ldots p_r^{m_r}$ where p_i is prime number, $p_1 \prec p_2 \prec \cdots p_r$ and m_i is a positive integer, called the multiplicity of p_i " [7].

For example, N = 1800=$2^3 * 3^2 * 5^2$. Conceptually, there is no duplicated item in transaction T. Hence we restrict the multiplicity only to m_i = 1 without losing any significant information. Therefore N can be produced by $P_1 P_2 \cdots P_r$.

To facilitate the understanding of the transformation process used in our method, let's examine it through an example. Let item set L= {A, B, C, D, E, F} and the transaction database, DB, be the first two columns of Table 1 with eight transactions. As shown in the fourth column of Table 1, DB can be presented by TVs which very smaller than original transactions.

TABLE 1
The transaction database DB and its Transaction Values

TID	Items	Transformed	TV
1	A, B, C, D, E	2, 3, 5, 7, 11	2310
2	A, B, C, D, F	2, 3, 5, 7, 13	2730
3	A, B, E	2, 3, 11	66
4	A, C, D, E	2, 5, 7, 11	770
5	C, D, F	5, 7, 13	455
6	A, C, D, F	2, 5, 7, 13	910
7	A, C, D	2, 5, 7	70
8	C, D, F	5, 7, 13	455

Obviously, the compactness rate for real data can be more than synthetic data used in the experiments. This is because; the size of the TV used for a transaction is almost independent of kind of dataset, but the average length of items in real datasets is bigger than in synthetic dataset. For example third transaction T= (*3, {A, B, E}*) in Table 1 presents good identification numbers bought by a customer; T_O= (3, {55123450, 55123452, 55123458}) from a market. Although the length of items in T_O is bigger than T but both T and T_O can be transformed into the same TV 66 by using our data transformation technique. Hence it is an item-length independent transformation technique. The experiments showed that by applying this data transformation technique, the size of real transaction database can be reduced more than half.

3.2. PC_Tree Construction

Using tree structure in mining algorithms makes two possibilities to enhance the performance of mining. Firstly, data compressing by a well-organized tree structure like FP-tree. Secondly, reducing search space by using pruning techniques. Thus the tree structures have been considered as a basic structure in previous data mining research [8, 11, 12]. This research introduces a novel tree structure called PC_Tree (Prime-based encoded and Compressed Tree). Unlike the previous methods, the PC_Tree is based on prime number characteristics which can makes use of both possibilities data compressing and pruning techniques to enhance efficiency.

A PC_Tree includes of a root and some nodes that formed sub trees as children of the root or descendants. The node structure consisted mainly of several different fields: value, local-count, global-count, status and link. The value field stores TV to records which transaction represented by this node. The local-count field set by 1 during inserting current TV and it is increased by 1 if its TV and current TV are equal. The global-count field registers support of pattern P which presented by its TV.

In fact during of insertion procedure the support of all frequent and infrequent patterns is registered in the global-count field. It can be used for interactive mining where min_sup is changed by user frequently [12]. The status field is to keep tracking of traversing. When a node visited in the traversing procedure the status field is changed from 0 to 1. The link field is to form sub trees or descendants of the root.

Fig. 2 shows step by step construction of PC_Tree for transactions shown in table 1. The construction operation mainly consists of insertion procedure that inserts TV(s) into PC_Tree based on definitions below:

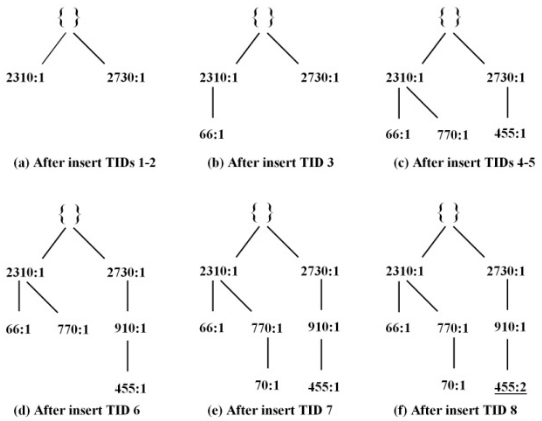
Definition 1: If TV of node n_r and n_s is equal then $r = s$. Insertion procedure increases local -

count field of node n_r by 1 if the current TV is equal with TV of n_r.

Definition 2: $R = (n_i, n_{i-1} \cdots n_j, root)$ is a descendant iff TV of node $n_r \in R$ $(i \le r \le j)$ can divide all TVs kept in nodes $R_r = (n_{r+1}, n_{r+2}, \cdots n_j, root)$.

Definition 3: TV of the root is assumed null and can be divided by all TVs.

Fig. 2(a) shows insertion of the first and second transactions. The second TV with value 2730 can not be divided by the first TV with value 2310 and it creates a new descendant using definition 2 and 3. Transactions 3-7 are inserted into their descendant based on definition 2 shown in Figure 2(b)-(e). The TV of eighth transaction with value 455 is equal with the fifth TV, and then the local-count field of fifth TV is increased by 1 using definition 1 shown in Figure 2(f) (shown in underline).



