PTclose: A novel algorithm for generation of closed frequent itemsets from dense and sparse datasets

J. Tahmores Nezhad[✉], M.H.Sadreddini[♣]

***Abstract*—In recent years, various algorithms for *mining closed frequent itemsets (CFI)* have been proposed. Different structures such as prefix sharing are used within these algorithms. However the name challenging problem in many of them is the high requirement of memory, especially in case of sparse datasets. Thus, most of the proposed methods are proper for dense datasets. In this paper we present a new approach to mining closed frequent itemsets using two structures, namely Patricia tree and PTArray. By using these two structures, both the response time and also memory consumption are reduced, significantly. The proposed method, called PTclose is suitable for both dense and sparse datasets. The algorithm is assessed throughout a set of experiments. The results narrate for the relative efficiency of the algorithm compared to other existing methods.**

***Index Terms*—Association rules, closed frequent itemset, condensed representation, Patricia tree**.

I. Introduction

Mining *Association Rules* (ARs) is a popular technique for discovery of interesting relations between items in large databases and transaction warehouses [1], [2], [3], [4], [7], [9], [10], [15], [16], [19]. The fundamental part of mining ARs is the process of mining frequent itemsets (FIs). The count of an itemset X is the number of transactions in D that contain X. The support of an itemset X is the proportion of transactions in D which contain X. In other words, the support of X is the count of X divided by the number of transactions. A FI can be roughly defined as a set of items which occur together, frequently (its support is greater than or equal to a specific threshold), Mining all frequent itemsets might not be a good idea. For example, if there is a frequent itemset with size k, then all 2^k nonempty subsets of the itemset have to be generated. However, since any subset of a frequent itemset is frequent, it is sufficient to discover only all the *maximal frequent itemsets* (MFIs). A frequent itemset X is called *maximal* if there is no other frequent itemset Y such that $X \subseteq Y$. Some existing algorithms only mine maximal frequent itemsets. However, mining only MFIs has the following problem: Given an MFI and its support , s, we know that all its subsets are frequent and the support of any of its subset is not less than s, but the exact values of their supports are not known. To solve this problem, another type of a frequent itemset, called *closed frequent itemset* (CFI), was proposed in [12]. A frequent itemset X is *closed* if none of its proper supersets have the same support. Thus, the set of all CFIs contains complete information for generating association rules.

Most of the frequent pattern mining algorithms, including Apriori [2], FP-growth [6], H-mine [5], and OP [8], mine all frequent itemsets. These algorithms have good performance in case that the pattern space is sparse and the value of support threshold is set high. However, when the value of support threshold drops low, the number of frequent itemsets goes up dramatically, and the performance of these algorithms deteriorates quickly because of the generation of a huge number of patterns.

The closed itemset mining, which was proposed in [12] for the first time, mines only those frequent itemsets having no proper superset with the same support. In recent years, many efficient algorithms have been proposed for mining frequent closed itemsets, such as A-close [12], CLOSET [13], and CHARM [20]. Various depth-first and level-wise search strategies have been used in these methods. It has been shown through a set of experiments that CLOSET+ [17] and FPclose [23] have a better performance in comparison with other existing mining algorithms, including CLOSET, CHARM and OP.

A typical challenge is that many of the proposed methods are suitable just for dense datasets, while many others are proper only for sparse ones. In this paper, we propose a new method for mining closed frequent itemsets, called PTClose. The algorithm constructs a compressed tree in memory, using the Patricia tree structure. In order to increase the efficiency, the algorithm uses the PTArray structure, which will be illustrated later. It will be shown, through a set of experiments, that the proposed scheme is proper for both dense and sparse data sets, in terms of runtime and memory usage.

The rest of the paper is organized as follows. In Section 2, a brief description of some major topics, including the Patricia tree structure, the PTArray and the CFI-tree is given. Section 3 is devoted to illustration of our proposed method for mining closed frequent itemsets. In Section 4, experimental results on synthetic and real-life data sets are shown. Finally, Section 5 concludes the paper.

