Maximal Frequent Itemsets Mining Using Database Encoding

Mohammad Nadimi-Shahraki, Norwati Mustapha, Md Nasir B Sulaiman, Ali B Mamat

***Abstract*—Frequent itemsets mining is a classic problem in data mining and plays an important role in data mining research for over a decade. However, the mining of the all frequent itemsets will lead to a massive number of itemsets. Fortunately, this problem can be reduced to the mining of maximal frequent itemsets. In this paper, we propose a new method for mining maximal frequent itemsets. Our method introduces an efficient database encoding technique, a novel tree structure called PC_Tree and also PC_Miner algorithm. The database encoding technique utilizes Prime number characteristics and transforms each transaction into a positive integer that has all properties of its items. The PC_Tree is a simple tree structure but yet powerful to capture whole of transactions by one database scan. The PC_Miner algorithm traverses the PC_Tree to mine maximal frequent itemsets. Experiments verify the efficiency and advantages of the proposed method.**

***Index Terms*—Database encoding, Frequent itemsets, Maximal frequent itemsets, Prime numbers.**

I. Introduction

Since the introduction of the Apriori algorithms [2], frequent pattern mining plays an important role in data mining research. The problem of mining all frequent itemsets is that if there is a large frequent itemset with size L, then almost all 2^L candidate subsets of the items might be generated. There are many contributions to enhance performance of mining all frequent itemsets. They have been mostly done base on three basic frequent itemsets mining methodologies: Apriori, FP-growth and Eclat [8].

In real application, the number of frequent itemsets produced from a transaction database can be very huge and it becomes impossible to find all frequent itemsets [5]. A reasonable solution is identifying a small representative set of itemsets from which all other frequent itemsets can be derived. Since frequent itemsets are upward closed, it is sufficient to find out only all maximal frequent itemsets (MFI). In particular, maximal frequent itemsets are those itemsets that are frequent but none of their supersets are frequent. Maximal frequent itemsets provide a compact representation of frequent itemsets effectively. In fact, they form the smallest set of itemsets from which all frequent itemsets can be derived [4].

In this research, we propose a new method to discover maximal frequent itemsets. Our method introduces an efficient database encoding technique, a tree structure called Prime-based encoded and Compressed Tree or PC_Tree and also PC_Miner algorithm. The database encoding technique utilizes prime numbers characteristics and transforms transaction database to a flat file called encoded file which each transaction presents by a positive integer.

The experiments shows that by applying this database encoding technique, the size of transaction database can be reduced more than half. The PC_Tree is a novel and simple tree structure but yet efficient consists of a root and some sub trees as the children of the root to capture whole of transactions by one database scan. The PC_Miner algorithm traverses the PC_Tree and prunes search space using the PC_Tree properties. It builds gcd (greatest common divisor) set of the nodes of the tree which are in search space to mine maximal frequent itemsets.

The rest of the paper is organized as follows. Section 2 introduces the basic concepts and reviews some related works. The PC_Tree and the PC_Miner are described in section 3. The experimental results show in section 4 and section 5 contains some conclusions and future works.

II. Preliminaries and Related Work

Let DB be a transaction database and X be the set of items from 1 to n. An itemset X is frequent if it contains at least σ transactions, where σ is the minimum support. An itemset X is a maximal frequent itemset if it is a frequent itemset and no superset of it is also a frequent itemset.

Mohammad Nadimi-Shahraki is with Department of Computer Engineering of Islamic Azad University, Najafabad branch, Iran and he is currently pursuing the Ph.D. degree in computer science from the University of Putra Malaysia, Emails: nadimi.mh@gmail.com.

Assistant Prof. Dr. Norwati Mustapha, Associate Prof. Dr. Md Nasir B Sulaiman, and Associate Prof. Dr. Ali B Mamat are with Faculty of Computer Science and Information Technology, University of Putra Malaysia (UPM), 43400 UPM, Selangor, Malaysia. Emails: {norwati,nasir,ali}@fsktm.upm.edu.my.

When the frequent patterns are long, mining all frequent itemsets is infeasible because of the exponential number of frequent itemsets. Researchers now turn to find Maximal Frequent Itemsets (MFI) [3]. Because mining of MFI is faster and all frequent itemsets can be built up from MFI and can be counted for support in a single scan of the database. Moreover, we can focus on any part of the MFI to do supervise data mining. Many efficient algorithms have been developed for mining maximal frequent itemsets in static database that mostly using the horizontal or vertical database layout.

