ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

A Modified Algorithm for finding Representative Pattern Sets

[1] R.Prabamanieswari, [2] D.S.Mahendran, [3] T.C. Raja Kumar
[1] Associate Professor, Department of Computer Science, Govindammal Aditanar College for Women, Tiruchendur, India
[2] Associate Professor, Department of Computer Science, Aditanar College of Arts & Science, Tiruchendur, India
[3] Associate Professor, Department of Computer Science, St. Xavier's College, Tirunelveli, India

Abstract: **- Many algorithms and techniques are developed for enumerating itemsets from transactional databases. They produce a large number of frequent itemsets when the minimum support threshold is low and /or the dataset is dense. A large number of discovered patterns makes further analysis of generated patterns troublesome. Therefore, it is important to find a small number of representative patterns to best approximate all other patterns. This paper modifies the algorithm MinRPset for finding representative pattern sets. It follows the same concepts such as ∈- covered and greedy method in MinRPset but, it uses NCFP-tree instead of CFP-tree for storing frequent itemsets in a compressed manner. The experiment results show that our algorithm gives better execution time to generate representative patterns sets for the mushroom dataset in an efficient manner comparing to the algorithm MinRPset.**

***Keywords*: Frequent itemset, NCFP-tree, greedy method, representative pattern sets.**

I. INTRODUCTION

Data mining is the step in the knowledge discovery process that attempts to discover novel and meaningful patterns in data. It is the process of extracting previously unknown potentially useful hidden predictive information from large amounts of data. Many algorithms and techniques are posed for enumerating itemsets from transactional databases. Let the transactional database D= $\{t_1, t_2, \ldots, t_n\}$, where t_j is a transaction containing a set of items, $j \in [1,n]$. Let $I = i_1, i_2 \ldots, i_m$ be a set of m distinct attributes and t be transaction that contains a set of items such that $T \subseteq I$. Each subset of I is called an itemset. If an itemset contains k items, then the itemset is called a k-itemset. The support of itemset X in database D is defined as the percentage of transactions in D containing X, that is, support $t_D(X) = \{t \mid t \in D \text{ and } X \subseteq t\}/D$. If the support of a pattern X is larger than a user specified threshold min-sup (min-sup $\in (0, 1]$), then X is called a frequent pattern. Given a transaction database D and a minimum support threshold min-sup, the task of frequent pattern mining is to find all the frequent patterns in D with respect to min-sup. In the early days, the size of the database and the generation of a reasonable amount of frequent itemsets were considered as the most costly aspects of frequent itemset mining, and the most energy went into minimizing the number of scans through the database. However, if the minimal support threshold is set too low, or the data is highly correlated, the number of frequent itemsets itself can be prohibitively large. To overcome this problem, recently several proposals have been made to construct a concise representation based on lossless compression methods such as closed itemsets [1]-[6] and constraints based frequent itemsets [7]-[10] instead of mining all frequent itemsets. The constraint based mining though useful, but can hardly be used for pre-computation, since different users are likely to have different constraints. The closed itemsets concise representation gives the less number of frequent representative patterns and all the frequent itemsets can be derived from them with exact support value. But, most of the applications will not need precise support information for frequent pattern. They approximate the frequency of every frequent pattern with a guaranteed maximal error bound. The good approximation algorithms such as RPglobal and RPlocal[18] need to perform substantial coverage checking that checks whether an itemset can be covered by another one in order to find representative patterns. They are very time-consuming and space-consuming. To improve the performance, RPglobal and RPlocal have to use some FP-tree like structures to index frequent itemsets and representative itemsets to reduce the number and the cost of coverage checking. The approximation algorithm MinRPset[20] utilizes several techniques to reduce the running time and the memory usage. In particular, it uses a tree structure called CFP-tree [21] to store frequent patterns compactly to find subsets when finding representative patterns. The proposed method

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

ModifiedRPset utilizes similar CFP-tree such as NCFP-tree to find frequent patterns and follows the remaining procedures which are same in MinRPset algorithm for generating representative pattern sets. It does not use a light-weight compression technique to compress *C(X)*s The rest of the paper is organized as follows: Section II presents the related work. Section III describes the proposed method for generating representative pattern sets with example. The experimental results are shown in section IV. Finally, section V concludes the paper.