(a) After insert TIDs 1-2 (b) After insert TID 3 (c) After insert TIDs 4-5

(d) After insert TID 6 (e) After insert TID 7 (f) After insert TID 8

Fig. 2 Step by step PC_Tree construction

Each TV in PC_Tree represents a pattern P and the support of pattern P or S (P) is registered in the global-count field. Given pattern P and Q have been presented by TVP and TVQ respectively, the PC_Tree has some nice below properties:

Property 1: The S (P) is computed by traversing only descendants of TVP.

Property 2: P and Q belong to descendant R and S (P) < S (Q) iff TVP can be divided by TVQ.

Property 3: Important procedures are almost done only by two simple mathematic operations product and division. Obviously using mathematic operation enhances the performance instead of string operation.

3.3. PC_Miner Algorithm

As explained in previous section, during of insertion each TV in the PC_Tree, the following procedures are done.

a) Item-frequency counting.
b) Local-counting.
c) Global-counting.
d) Descendant constructing.

The PC_Miner algorithm traverses the completed PC_Tree to discover the MFP in top-down direction. There is no need to database scan again, because all information about items and patterns are stored in the PC_Tree. However Figure 2(f) as a completed PC_Tree didn't show some information like global-count stored in the tree. The miner algorithm makes use of a combined pruning strategy including both superset infrequency and subset frequency pruning. As a result the search space is reduced, which dramatically reduces the computation and memory requirement and enhances the efficiency. The superset infrequency pruning is assisted by the frequency of items computed during the PC-Tree construction. Table 2 shows the item frequency and considered prime number for transaction database shown in Table 1.

TABLE 2
Frequency of items and supposed prime numbers

Item	Prime number	Item Frequency
A	2	6
B	3	3
C	5	7
D	7	7
E	11	3
F	13	4

4. EXPERIMENTAL RESULTS

In this section, we evaluate the performance of our method. All experiments were performed in a time-sharing environment in a 2.4 GHz PC. All the algorithms are implemented using C++.

In first experiment we use synthetic sparse datasets T10I4D100k generated by the program developed at IBM Almaden Research Center [2]. The number of transactions, the average transaction length and the average frequent pattern length of T10I4D100k are set to 100k, 10 and 4 respectively.

We consider *p*% of this dataset where *p* will be increased from 10 to 100 to evaluate how the data transform technique can compact the size of dataset. Fig.4 shows comparison of the size of original dataset with the size of transformed dataset using our data transform technique.

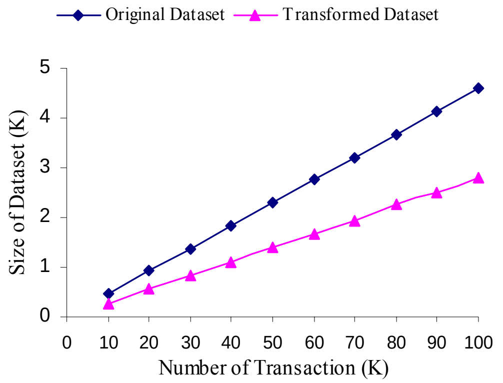


Fig.4. Compactness of data transformation technique

In second experiment, we show the accuracy and correctness of the method. This experiment is conducted by using T10I4D100k and real dense dataset mushroom from UCI Irvine Machine Learning Repository [4] The Mushroom dataset records consists of the characteristics of various mushroom species, and the number of records, the number of items and the average record length are set to 8124, 119 and 23 respectively. Fig. 5 and 6 show the numbers of frequent patterns discovered for the tests at varying min_sup on this datasets. The results verify that our method can find all frequent patterns.

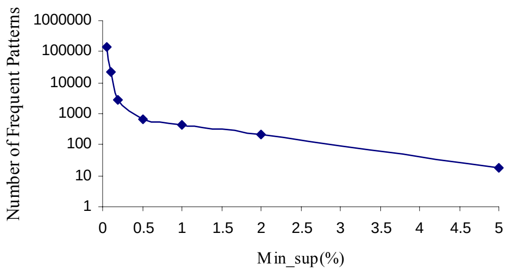


Fig.5. Number of Frequent Patterns in T10I4D100k

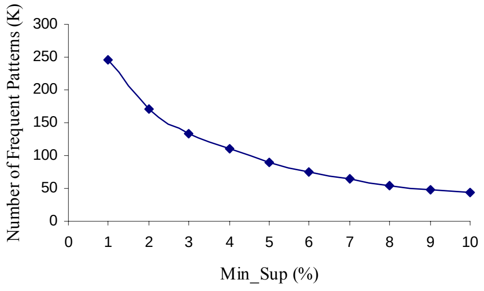


Fig.6. Number of Frequent Patterns in Mushroom

Third experiment is to compare the performance of the PC-Miner with the Flex algorithms on the dataset T10I4D100k. To allow a fair comparison of algorithms, firstly find all MFP by using PC_Miner and Flex separately. Then all FP is generated by same procedure run in cached mode. Fig. 7 shows the PC_Miner algorithm outperforms the Flex algorithms.

