Chapter 42
PC_Tree: Prime-Based and Compressed Tree for Maximal Frequent Patterns Mining

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

Abstract Knowledge discovery or extracting knowledge from large amount of data is a desirable task in competitive businesses. Data mining is an essential step in knowledge discovery process. Frequent patterns play an important role in data mining tasks such as clustering, classification, and prediction and association analysis. However, the mining of all frequent patterns will lead to a massive number of patterns. A reasonable solution is identifying maximal frequent patterns which form the smallest representative set of patterns to generate all frequent patterns. This research proposes a new method for mining maximal frequent patterns. The method includes an efficient database encoding technique, a novel tree structure called PC_Tree and PC_Miner algorithm. Experiment results verify the compactness and performance.

Keywords Maximal frequent pattern · frequent pattern · prime number · database encoding

42.1 Problem of Maximal Frequent Patterns Mining

Since the introduction of the Apriori algorithms [1], frequent patterns mining plays an important role in data mining research for over a decade. Frequent patterns are itemsets or substructures that exist in a dataset with frequency no less than a user specified threshold.

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

M. Nadimi-Shahraki (✉)
Department of Computer Engineering, Islamic Azad University, Najafabad branch, Iran and PhD student of Computer Science, Faculty of Computer Science and Information Technology, Putra University of Malaysia, 43400 UPM, Selangor, Malaysia
E-mail: admin1@iaun.ac.ir

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 $\sigma(0 \leq \sigma \leq |D|)$.

In many real applications especially in dense data with long frequent patterns enumerating all possible 2^L – two subsets of an L length pattern is infeasible [2]. A reasonable solution is identifying a smaller representative set of patterns from which all other frequent patterns can be derived [3]. Maximal frequent patterns (MFP) form the smallest representative set of patterns to generate all frequent patterns [4]. 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.

This research introduces a new method to find all MFP using only one database scan. The method includes an efficient database encoding technique, a novel tree structure called Prime-based encoded and Compressed Tree or PC_Tree and also PC_Miner algorithm. The database encoding technique utilizes prime number theory and transforms all items from each transaction into only a positive integer. The PC_Tree is a novel and simple tree structure but yet efficient to capture whole of transactions information. The PC_Miner algorithm traverses the PC_Tree efficiently using pruning techniques to find the MFP.

42.2 Related Work

Many efficient algorithms have been introduced to solve the problem of maximal frequent pattern mining more efficiently. They are almost based on three fundamental frequent patterns mining methodologies: Apriori, FP-growth and Eclat [5]. Mostly, they traverse the search space to find MFP. The key to an efficient traversing is the pruning techniques which can remove some branches in the search space. The pruning techniques 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 [6] 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 [4] 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 Depth Project [7] 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 [2] extends the idea in DepthProject. It uses a search strategy has been improved by an effective pruning mechanisms. The MaxMiner, DepthProject and Mafia use Rymon's set enumeration [8] to enumerate all the patterns. Thus these algorithms avoid having to compute the support of all the frequent patterns. The Flex [9] 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. Recently a new two-way-hybrid algorithm [10] for mining MFP uses a flexible two-way-hybrid search method. 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.

42.3 Proposed Method

This research proposes a new method to mine all MFP in only one database scan efficiently. The method introduces an efficient database encoding technique, a novel tree structure called PC_Tree to capture transactions information and PC_Miner algorithm to mine MFP.

42.3.1 Database Encoding Technique

The presentation and encoding of database is an essential consideration in almost all algorithms. The most commonly database layout is the horizontal and vertical layout [11]. In both layouts, the size of database is very large. The database encoding is a useful technique 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 database encoding 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 $= \{\mathrm{i}_j \ldots \mathrm{i}_k\}$ is the transaction-items or pattern X. While the PC_Tree algorithm reads transaction T, the encoding procedure considers a prime number p_r for each item i_r in pattern X, and then $\mathrm{TV}_{\mathrm{tid}}$ is computed by Eq. (42.1). Therefore, all transactions can be represented in new layout using this encoding technique.

$$TV_{tid} = \prod_{j}^{k} p_r \ \mathrm{T} = (\mathrm{tid}, \mathrm{X}) \tag{42.1}$$

$$\mathrm{X}* = \{\mathrm{i}_j \ldots \mathrm{i}_k\} \text{ and } \mathrm{i}_r \text{ is presented by } p_r)$$

Table 42.1 The transaction database DB and its Transaction Values

TID	Items	Encoded	TV
1	A, B, C, D, E	2, 3, 5, 7, 11	2,310
2	A, B, C, D, F	2, 3, 5, 7, 13	2,730
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

The encoding technique utilizes Eq. (42.1) based on simple following definitions.