II. RELATED WORK

Previously, the ideas of approximating frequent patterns have been probed in some related studies. For example, Mannila and Toivonen [11] show that approximate association rules are interesting and useful. In [14], the notion of free-sets is proposed and it can be used to approximate closely the support of frequent itemsets. However, none of these studies systematically explored the problem of designing and mining condensed frequent pattern bases with a guaranteed maximal error bound. Disjunction-free generators [13] and δ-free sets [14] give the support of all frequent patterns approximately. Both disjunction-free generators and δ-free sets also require a border to be lossless. Jian Pei et al [12] consider two types of condensed FP-bases: the downward condensed FP-base Bd and the max-pattern-based condensed FP-base Bm. They also specify that computing a condensed FP-base can also be performed on a relative, percentage based error bound k% instead of an absolute error bound k. In that case, supub−suplb / suplb≤ k% should be satisfied for frequent patterns. But, all these approaches are less efficient than the approach which is based on closed pattern approach to approximate the support count.

Recently, several approaches have been proposed to tackle the concise representation of frequent itemsets, two key criteria being employed for evaluating the concise representation of itemsets are the coverage criterion and frequency criterion. The methods like top-k frequent patterns [15], top-k redundancy-aware patterns [16], and error-tolerant patterns [17]try to rank the importance of individual patterns, or revise the frequency concept to reduce the number of frequent patterns. But, choosing an appropriate k for a given domain is usually not easy and there are no theoretical guarantees on the level of approximation for a given k. The major problem with these approaches is that the frequency (or the support measure) is not considered. However, these methods generally do not provide a good representation of the collection of frequent patterns. Therefore, this study concentrates both the key criteria for the concise representation of itemsets.

Xin et al. [18] propose the concept of δ-covered to generalize the concept of frequent closed pattern. A pattern X1 is δ-covered by another pattern X2 if X1is a subset of X2 and (supp(X1) − supp(X2))/supp(X1) ≤ δ. They develop two algorithms, RPglobal and RPlocal. These algorithms need to perform substantial coverage checking that checks whether an item set can be covered by another one. RPglobal is very time-consuming and space-consuming. It is feasible only when the number of frequent patterns is not large. RPlocal is very efficient, but it produces more representative patterns than RPglobal. To improve the performance, RPglobal and RPlocal have to use some FP-tree-like structures to index frequent item sets and representative item sets to reduce the number and the cost of coverage checking.

Jianzhong Li et al. [19] devise two algorithms, RP-FP and RP-GD, to mine a representative set that summarizes frequent sndub graphs. RP-FP derives a representative set from frequent closed sub graphs, whereas RP-GD mines a representative set from graph databases directly. Based on the concept of δ –cover, they proposed three new concepts like jump value, δ –jump pattern, and δ -cover graph but these concepts are only used for graph mining.

Liu et al [20] analyse the bottlenecks of RPglobal and RPlocal and develop two algorithms, MinRPset and FlexRPset, to solve the problem. The algorithm MinRPset is similar to RPglobal, but it utilizes several techniques to reduce running time and memory usage. In particular, MinRPset uses a tree structure called CFP-tree [28] to store frequent patterns compactly. The algorithm FlexRPset is developed based on MinRPset. It provides one extra parameter K, which allows users to make a trade-off between efficiency and the number of representative patterns selected. In [18] and [20], the relative error (supp(X1) − supp(X2))/supp (X1) is used. But, MinRPset has the extra benefits besides giving fewer representative patterns.

MinRPset uses the compressed tree structure such as CFP-tree for storing all frequent itemsets which is comparable to that of storing frequent closed itemsets. The CFP-tree structure also supports efficient retrieval of patterns that are δ-covered by a given pattern. Hence, our study focuses the modified CFP-tree such as NCFP-tree [22] and applies the MinRPset algorithm for finding minimum representative pattern sets.