[✉] Department of Computer Engineering, School of Engineering, Islamic Azad University of Shiraz, Shiraz, Iran, Email: tahmores@gmail.com

[♣] Department of Computer Science & Engineering, School of Engineering, Shiraz University, Shiraz, Iran, Email: sadredin@shirazu.ac.ir

II. Definitions

Before illustrating the new CFI mining method, in this section, we give some brief definitions for a number of related topics, including the Patricia tree, the PTArray and the CFI-tree structures. These are the major structures that are used in the proposed scheme.

A. Patricia tree structure

The Patricia tree is a compact tree, used to represent all relevant frequency information within a dataset. Each branch of the tree represents a frequent itemset. The nodes along the branches are stored in decreasing order of frequency from the root through the leaves. Overlapping itemsets in a Patricia tree share prefixes of the corresponding branches. That's why it is known as a compressed structure. The Patricia tree coalesces a set of sequential nodes (within a chain), having the same support value, c, into a single node with support of c. Each Patricia tree is associated with a header table, namely T.header, which is used to hold items and their frequencies in decreasing order of the counts. Each entry in a header table points to a linked list containing identical nodes in the Patricia tree.

The construction process of the Patricia tree is as follows. First, the whole dataset is scanned and the all frequent items are identified, sorted by their frequencies and stored in header table entries. Infrequent items are then removed from the dataset. In the second scan, each transaction is scanned and the set of frequent items in it is inserted into the Patricia tree as a branch. Each node of a Patricia tree contains a counter to store the number of transactions containing the itemset represented by the path from the root to that node. If an itemset shares a prefix with an itemset already in the tree, the counter is incremented by 1, rather than inserting a new node.

The traversal procedure of the Patricia tree is based on the following principle: If X and Y are two itemsets, the count of itemset X∪Y in the dataset exactly equals the count of Y in a fraction of the dataset that contains X. A restriction of the dataset containing X, is called X's conditional pattern base. The Patricia tree constructed for the typical itemset, X, with respect to the conditional pattern base is called X's conditional Patricia tree and denoted by T_X.

Given an item i in T_X.header, the linked list is followed beginning from i in T_X.header (to visit all branches that contain item i). The portion of these branches from i to the root forms the conditional pattern base of X∪{i}. Thus, the traversal obtains all frequent items in this conditional pattern base. Then it constructs the conditional Patricia tree $T_{X\cup\{i\}}$ by first initializing its header table using the frequent items found, then revisiting the branches of T_X along the linked list of i and inserting the corresponding itemsets in $T_{X\cup\{i\}}$.

The above procedure is executed recursively until only one branch remains in the resulting Patricia tree. Generation of the complete set of closed frequent itemsets can be achieved straightforward, from single path Patricia trees.

Table I shows a transactional data set. Items of each transaction are sorted in decreasing order of their frequencies.

Table I. A transactional data set, items sorted in decreasing order of the frequency

TID	Items
1	B A D F G H
2	B L
3	B A D F L H
4	B A D F L G
5	B L G H
6	B A D F

In Fig.1, a Patricia tree constructed for this data set is shown.

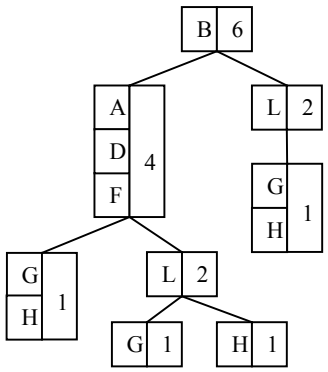


Fig.1 A Patricia tree for the data set of Table I

*B. **PTArray***

As claimed in [23], through numerous experiments, it has been shown that about 80 percent of the CPU time is used for traversing trees. By using a simple additional data structure, named PTArray, it will be possible to increase the traversal speed. As mentioned before, for each item i in the header of a conditional Patricia tree T_X, two traversals of T_X are performed in order to construct the new conditional Patricia tree $T_{X\cup\{i\}}$. The first traversal is run to find all frequent items in the conditional pattern base of X∪{i} and initializes the Patricia tree $T_{X\cup\{i\}}$ by constructing its header table. Through the second traversal, the new tree $T_{X\cup\{i\}}$ is constructed. The first scan of T_X can be easily ignored by constructing the *Patricia Tree Array* called *PTArray* of T, while building T_X.