A positive integer N can be expressed by unique product
$\mathrm{N} = p_1^{m_1} p_2^{m_2} \ldots p_r^{m_r}$ 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 [12].

For example, $\mathrm{N} = 1,800 = 2^3 * 3^2 * 5^2$. Here we restrict the multiplicity to $\mathrm{m}_i = 1$ because there is no duplicated item in transaction T.

To facilitate the process of the database encoding technique 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 42.1 with eight transactions. The fourth column of Table 42.1 shows $\mathrm{TV}_{\mathrm{tid}}$ computed for all transactions.

42.3.2 PC_Tree Construction

Using tree structure in mining algorithms makes two possibilities to enhance the performance of mining. Firstly, data compressing by well-organized tree structures 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 [5, 9, 13]. This research introduces a novel tree structure called PC_Tree (Prime-based encoded and Compressed Tree). The PC_Tree 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 [13]. The status field is to keep track-

TID	TV
1	2310
2	2730
3	66
4	770
5	455
6	910
7	70
8	455

(a) TVs of the DB

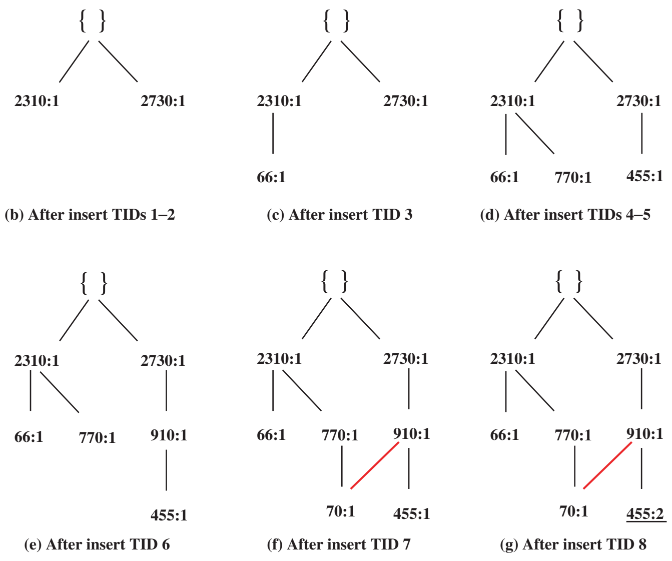

Fig. 42.1 Construction of PC_Tree

ing 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.

Figure 42.1 shows step by step construction of PC_Tree for transactions shown in Table 42.1 which summarized in Fig. 42.1a.

The construction operation mainly consists of insertion procedure that inserts TV(s) into PC_Tree based on definitions below:

Definition 42.1. 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 42.2. $R = (n_i, n_{i-1} \ldots n_j, root)$ IS A DESCENDANT IFF TV OF NODE $n_r \in R(i \leq r \leq j)$ CAN DIVIDE ALL TVS KEPT IN NODES DESCENDANT $R_r = (n_{r+1}, n_{r+2}, \ldots n_j, root)$.

Definition 42.3. TV OF THE ROOT IS ASSUMED NULL AND CAN BE DIVIDED BY ALL TVS.

Definition 42.4. NODEn_rCAN BE BELONGED TO MORE THAN ONE DESCENDANT.

Figure 42.1b shows insertion of the first and second transactions. The second TV with value 2,730 can not be divided by the first TV with value 2,310 and it creates a new descendant using definition 2 and 3. Transactions 3–6 are inserted into their descendant based on definition 2 shown in Fig. 42.1c–e. Insertion of the seventh transaction applies definition 4 when TV 70 is belonged to two descendants (second descendant shown in the red and bold line) shown in Fig. 42.1f. The TV of eighth transaction with value 455 is equal with the fourth TV, then the local-count field of forth TV is increased by 1 using definition 1 shown in Fig. 42.1g (shown in underline).

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 42.1. The S (P) is computed by traversing only descendants of TVP.

Property 42.2. P and Q belong to descendant R and S (P) $<$ S (Q) iff TVP can be divided by TVQ.

Property 42.3. S (P) $\geq \sigma$ and S (P's fathers in its descendants) $\prec \sigma$ IFF P is a Maximal Frequent Pattern.