III. PROPOSED ALGORITHM

Our proposed algorithm ModifiedRPset is similar to MinRPset. In our approach, first we mine the patterns with with support ≥ min_sup * (1−□_) and store them in a NCFP-tree. Then, we use NCFP-tree and generate C(X)—the set of frequent patterns that X covers—for every

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

pattern X ∈ Fˆ. Here, we take F be the set of frequent patterns in a dataset D with respect to threshold min_sup and Fˆ be the set of patterns with support no less than min_sup * (1−□) in D. Therefore, obviously F ⊆ Fˆ .While finding C(X) for every pattern X ∈ Fˆ, we get | Fˆ| sets. The elements of these sets are frequent patterns in F. i.e., S = {C(X) | X ∈ Fˆ} corresponds to frequent patterns in F. Finally, we find minimum number of representative pattern sets. Here, we use the optimal algorithm greedy method [23] to find the minimum number of representative pattern sets. As a result, finding a minimum representative pattern sets is now equivalent to finding a minimum number of sets in S that can cover all the frequent patterns in F.The overview of our proposed approach is given below:

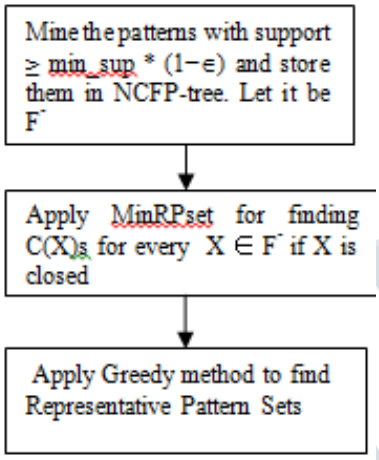

Fig 1. Overview of ModifiedRPset

The proposed algorithm is given below:

Algorithm: ModifiedRPset
Input:
D is the database
min-supp is the minimum support threshold
Output:
Minimum number of Representative Pattern Sets

Description:
1. Mine patterns with support ≥ min_sup * (1−□) and store them in a NCFP-tree
2. Apply MinRPset for finding C(X)s
3. Remove non-closed entries from C(X)s
4. Apply the greedy set cover algorithm on C(X)s to find minimum number of Representative Pattern Sets

A. EXAMPLE

Consider the Transactional Database given in Table I. It has seven transactions, that is |D|=7. Assume min-sup=40%. The set of frequent itemsets are determined and are given in Table II.

Table I. Transaction Database

TID	TRANSACTION
1	a, c, e, f, m, p
2	a, b, f, m, p
3	a, b, d, f, g
4	d, e, f, h, p
5	a, c, d, m, v
6	a, c, h, m, s
7	a, f, m, p, u

Table II. Frequent Itemsets

All Patterns (min_sup = 40%)
c:3, d:3, p:4, f:5, m:5, a:6
cm:3, ca:3, pf:4, pm:3, pa:3, fm:3, fa:4, ma:5
cma:3, pfm:3, pfa:3, pma:3, fma:3
pfma:3

The above set of frequent itemsets is stored in a NCFP-tree in a compressed form. The following Fig shows the NCFP-tree and Table III shows the corresponding compact representation of frequent itemsets

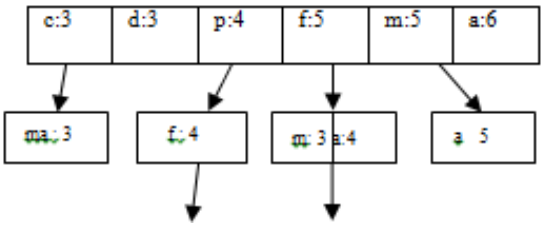

Fig. 2 NCFP-tree Construction

Table III. Frequent itemset (compact form)

Frequent Itemsets(Compact Form) (min_sup = 40%)
cma:3
d:3
pfma:3 pf:4
fma:3 fa:4 f:5
ma:5
a:6

Now, we have to find out C(X)s from Fig 2 by applying MinRPset algorithm. The C(X) contains the subsets of X that can be □-covered by X, for every X ∈ Fˆ. (Fˆ denotes frequent patterns with support ≥ min_sup * (1−□) and F denotes frequent patterns with support ≥ min_sup). The C(X) is determined only for closed patterns. Therefore, for

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 5, Issue 3, March 2018

each itemset belongs to Table III is first checked whether it is closed or not then C(X) is determined. This example determines C(X)s only for the itemsets which are given in Table IV.