The Pincer-Search algorithm [9] uses horizontal data format. It didn't construct the candidates in a bottom-up manner like Apriori. The Pincer-Search algorithm combines a bottom-up and a top down techniques to find the maximal frequent itemsets. The bottom up process finds frequent itemsets, and non frequent itemsets. Then non frequent itemsets are used by a top down process to refine a set of potential maximal frequent itemsets. MaxMiner [4] is another algorithm for finding the maximal elements. It uses a breadth-first traversal of the search space; and reduces database scanning by using a look ahead pruning strategy. DepthProject demonstrated an order of magnitude improvement over previous algorithms for mining maximal frequent itemsets [1]. Both DepthProject and Mafia [5] mine a superset of the MFI, and require a post-pruning to eliminate non-maximal patterns. Recently a new two-way-hybrid algorithm for mining maximal frequent itemsets has been proposed [6]. A flexible two-way-hybrid search method is given. The two-way-hybrid search begins the mining procedure in both the top-down and bottom-up directions at the same time. Moreover, information gathered in the bottom-up can be used to prune the search space in the other top down direction. The experiments showed that pruning strategies are implied in this method, can reduce the original search space.

III. The Proposed Method

In this research, we proposed a new method to discover maximal frequent itemsets efficiently. Our method introduces an efficient database encoding technique, a tree structure called PC_Tree and also PC_Miner algorithm.

A. The Database Encoding

The presentation and encoding of database is an important consideration in almost all algorithms. The most commonly used layout is the horizontal database layout and vertical one [12]. In both layouts, the size of the database is very large. Reducing of the size of the transaction database can enhance performance of mining algorithms.

In our algorithm, transaction database is transformed to a new presentation by database encoding in a data pre-processing phase. The database encoding is a useful technique which can reduce the size of database and improve the efficiency of mining. In this research, instead of maintaining a large table in the transaction database, one encoded file is considered. The encoded file is a flat file created by the database encoding technique as a new presentation of transaction database.

The encoded file includes positive integers called Value. Every Value presents the entire items that occur in the transaction. In fact, all items in one transaction are converted into only one positive integer that has all properties of these items. Our database encoding technique makes use of prime number characteristics.

An integer P is a prime integer or prime number if $P>1$ and only positive divisors of P are 1 and P. A positive integer N can be expressed as a product of prime numbers and this factorization is unique except for the order of the factors. Let $p_1, p_2 \ldots p_r$ be the *distinct* prime factors of N, so that $p_1 < p_2 < \ldots < p_r$. All repeated factors can be collected together and expressed using exponents, such that $N = p_1^{m1} p_2^{m2} \ldots p_r^{mr}$, where each m_i is a positive integer, called the *multiplicity* of p_i, and this factorization of N is called the *standard form* of N [7]. For example, $N = 1800 = 2^3 * 3^2 * 5^2$. For our purposes, we are particularly interested in multiplicity $m_i = 1$ because there is no duplicated item in transactions.

To facilitate the process of the database encoding technique used in our method, let's examine it through an example. Let the transaction database, *DB*, be the first two columns of Table 1 with six transactions and item set I= {A, C, D, T, W}.

Table 1. A transaction database and its V_{TID}

TID	Items	Encoded	V_{TID}
1	A,C,T,W	2,3,7,11	**462**
2	C,D,W	3,5,11	**165**
3	A,C,T,W	2,3,7,11	**462**
4	A,C,D,W	2,3,5,11	**330**
5	A,C,D,T,W	2,3,5,7,11	**2310**
6	C,D,T	3,5,7	**105**

Each item i_j is presented by one prime number P_j as shown in third column of Table 1. We used very simple equation (1) to compute value V_{TID} for every transaction. In fact equation (1) makes a new composite integer by using the prime numbers. The V_{TID} is product all considered prime numbers according to participated items in the transaction. For example for fourth transaction where itemsets = {A, C, D, W} using equation (1), $V_4=2*3*5*11=330$.

$$V_{TID}=\prod P_i \qquad (1)$$

The fourth column of Table 1 shows computed V_{TID} for all transactions. By this way, the encoded file is much smaller can be loaded into memory more easily than the original transaction database. Our experiments

on some benchmark datasets showed that by applying this database encoding technique, the size of database can be reduced more than half as shown in Fig 2.