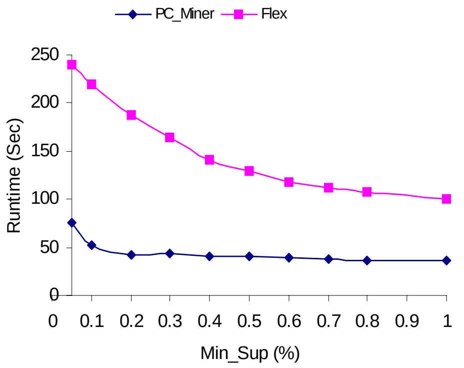


Fig.7. Performance of PC-Miner vs. Flex

5. CONCLUSION AND FUTURE WORKS

In this paper, we proposed a numerical method to mine all frequent patterns efficiently. Our method introduced an efficient data transformation technique, a novel tree structure called Prime-based encoded and Compressed Tree or PC_Tree and also PC_Miner algorithm. The experiments verified the compactness of the data transformation technique. The PC_Tree presented well-organized tree structure with nice properties to capture transaction information. The PC_Miner reduced the search space using a combined pruning strategy to traverses the PC_Tree efficiently. The experimental results showed the PC_Miner algorithm outperforms the Flex algorithms.

Particularly, our method introduces a new method based on prime number characteristics to find completed frequent patterns by using maximal frequent patterns. However, the numerical approach can be extended for all data mining tasks. For example, it can be applied in incremental mining of frequent patterns where database transactions are inserted, deleted, and/or modified. In addition, it can also be used for interactive mining of frequent patterns where minimum support threshold can be changed to find new correlation between patterns.

REFERENCES

[1] Agarwal R. C., C. C. Aggarwal, and V. V. V. Prasad, Depth First Generation of Long Patterns, *sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 108-118, 2000.

[2] Agrawal R. and R. Srikant, Fast Algorithms for Mining Association Rules, *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, vol. 1215, pp. 487499, 1994.

[3] Bayardo Jr R. J., Efficiently Mining Long Patterns from Databases, *ACM SIGMOD international conference on Management of data*, pp. 85-93, 1998.

[4] Blake C. and C. Merz, Uci Repository of Machine Learning Databases. University of California – Irvine, Irvine, Ca, 1998.

[5] Burdick D., M. Calimlim, and J. Gehrke, Mafia: A Maximal Frequent Itemset Algorithm for Transactional Databases, *17th International Conference on Data Engineering*, pp. 443-452, 2001.

[6] Ceglar A. and J. F. Roddick, Association Mining, *ACM Computing Surveys (CSUR)*, vol. 38, 2006.

[7] Cormen T. T., C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*: MIT Press Cambridge, MA, USA, 1990.

[8] Han J., H. Cheng, D. Xin, and X. Yan, Frequent Pattern Mining: Current Status and Future Directions, *Data Mining and Knowledge Discovery*, vol. 15, pp. 55-86, 2007.

[9] Han J., J. Pei, Y. Yin, and R. Mao, Mining Frequent Patterns without Candidate Generation: A Frequent-Pattern Tree Approach, *Data Mining and Knowledge Discovery*, vol. 8, pp. 53-87, 2004.

[10] Lin D. I. and Z. M. Kedem, Pincer-Search: A New Algorithm for Discovering the Maximum Frequent Set, *Advances in Database Technology--EDBT'98: 6th International Conference on Extending Database Technology, Valencia, Spain.*, 1998.

[11] Mustapha N., M. N. Sulaiman, M. Othman, and M. H. Selamat, Fast Discovery of Long Patterns for Association Rules, *International Journal of Computer Mathematics*, vol. 80, pp. 967-976, 2003.

[12] Nadimi-Shahraki M.H., N. Mustapha, M. N. Sulaiman, and A. Mamat, Incremental Updating of Frequent Pattern: Basic Algorithms, *Proceedings of the second International Conference on Information Systems Technology and Management (ICISTM 08)*, pp. 145-148, 2008.

[13] Nadimi-Shahraki M.H., N. Mustapha, M. N. B. Sulaiman, and A. B. Mamat, A New Method for Mining Maximal Frequent Itemsets, presented at International IEEE Symposium on Information Technology, 2008. ITSim 2008., Malaysia, 2008, pp. 309-312.

[14] Rymon R., Search through Systematic Set Enumeration, *Third International Conference on Principles of Knowledge Representation and Reasoning*, pp. 539-550, 1992.

[15] Zaki M. J., Scalable Algorithms for Association Mining, *IEEE TRANSACTIONS ON KNOWLEDGE AND DATA ENGINEERING*, pp. 372-390, 2000.