Table IV. Closed Frequent Itemsets

Closed Patterns (min-sup=40%)
f:5, a:6
pf:4, fa:4,ma:5
cma:3
pfma:3

For example, let X= pfma with supp=3 and □=0.25. We find C(X) = {{fma: 3}, {pf: 4}, {fa: 4}}. Similarly, we find C(X) for each itemset X. Finally greedy set cover algorithm is applied for C(X)s and minimum representative pattern sets is discovered as { {pfma:3}, {cma:3}, {d:3}}.

IV. EXPERIMENTAL RESULTS

The experiments are carried out on the computer with the configuration such as Intel(R) Core(TM) i3CPU, 3 GB RAM, 2.53 GHz Speed and Windows 7 Operating System. The MinRPset and ModifiedRPset approaches are implemented in java. They do not use a light-weight compression technique to compress C(X)s during the implementation. The experiments are evaluated on mushroom dataset. The mushroom dataset contains the characteristics of various species of mushrooms. It has 119 items and 8124 transactions. The minimum, maximum and average length of its transaction is 23. It is obtained from the UCI repository of machine learning databases.

Running Time

Here, the experiment considers execution time only for finding performance of the algorithms. The two algorithms are tested on the mushroom dataset.

Fig 3 and Fig 4 show the running time of MinPRset and ModifiedRPset algorithms. Fig 3 shows when changing min-supp from 10% to 90% in increments of 20% and □ = 0, there is a small difference between ModifiedRPset and MinRPset. Similarly, Fig 4 also shows when changing □ from 0.1 to 0.5 and min-supp is fixed as 0.4, ModifiedRPset gives better performance than MinRPset

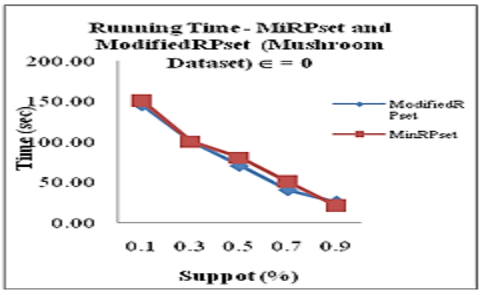

Fig 3. Running Time when □ = 0

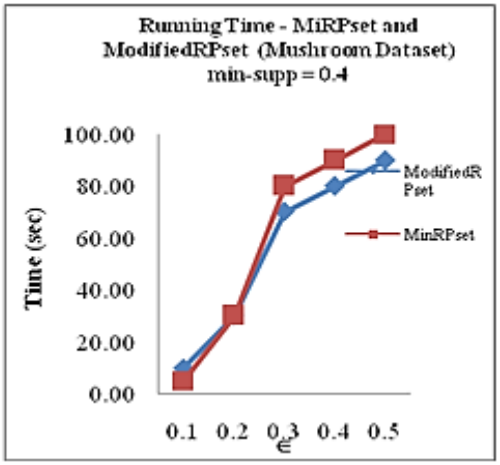

Fig 4. Running Time when min-supp =0.4

V. CONCLUSION

In this paper, we considered NCFP-tree and the generalized version of set covering called Greedy method to find the representative pattern sets. The Greedy algorithm may provide an efficient solution that is close to optimal. But, there is no general template on how to apply the greedy method to a given problem. As for future research, we will investigate how to remove the already covered sets by applying modification in the known greedy solution to give better results.

REFERENCES

[1] Pasquier N, Bastide Y, Taouil R and Lakhal, "Pruning Closed Itemset Lattices for Association Rules", Proc. BDA conf., pp 177-196, 1998.

[2] Pasquier N, Bastide Y, Taouil R and Lakhal L, "Efficient Mining of Association Rules using Closed Itemset Lattices", Information Systems, vol 24, No 1, pp 25-46,1999.

[3] Pasquier N, Bastide Y, Taouil R and Lakhal, " Discovering frequent closed itemsets for association rules", In: Proc 7th Int Conf on Database Theory (ICDT'99), Jerusalem, Israel, pp 398–416,1999.

[4] Zaki M, "Generating non-redundant association rules", In: Proc 2000 ACM SIGKDD Int Conf Knowledge Discovery in Database(KDD'00), Boston, USA, pp 34–43, 2000.

[5] J. Pei, J. Han, and R. Mao, "CLOSET: An efficient algorithm for mining frequent closed itemsets",In ACM

SIGMOD'00 Workshop on Research Issues in Data Mining and Knowledge Discovery, pp 21–30, 2000.

