IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Volume-6, Issue-3 E-ISSN: 2347-2693

Comparative Study of FPclose, CFPtree-closed and NCFPGEN Algorithms

R. Prabamanieswari[1*], D.S. Mahendran[2], T.C. Raja Kumar[3]

[1*] Department of Computer Science, Govindammal Aditanar College for Women, Tiruchendur, India
[2]Department of Computer Science, Aditanar College of Arts and Science, Tiruchendur, India
[3]*Department of Computer Science, St. Xaviers College, Tirunelveli,* India

[*]*Corresponding Author: prabacs_2006@yahoo.com, Tel.: 919894796214*

Available online at: www.ijcseonline.org

Received: 24/Feb//2018, Revised: 03/Mar2018, Accepted: 23/Mar/2018, Published: 30/Mar/2018

***Abstract*—**Closed itemset mining plays an important role in mining compressed representations or summaries of the set of frequent patterns. It uses the support information of itemsets and the superset–subset relationship among itemsets for removing redundancy. Closed itemset is always smaller or equal in cardinality comparing to other concise representation such as frequent free sets and generators. But, the compression using this closed-pattern approach may not be very effective, since slightly different count often exists between super and sub patterns. This paper focuses to compare two closed itemset algorithms such as FPclose and CFPtree-closed and a frequent itemset algorithm NCFPGEN for studying and determining the performance of these algorithms in the execution time aspect.

***Keywords*—**frequent itemset, closed itemset

I. INTRODUCTION

Data Mining is the process of extracting previously unknown and potentially useful hidden predictive information from large amounts of data. Frequent pattern mining is a fundamental research topic that has been applied to different kinds of databases. 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 [2-7] [22] and constraints based frequent itemsets [8-11] 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 representation gives the less number of frequent representative patterns based on support value. This concise representation not only uses the support information of itemsets but also uses the superset–subset relationship among itemsets for removing redundancy. However, the compression using the closed-pattern approach may not be very effective, since slightly different count often exists between super and sub patterns. This paper focuses to compare both closed itemset algorithms FPclose and CFPtree-closed with a frequent itemset algorithm NCFPGEN for generating NCFP-tree which stores all frequent itemsets in compressed manner. It analyses these algorithms because they are based on similar FP-tree concept and they follow the depth-first search technique.

The rest of the paper is organized as follows: Section I contains the introduction of frequent itemset mining. Section II gives the basic definitions of frequent itemsets and closed itemsets. Section III describes the related work. Section IV discusses the algorithms FPclose and CFPtree-closed and NCFPGEN for generating NCFP-tree with example. The experimental results are shown in section V. Section VI describes the results and discussion of the algorithms FPclose and CFPtree-closed and NCFPGEN. Finally, Section VI concludes the paper.

II. BASIC DEFINITIONS

This section gives the basic definitions of frequent itemsets and frequent closed itemsets. Let $I=I_1, I_2, \ldots, I_m$ be a set of m distinct attributes, T be transaction that contains a set of items such that $T \subseteq I$ and D be a database with different transaction records Ts.

Definition 1. (itemset).
An itemset X is a finite subset of I, the set of possible items.
Definition 2. (transactions and transaction databases).

A transaction t is a pair ≺i, X≻ consisting of a transaction identifier tid (t) = i ∈ N and an itemset (t) = X ⊆ I. A transaction database D is a set of transactions with unique transaction identifiers. The set SD of all itemsets in D is SD = { X: ≺i, X≻ i ∈D }.

Definition 3. (support/frequencies).

The support of X in D is supp(X, D) = |cover(X, D)| where cover (X, D) = {tid | (tid, I) D, X ⊆I}.

Definition 4. (Frequent itemset mining).

Given a transaction database D and a real value σ ∈ [0, 1], find all σ frequent itemsets, i.e., determine the collection F (σ, D) = {X⊆I: support (X, D) ≥ σ} of σ – frequent itemsets in D.

Definition 5. (Closed frequent itemsets).

An itemset X ∈ F (σ, D) is closed, if there exists no proper super-itemset Y such that Y has the same support count as X in F. The collection of closed σ -frequent itemsets in D is denoted by C (σ, D).

III. Related Work

Many algorithms and techniques are proposed for enumerating itemsets from transactional databases. It has been observed that the complete set of frequent patterns often contains a lot of redundancy [21] i.e.) many frequent patterns have similar items and supporting transactions. To overcome this problem, several approaches have been made to construct a concise representation of the frequent itemsets. Two major approaches have been developed in this direction: lossless compression and lossy approximation. The closed frequent patterns [3] and non derivable itemsets [13] methods are generally referred to as lossless compression since we can fully recover the exact frequency of any frequent itemsets. The maximal frequent pattern [17] is called as lossy compression since we cannot recover the exact frequencies. In addition to these approaches, recently many proposals such as generators [14], disjunction-free generators [15], δ-free sets [16], top-k frequent closed patterns [18] and redundancy-aware top k patterns [19] have been made to construct a concise (compressed) representation of the frequent itemsets, instead of mining all frequent itemsets. But, the type of concise representation that received a lot of attention in the literature is the closed itemsets because the number of closed patterns is always smaller or equal in cardinality than the set of frequent free sets [15-16], lesser than that of generators [14] and the number of non derivable patterns [13] is larger than that of closed patterns on some datasets. Furthermore, the set of generators itself is not lossless. Hence, this study focuses to concentrate on closed itemset.