PTArray is a matrix, each element of which corresponds to the counter of an ordered pair of items in the header table of T_X. Since the PTArray is a symmetric matrix, there is no need to set a counter for both item pairs (i_j,i_k) and (i_k,i_j) and thus it is enough to store the counters for all pairs (i_k,i_j) such that $k < j$.

C. CFI-tree

The best way we can verify that a generated itemset is closed or not, is to use a compressed tree containing previously found closed frequent itemsets. If a typical itemset is found to be closed, then it can be inserted into the tree. If the current itemset Sc can be subsumed by another already found closed itemset Sa, they must have the following relationships: (1) Sc and Sa have the same support; (2) length of Sc is smaller than that of

Sa; and (3) all the items in Sc are contained in Sa. Using these heuristics, the structure of tree can be improved. Each closed itemset is inserted into the CFI-tree according to the header table order, and at each node the path length from the root down to the node is also recorded. The fields stored in each node are itemID, support, and the length (distance from the root node). When several closed itemsets share some common prefixes, the support of a node in the common prefix will be the maximum one among the supports of itemsets sharing the common prefix. Fig.2 shows a CFI-tree constructed for the CFIs found from the Patricia tree shown in Fig.1.

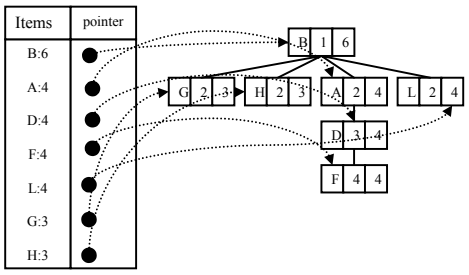


Fig.2 A CFI-tree for the CFIs found from the Patricia tree of Fig 1

III. The Proposed Algorithm: PTclose

In this section, we present our new algorithm, PTclose which can be used for mining of all closed frequent itemsets, having a better speed and less requirement of main memory, in comparison with the proposed methods in the literature. In this algorithm we use a technique called PTArray, which reduces the time to scan the Patricia tree, significantly. we use local CFI-trees in order to verify the generated closed frequent itemsets, whether they are closed or not. This reduces the total time of generating closed frequent itemsets, effectively.

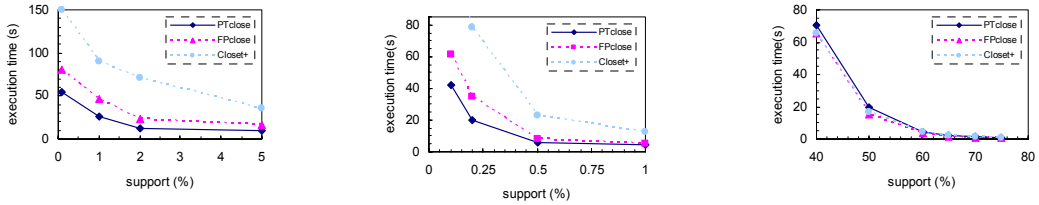
The proposed algorithm is presented in Fig.3. As shown in this figure the algorithm has two inputs: a Patricia tree, denoted by T and a CFI-tree, denoted by C. The output of the algorithm is the completed CFI-tree. In the first call of the algorithm (from the main module of the program) the first parameter (T) is the Patricia tree constructed for the main dataset and the second parameter (C) is an empty tree. As the first step of the algorithm it is verified whether T is a single path tree. if so, all candidate CFIs are generated from T. For this purpose, the following steps are followed: let $(i_1{:}c_1,i_2{:}c_2,\ldots,i_k{:}c_k)$ be a representation for the tree path, where $i_j{:}c_j$ means that the typical item i_j has the counter c_j. Beginning with i_1, the counters are compared for each pair of adjacent items (such as $i_j{:}c_j$ and $i_{j+1}{:}c_{j+1}$) if c_j does not equal c_{j+1}, we select the itemset $\{i_1,i_2,\ldots,i_j\}$ as a candidate CFI. The candidate itemset is then compared with all the CFIs within C. If it is a closed itemset, it will be inserted into C and all CFI-tree existing in memory will be updated. In other words the candidate CFI is inserted into all CFI-trees. This is because the algorithm is called recursively and the generated CFI of each iteration must be accessible for the previous call environments. On the other hand, if the Patricia tree is not single path, the algorithm execution continues from line 6. In this part for each item within the header table, the lines 7 through 19 are executed. The condition mentioned in line 7 implies that there is no closed frequent itemset, X, such that T.base and X have the same support value and T.base$\subseteq$X. In line 8 if the PTArray is defined, it is used to compute the conditional pattern base of T.base$\cup${i} and also the support value of items. However, if the PTArray is not defined, support of items for T.base$\cup${i} is measured, by traversing of the conditional Patricia tree (shown in line 11).