[6] J. Wang, J. Han and J. Pei, "Closet+: Searching for the best strategies for mining frequent closed itemsets," in Proc. KDD, New York, NY, USA, pp. 236–245, 2003.

[7] Ng R, Lakshmanan LVS, Han J and Pang A, "Exploratory mining and pruning optimizations of constrained associations rules", In: Proc 1998 ACM-SIGMOD Int Conf Management of Data(SIGMOD'98), Seattle, USA, pp 13–24, 1998.

[8] Lakshmanan LVS, Ng R, Han J and Pang A, "Optimization of constrained frequent set queries with2-variable constraints", In: Proc 1999 ACM-SIGMOD Int Conf Management of Data (SIGMOD'99),Philadelphia, USA, pp 157–168,1999.

[9] Pei J, Han J and Lakshmanan LVS, "Mining frequent itemsets with convertible constraints", In: Proc 2001Int Conf Data Engineering (ICDE'01), Heidelberg, Germany, pp 433–332, 2001.

[10] R. Srikant, Q. Vu, R. Agrawal, Mining association rules with item constraints, in: Proceedings of the 3rd ACMSIGKDD Conference, 1997, pp. 67–73.

[11] Mannila H and Toivonen H, "Multiple uses of frequent sets and condensed representations (extended abstract)", In: Knowledge Discovery and Data Mining, pp 189–194,1996.

[12] Jian Pei, Guozhu Dong, Wei Zouand Jiawei Han, "Mining Condensed Frequent-Pattern Bases",Knowledge and Information Systems, 2004, DOI 10.1007/s10115-003-0133-6.

[13] A. Bykowski and C. Rigotti, "A condensed representation to find frequent patterns," in Proc. PODS, New York, NY, USA,pp. 267-273, 2001.

[14] J.F. Boulicaut, A. Bykowski, andC. Rigotti, "Free-sets: A condensed representation of boolean data for theapproximation of frequency queries," Data Mining Knowl. Discov., vol. 7, no. 1, pp.5–22, 2003

[15] J. Wang, J. Han, Y. Lu, and P. Tzvetkov, "TFP: An efficient algorithmfor mining top-k frequent closed itemsets," IEEE Trans. Knowl. Data Eng., vol. 17, no. 5, pp. 652–664, May 2005.

[16] D. Xin, H. Cheng, X. Yan, and J. Han,"Extracting redundancy-aware top-k patterns," in Proc. KDD,Philadelphia, PA, USA,pp. 444–453,2006.

[17] M. T. Yang, R. Kasturi, and A. Sivasubramaniam., "An AutomaticScheduler for Real-Time Vision Applications", In Proceedings of the International Parallel and Distributed Processing Symposium(IPDPS), 2001.

[18] D. Xin, J. Han, X. Yan, and H. Cheng, "Mining compressed frequen-pattern sets," in Proc. 31st Int. Conf.VLDB, Trondheim, Norway, pp. 709–720,2005.

[19] Jianzhong Li, Yong Liu, and Hong Gao, Efficient Algorithms for Summarizing Graph Patterns, IEEE Transactions on Knowledge and Data Engineering, Vol. 23, No. 9,pp 1388-1405, 2011.

[20] Guimei Liu, Haojun Zhang, and Limsoon Wong, "A Flexible Approach to Finding RepresentativePattern Sets", IEEE Transactions on Knowledge and Data Engineering, Vol. 26, No. 7,pp 1562-1574,, July 2014.

[21] G. Liu, H. Lu, and J. X. Yu, "CFP-tree: A compact disk-based structure for storing and querying frequent itemsets", Inf. Syst., vol. 32, no.2, pp. 295–319, 2007.

[22] R.Prabamanieswari, D.S.Mahendran, T.C. Raja Kumar, "NCFP-tree: A Non-Recursive Approach to CFPtree using Single Conditional Database", International Journal for Research in Applied Science & Engineering Technology (IJRASET), Volume 5 Issue XI November 2017.

[23] V. Chvatal, "A greedy heuristic for the set-covering problem,"Math. Oper. Res., vol. 4, no. 3, pp. 233–235, 1979.