B. *The PC_Tree & PC_Miner Algorithms*

It this research, we introduce a novel and simple tree structure called Prime-based encoded and Compressed Tree or PC_Tree. Mostly, researchers have proposed tree structure in static and incremental frequent itemset mining [10, 11] that motivate us to introduce a simple tree structure but yet efficient.

A PC_Tree is a tree structure consists of a root and some sub trees as the children of the root. Node's structure in the tree has three fields: Value, Count, and Link. The Value field records which transaction this node represents, the Count field registers the number of records reaching this value. Database encoding and tree construction can be done with together by only one database scan.

To illustrate the construction of the PC_Tree, figure 1 shows the PC_Tree for table 1. Firstly, the root node is created and it is considered that all Values can be as a divisor for root node. Then PC_Tree construction algorithm reads a Value from encoded file called Next and calls the function Tree_construction. The function Tree_construction is performed such that every child node can divide its parent node for example node 165 is a divisor for the parent node 330 and if the Next is equal value of the current node then current node's count is increased by 1 like 462:2.

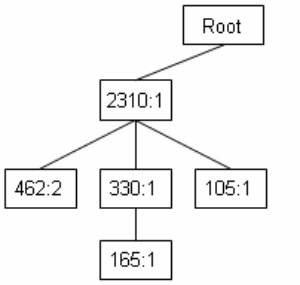

Fig 1. The PC_Tree for Table 1.

The PC_Tree has some nice properties as described below:

Property 1: Nodes are arranged according to V_{TID} order, which is a fixed global ordering.

Property 2: The frequency of itemsets X registered in node N is equal the sum all Count field of N-tree (sub tree from root to N). For example frequency of itemset X= {ACTW} or Value=462 is equal 1+2=3. As a good result, there is no need to database scan for counting of itemsets and its subset that registered in the PC_Tree.

Property 3: According to property 2, the frequency of itemset X registered in node N is less than frequency of itemset Y registered in node M where M is a child of N in the PC_Tree.

Property 4: Given σ as minimum support and Y is a child for X in the PC_Tree, If Count (Y) ≥ σ and Count (X) < σ then Y is a Maximal Frequent itemset.

As showed in section 3.1, the output of the database encoding phase is the encoded file includes all V_{TID} for 0< TID < |D| where |D| is number of transactions. Therefore in the worst case, maximum number of nodes in the PC_Tree is |D|.

Let us introduce our PC_Miner algorithm and show how it mines MFI. The PC_Miner utilizes gcd theory. The *greatest common divisor* of two integers *a* and *b*, not both zero, is the largest of the common divisors of *a* and *b*; it is denoted gcd (*a*, *b*). For example, gcd (12, 16) = 4, gcd (5, 11) =1, and gcd (0, 12) = 12. If *a* and *b* are not both 0, then gcd(*a*, *b*) is an integer between 1 and min(|*a*|, |*b*|) [7].

The PC_Miner algorithm traverses the PC_Tree. It prunes search space using the properties of the PC_Tree and builds the gcd sets to find the MFI.

For example, Given the transaction database shown in Table 1 which its PC_Tree shown in Fig 1 and let σ=*3*. According to the PC_Tree properties and gcd set used in PC_Miner algorithm, node 462:1+2 (or itemset ACTW) and node 165:1+1+1 (or itemset CDW) will be discovered as MFI.

IV. Experimental Results

In this section, we evaluate the performance of our method. All the experiments are performed on PC with CPU Intel P4 2.8 GHz, 1 Gigabytes main memory, and running Microsoft Windows XP. All the algorithms are implemented using Microsoft Visual C++ 6.0.

In first experiment, the Synthetic data used in our experiments are generated using IBM data generator which has been used in most studies on frequent itemsets mining. We generate five datasets with number of items 1000, average transaction length 10 and number of transaction 1000 to 10000 that called D1, D2, D4, D8, and D10 respectively. In these experiments, our database encoding technique shows good results that it can reduce the size of these datasets more than half as shown in Figure 2.

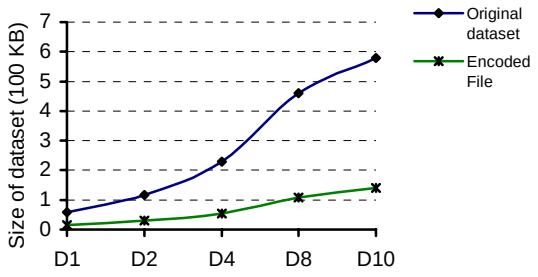

Fig 2. The results of Database encoding technique

Most important result is, in our technique a prime number is considered for each item independent of the size of items. Obviously the size of items in real transaction databases is bigger than the size of items in

benchmark datasets. It means our database encoding technique can be more efficient for reducing the size of real transaction database.