Property 42.4. Nodes are arranged according to TV order, which is a fixed global ordering. In fact the PC_Tree is an independent-frequency tree structure.

Property 42.5. 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.

42.3.2.1 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 Fig. 42.1g 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 42.2 shows the item frequency and considered prime number for transaction database shown in Table 42.1.

To illustrate the pruning strategy used in the PC_Miner algorithm let's examine it by some examples. Figure 42.2 shows the completed PC_Tree using TVs and patterns respectively. Given $\sigma = 5$, according to Table 42.2, infrequent item set (IFI) consists of {B, E, F} then all superset of IFI like node (A, B, C, D, E) can be

Table 42.2 Frequency of items and considered 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

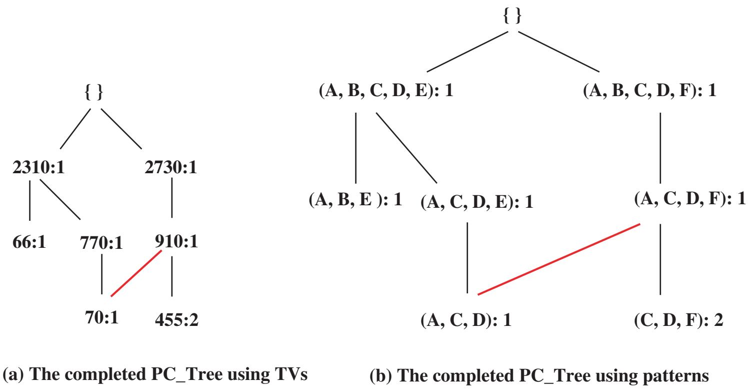

Fig. 42.2 The completed PC_Tree

pruned – the superset infrequency pruning. Given σ has been changed from 5 to 2 then all items are frequent. While the PC_Miner traverses left sub tree or descendant of node (A, B, C, D, E), node (A, C, D, E) is found as a maximal frequent – the property 3. Then all its subsets (here is only node (A, C, D)) are pruned – the subset frequency pruning.

42.4 Experimental Results

In this section, the accuracy and performance of the method is evaluated. All the experiments are performed on PC with CPU Intel P4 2.8 GHz, 2 GB 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 patterns mining. We generate five datasets with number of items 1,000, average transaction length ten and number of transaction 1,000–10,000 that called D1, D2, D4, D8, and D10 respectively. In this experiment, the compactness of our database encoding technique is verified separately. In fact all TVs transformed are stored in a flat file

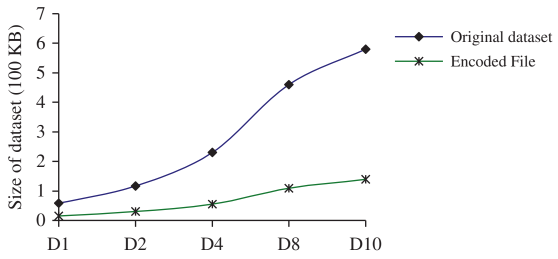

Fig. 42.3 The compactness of the database encoding technique

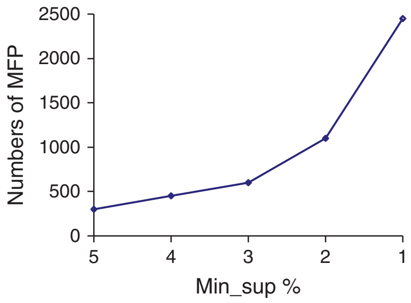

Fig. 42.4 The number of MFP discovered for T10.I6.D10k

called encoded file then size of this file is compared with the size of original dataset. It shows good results and reducing the size of these datasets more than half as shown in Fig. 42.3.

In second experiment, we show accuracy and correctness of the method. The test dataset T10.I6.D10K is also generated synthetically by the IBM data generator. Figure 42.4 shows the numbers of maximal frequent patterns discovered for the tests at varying min_sup on this dataset.

In third experiment, in order to evaluate the scalability of our new algorithm, we applied it as well as Apriori to four IBM dataset generated in experiment 1. Figure 42.5 shows the performance of two algorithms as a function of the numbers of transactions for min_sup 2%. When number of transaction is less than 5,000 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.

Fourth experiment is to compare the performance of the PC-Miner with the Apriori and Flex algorithms on the dataset mushroom, which is a real and dense dataset contains characteristics of various species of mushrooms [14]. In dataset mushroom, there are 8,124 transactions, the number of items is 119 and the average transaction length is 23. Figure 42.6 shows the PC_Miner algorithm outperforms both Apriori and Flex algorithms.