Algorithm PTclose (T,C)
Input : T, a Patricia tree
C, a CFI-tree for T
Output : complete CFI-tree

```
1.  if T is single path
2.     for each CFI cfi in T
3.        if IsClosed(cfi,C)
4.           insert cfi into C
5.           update all CFI-trees in
               memory
6.  else for each i in T.header
7.     if IsClosed(T.base∪{i},C)
8.        if T.PTArray is defined
9.           compute conditional pattern
             base for i from T.PTArray
10.       else
11.          compute conditional pattern
             base for i from Patricia tree T
12.       initial Patricia tree T_{T.base∪{i}}
13.       if PTArrayIsNeeded (T_{T.base∪{i}})
14.          initial PTArray A_{T.base∪{i}}
15.       construct T_{T.base∪{i}}
16.       if A_{T.base∪{i}} is defined
17.          fill A_{T.base∪{i}}
18.       initial CFI-tree C_{T.base∪{i}}
19.       PTclose(T_{T.base∪{i}},C_{T.base∪{i}})
20. end
```

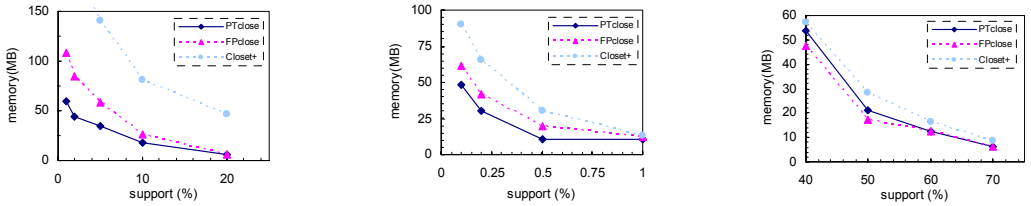
Fig.3 PTclose: the proposed algorithm for mining closed frequent itemsets

The function IsClosed(Y,C) is developed to verify whether the itemset Y is closed, using the C tree. The function has the following procedure: let $Y=\{i_1,i_2,\ldots,i_p\}$ be an itemset having the counter c. Assume the ordering of the items within the header table of the current CFI-tree be as follows $i_1,i_2,\ldots,i_p$. The counter of each node within the linked list of i_p is compared with C. Then using the level field we compare Y with its ancestors, in order to verify it for being closed. Here, the level parameter in each node is used to reduce the comparison time. This function returns a true value, if there is no closed frequent itemset, Z in the CFI-tree, such that Z is a superset of Y and the counter of Y is greater than or equal to the counter of Z.



(a) T40.I10.D100k.N1k.L2k (b) Kosarak (c) Chess

Fig.4 The execution time of mining closed frequent itemsets



(a) T40.I10.D100k.N1k.L2k (b) Kosarak (c) Chess

Fig.5 The memory consumption of mining frequent itemsets

In lines 12 through 18 conditional Patricia tree and the conditional CFI-tree for T.base∪{i} are constructed. The function PTArrayIsNeeded is used to determine whether the PTArray is required or not. The PTArray technique has a good performance, especially for sparse and huge datasets. PTArray structure reduces the time to scan items. Moreover, the Patricia trees of next levels can be initialized directly, using this structure. When the dataset is dense, there is no need to construct PTArrays, since the generated tree is compressed and its traverse can be accomplished quickly.