In second experiment, we show accuracy and correctness of the PC_Miner. The test dataset T10.I6.D10K is also generated synthetically by the IBM data generator. Figure 3 shows the numbers of maximal frequent itemsets discovered for the tests at varying minimal supports on this dataset.

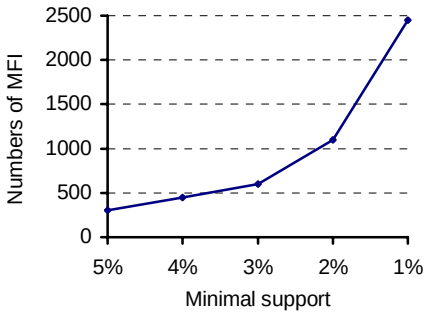

Fig 3. Numbers of MFI for T10I6D10k

In third experiment, we compare the performance of the PC_Miner algorithm with the Apriori algorithm. In order to evaluate the effectiveness of our new algorithm, we applied it as well as Apriori to four IBM dataset generated in experiment 1. Figure 4 shows the performance of two algorithms as a function of the numbers of transactions.

As shown in Figure 4, when number of transaction is less than 5000 Apriori slightly outperforms the PC_Miner in execution time. When the numbers of transactions are increased, the execution time of Apriori degraded as compared to the PC_Miner.

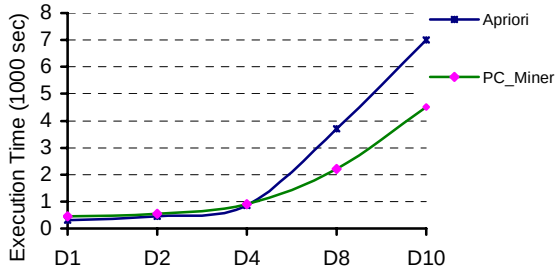

Fig 3. Performance of PC_Miner vs. Apriori

V. Conclusion and Future Works

In this paper, we proposed a new method to discover maximal frequent itemsets efficiently. Our method introduces an efficient database encoding technique, a tree structure called Prime-based encoded and Compressed Tree or PC_Tree and also PC_Miner algorithm. The experiments showed the database encoding technique can reduce the size of transaction database more than half and it can enhance the performance of mining algorithms. And also we showed that our PC_Miner algorithm can discover all MFI using this encoding technique.

Here have been considered some direction as future works. The first, using the optimal data structures, better memory management and pruning method to enhance the efficiency of our method. Then, it can be extended to generate the frequent itemsets. Finally, It will be improved by a new matrix structure which keeps all information about current frequent itemsets for incremental mining of frequent itemsets in dynamic databases where transaction database is updated or minimum support threshold can be changed [11].

References

[1] R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad, "Depth first generation of long patterns," *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 108-118, 2000.

[2] R. Agrawal and R. Srikant, "Fast algorithms for mining association rules," *Proc. 20th Int. Conf. Very Large Data Bases, VLDB*, vol. 1215, pp. 487499, 1994.

[3] S. Bashir and A. R. Baig, "HybridMiner: Mining Maximal Frequent Itemsets Using Hybrid Database Representation Approach," *9th International Multitopic Conference, IEEE INMIC 2005*, pp. 1-7, 2005.

[4] R. J. Bayardo Jr, "Efficiently mining long patterns from databases," *Proceedings of the 1998 ACM SIGMOD international conference on Management of data*, pp. 85-93, 1998.

[5] D. Burdick, M. Calimlim, and J. Gehrke, "Mafia: A maximal frequent itemset algorithm for transactional databases," *Proceedings of the 17th International Conference on Data Engineering*, pp. 443-452, 2001.

[6] F. Chen and M. Li, "A Two-Way Hybrid Algorithm for Maximal Frequent Itemsets Mining," *Fuzzy Systems and Knowledge Discovery, 2007. FSKD 2007. Fourth International Conference on*, vol. 3, 2007.

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

[8] J. Han, 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] D. I. Lin 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, March 23-27, 1998: Proceedings*, 1998.

[10] N. Mustapha, 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.

[11] M. Nadimi-Shahraki, 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.

[12] M. J. Zaki, "Scalable algorithms for association mining," *Knowledge and Data Engineering, IEEE Transactions on*, vol. 12, pp. 372-390, 2000.