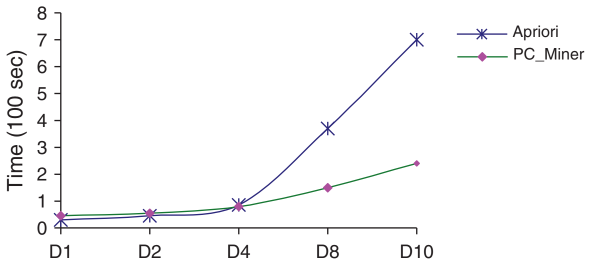

Fig. 42.5 The scalability of PC_Miner vs. Apriori on datasets D1-D10

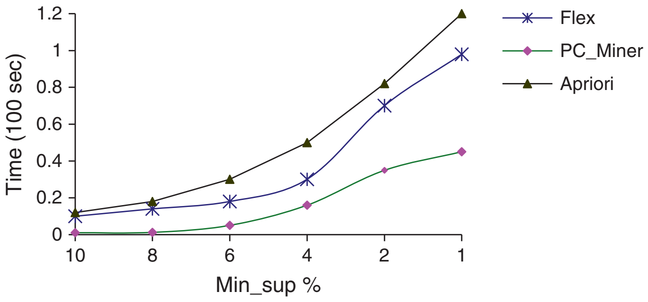

Fig. 42.6 The performance of PC-miner vs. Flex on dataset mushroom

42.5 Conclusion and Future Works

In this paper, we proposed a new method to discover maximal frequent patterns efficiently. Our method introduced an efficient database encoding 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 database encoding 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 Apriori and Flex algorithms. For interactive mining where min_sup can be changed frequently, the information kept in the PC_Tree can be used and tree restructuring is no needed.

In fact this research introduced a number-theoretic method to discover MFP that made use of prime number theory and simple computation based on division operation and a combined pruning strategy. There are some directions for future improvement optimal data structures, better memory management and pruning method to enhance the efficiency. This method also can be extended for incremental frequent patterns mining where transaction database is updated or minimum support threshold can be changed [13].

References

1. R. Agrawal and R. Srikant, Fast algorithms for mining association rules, 20th International Conference. Very Large Data Bases, VLDB, 1215: 487–499 (1994).
2. D. Burdick, M. Calimlim, and J. Gehrke, Mafia: A maximal frequent itemset algorithm for transactional databases, 17th International Conference on Data Engineering: pp. 443–452 (2001).
3. S. Bashir and A.R. Baig, HybridMiner: Mining Maximal Frequent Itemsets Using Hybrid Database Representation Approach, 9th International Multitopic Conference, IEEE INMIC: pp. 1–7 (2005).
4. R.J. Bayardo Jr, Efficiently mining long patterns from databases, ACM SIGMOD International Conference on Management of Data: pp. 85–93 (1998).
5. J. Han, H. Cheng, D. Xin, and X. Yan, Frequent pattern mining: Current status and future directions, Data Mining and Knowledge Discovery, 15(1): 55–86 (2007).
6. 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 (1998).
7. R.C. Agarwal, 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).
8. R. Rymon, Search through systematic set enumeration, Third International Conference on Principles of Knowledge Representation and Reasoning: pp. 539–550 (1992).
9. N. Mustapha, M.N. Sulaiman, M. Othman, and M.H. Selamat, FAST DISCOVERY OF LONG PATTERNS FOR ASSOCIATION RULES, International Journal of Computer Mathematics, 80(8): 967–976 (2003).
10. F. Chen, and M. Li, A Two-Way Hybrid Algorithm for Maximal Frequent Itemsets Mining, Fourth International Conference on Fuzzy Systems and Knowledge Discovery, FSKD 2007.
11. M.J. Zaki, Scalable algorithms for association mining, IEEE Transactions on Knowledge and Data Engineering, 12(3): 372–390 (2000).
12. T.T. Cormen, C.E. Leiserson, and R.L. Rivest, Introduction to algorithms: MIT Press, Cambridge, MA (1990).
13. M. Nadimi-Shahraki, N. Mustapha, M.N. Sulaiman, and A. Mamat, Incremental updating of frequent pattern: basic algorithms, Second International Conference on Information Systems Technology and Management (ICISTM 08) (1): 145–148 (2008).
14. C.B.a.C. Merz, UCI Repository of Machine Learning, 1998.