Note that datasets or conditional patterns are modified during the sequential calls of the recursive function. In order to estimate that a dataset is sparse or dense, the nodes in each level of the tree are enumerated, during the construction of each Patricia tree. If the first quarter part of the tree contains less than 15% of all nodes, then the dataset is known to be dense, otherwise, it will be identified as sparse [23]. If the dataset is dense, we do not construct the PTArray for the next level of the Patricia tree, else it will be constructed for each Patricia tree of the next iteration. After the algorithm execution is completed, C_0 includes all of the CFIs, mined from the main dataset.

IV. Experimental Results

In order to evaluate the proposed algorithm and compare it with other existing methods, we performed an experiment over some real-life and synthetic datasets [24]. Chess and Kosarak are two real datasets while T40.I10.D100.N1k.L2k is synthetic. Chess is a dense dataset and the other two have a sparse nature. The general statistics of the datasets are presented in Table II. In this experiment our proposed algorithm, PTclose, is compared with FPclose and Closet+, as two efficient algorithms in this field. The algorithms were implemented in C++ and executed in windows XP (Service Pack 2) over a system with a 2.66GHz CPU, 256MB of RAM and having 512KB cache memory.

Table II. General statistics of datasets used in experiments

Dataset	Transactions	AvgTS
Chess	3,196	35.53
T40.I10.D100k.N1k.L2k	100,000	39.54
Kosarak	229,148	8.07

A. *Execution time*

As the first evaluation factor, we compared the execution times of the algorithms for mining all closed frequent itemsets over the three mentioned datasets. The experiments were performed iteratively, for different values of the minimum support threshold. The results of this comparison are shown in Fig.4. It can be observed in this figure that the proposed algorithm has a much better performance than the others for sparse datasets. However the results show that it performs similar to FPclose and Closet+, when the dataset is dense.

B. *Memory consumption*

The other factor considered for evaluation of the algorithms, was their requirements for main memory. Fig.5 indicates the amount of memory consumption of each algorithm for different values of minimum support threshold (over each dataset). This figure implies that PTclose is proper for sparse datasets (with respect to its less memory consumption) rather than other algorithms.

V. Conclusion

In this paper we proposed a new algorithm called PTclose for mining of closed frequent itemsets with relatively low response time and memory consumption. This algorithm uses the Patricia tree structure and thus requires a shorter time to scan the items, compared to other existing algorithms. In this algorithm a new technique, called

PTArray is used. This technique reduces the scan time of the Patricia tree significantly. We use local CFI-trees in order to verify whether the generated closed frequent itemsets, are closed or not. This reduces the total time of generating closed frequent itemsets, effectively. The experimental results show that PTclose is a suitable algorithm for both dense and sparse datasets with respect to speed and memory consumption. It is more efficient for sparse datasets and has an equal performance for dense datasets.

REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami, "Mining Association Rules between Sets of Items in Large Databases," Proc. ACMSIGMOD Int'l Conf. Management of Data, pp. 207-216, May 1993.

[2] R. Agrawal and R. Srikant, "Fast Algorithms for Mining Association Rules," Proc. Int'l Conf. Very Large Data Bases, pp. 487-499, Sept. 1994.

[3] R. Agrawal and R. Srikant, "Mining Sequential Patterns," Proc. Int'l Conf. Data Eng., pp. 3-14, Mar. 1995.

[4] C. Borgelt, "Efficient Implementations of Apriori and Eclat," Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations, CEUR Workshop Proc., vol. 80, Nov. 2003.

[5] J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang. Hmine: Hyper-structure mining of frequent patterns in large databases. In *Proc. of IEEE Intl. Conference on Data Mining*, pages 441–448, 2001.

[6] J. Han, J. Pei, and Y. Yin, "Mining Frequent Patterns without Candidate Generation," Proc. ACM-SIGMOD Int'l Conf. Management of Data, pp. 1-12, May 2000.

[7] M. Kamber, J. Han, and J. Chiang, "Metarule-Guided Mining of Multi-Dimensional Association Rules Using Data Cubes," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 207-210, Aug. 1997.

[8] J. Liu, Y. Pan, K. Wang, and J. Han. Mining frequent item sets by opportunistic projection. In *Proc. of the 8th ACM SIGKDD Intl. Conference on Knowledge Discovery and Data Mining*, pages 229–238, July 2002.

[9] H. Mannila, H. Toivonen, and I. Verkamo, "Efficient Algorithms for Discovering Association Rules," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 181-192, July 1994.

[10] H. Mannila, H. Toivonen, and I. Verkamo, "Discovery of Frequent Episodes in Event Sequences," Data Mining and Knowledge Discovery, vol. 1, no. 3, pp. 259-289, 1997.

[11] S. Orlando, C. Lucchese, P. Palmerini, R. Perego, and F. Silvestri, "kDCI: A Multi-Strategy Algorithm for Mining Frequent Sets," Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations, CEUR Workshop Proc., vol. 80, Nov. 2003.

[12] N. Pasquier, Y. Bastide, R. Taouil, and L. Lakhal, "Discovering Frequent Closed Itemsets for Association Rules," Proc. Int'l Conf. Database Theory, pp. 398-416, Jan. 1999.

[13] J. Pei, J. Han, and R. Mao, "CLOSET: An Efficient Algorithm for Mining Frequent Closed Itemsets," Proc. ACM SIGMOD Workshop Research Issues in Data Mining and Knowledge Discovery, pp. 21-30, May 2000.

[14] A. Pietracaprina and D. Zandolin, "Mining Frequent Itemsets Using Patricia Tries," Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations, CEUR Workshop Proc., vol. 80, Nov. 2003.

[15] A. Savasere, E. Omiecinski, and S. Navathe, "An Efficient Algorithm for Mining Association Rules in Large Databases," Proc. Int'l Conf. Very Large Data Bases, pp. 432-443, Sept. 1995.

[16] H. Toivonen, "Sampling Large Databases for Association Rules," Proc. Int'l Conf. Very Large Data Bases, pp. 134-145, Sept. 1996.

[17] Wang, J. Han, and J. Pei, "CLOSET+: Searching for the Best Strategies for Mining Frequent Closed Itemsets," Proc. Int'l Conf. Knowledge Discovery and Data Mining, pp. 236-245, Aug. 2003.

[18] Proc. IEEE ICDM Workshop Frequent Itemset Mining Implementations, B. Goethals and M.J. Zaki, eds., CEUR Workshop Proc., vol. 80, Nov. 2003, Available: http://CEUR-WS.org/Vol-90

[19] M.J. Zaki, "Scalable Algorithms for Association Mining," IEEE Trans. Knowledge and Data Mining, vol. 12, no. 3, pp. 372-390, 2000.

[20] M.J. Zaki and C. Hsiao, "CHARM: An Efficient Algorithm for Closed Itemset Mining," Proc. SIAM Int'l Conf. Data Mining, pp. 457-473, Apr. 2002.

[21] M.J. Zaki and K. Gouda, "Fast Vertical Mining Using Diffsets," Proc. ACM SIGKDD Int'l Conf. Knowledge Discovery and Data Mining, pp. 326-335, Aug. 2003.

[22] Q. Zou, W.W. Chu, and B. Lu, "SmartMiner: A Depth First Algorithm Guided by Tail Information for Mining Maximal Frequent Itemsets," Proc. IEEE Int'l Conf. Data Mining, Dec. 2002.

[23] Gosta Grahne and Jianfei Zhu. Efficiently using prefixtrees in mining frequent itemsets. In Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, November 2003.

[24] http://fimi.cs.helsinki.fi, 2003.