Frequent closed patterns preserve the exact support of all frequent patterns. The concept of closed frequent patterns is proposed by Pasquier et al. The Close [2-3] and the AClose [4] algorithms perform the breadth first search for the generators of the frequent closed itemsets in a level wise manner. These kinds of patterns are concise in the sense that all of the frequent patterns can be derived from them. Unfortunately, the number of patterns generated in these approaches is still too large to handle. CLOSET [6] is an extension of the FP-growth algorithm [1] which constructs a frequent pattern tree FP-tree and recursively builds conditional FP-trees in a bottom-up tree search manner. Although CLOSET uses several optimization techniques to enhance the mining performance, its performance still suffers in sparse datasets or when the support threshold is low. The algorithm CLOSET+ [7] uses one global prefix-tree for keeping track of all closed itemsets.

IV. Discussion of FPclose and CFPtree-closed and NCFPGEN with example

FPclose [12] is one of the best algorithms for mining closed frequent itemsets, even when compared to CLOSET+. In this algorithm, a CFI-tree (closed frequent itemsets- tree), another variation of the FP-tree, is used for testing the closeness of frequent itemsets. CFI-tree depends on FP-tree T_X and is denoted as C_X. The itemset X is represented as an attribute of T, T: base. The CFI-tree C_X always stores all already found CFIs containing itemset X and their counts. In a CFI-tree, each node in the sub tree has four fields: item-name, count, node-link, and level. Here, level is still used for subset testing. The count field is needed because when comparing Y with a set Z in the tree, it is not the case that Y ⊂ Z and Y and Z have the same count is verified. The order of the items in a CFI-tree's header table is the same as the order of items in header table of its corresponding FP-tree.

In FPclose, the insertion of a CFI into a CFI-tree is similar to the insertion of a transaction into an FP-tree, except now the count of a node is not incremented; it is always replaced by maximum count up-to-date. Suppose a newly found frequent itemset Y which contains X, then only needs to be compared with the CFI's in CFI-tree is C_X. In CFI-tree, if there is no superset of Y with same support as Y, then Y is considered as closed. The item order in FP-tree and CFI-trees is same because they are both for baseϕ. Here, a node x: l: c means that the node is for item x, its level is l and its count is c.

The CFPtree-closed algorithm [20] utilizes CFP-tree for finding frequent closed itemsets. The CFP-tree is constructed from a database with respect to a given minimum support threshold which is lossless, that is, it contains the complete set of frequent itemsets. The CFP-tree structure is different from the FPtree structure proposed by Han et al. [1] on several aspects: (1) A FP-tree stores projected transactions during the mining process, while a CFP-tree stores discovered frequent itemsets. (2) There is only one pointer in a CFP-tree entry, but there are four pointers in a FP-tree

node. Hence the CFP-tree structure is more disk-friendly than the FP-tree structure. (3) Most importantly, there is only prefix sharing in FP-trees, while there are both prefix sharing and suffix sharing in CFP-trees. The space saved by suffix sharing is much more significant than that by prefix sharing. Suffix sharing makes it feasible to compute and store all frequent itemsets in a CFP-tree even when it is not feasible to store all frequent itemsets in other structures. Here, the path from root to particular node represents particular frequent pattern.

The concise representations of frequent itemsets such as frequent closed itemsets, generators, disjunction-free sets, non-derivable frequent itemsets and cover equivalent classes are efficiently produced from a CFP-tree using the query processing algorithms because these concise representations use the support information of itemsets and the superset–subset relationship among itemsets to remove redundancy. The closed itemsets in a CFP-tree are identified as follows: For each itemset l in the CFP-tree, the search for its subsets in the CFP-tree is performed. If a subset of l has the same support as l, then the subset is marked as non-closed. We traverse the CFP-tree in depth-first order from left to right, and search for the subsets of every itemset being visited. According to the left containment property, when an itemset is visited, all the supersets of the itemset have been visited except those that are in the sub tree pointed by l. Therefore, if an itemset is not marked as non-closed when it is visited, then the itemset is a closed itemset if all the entries in its child node are less frequent than it.

The NCFPGEN algorithm [23] generates frequent itemsets using NCFP-tree. The NCFP-tree (Non-Recursive CFP-tree) is similar to CFP-tree structure but, it differs in processing the conditional database. The main difference is that a new single extended conditional database is created initially instead of creating multiple conditional databases recursively. In CFP-tree, a new conditional database is created for each itemset every time by applying push-right step. The NCFP-tree utilizes the extended conditional database in an efficient manner by keeping the transactions start with the itemsets which are not yet processed. It does not follow the recursive approach.

Consider the following Transactional Database. It has seven transactions, that is |D|=7. Assume min-sup=40%. The set of frequent itemsets and frequent closed itemsets are discovered. Table 2 shows all frequent itemsets and table 4 shows frequent itemsets in compact form. The closed itemsets are given in table 3.

Table 1. 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 2. 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

Table 3. Closed Frequent Itemsets

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

Table 4. 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

It is known that CFP-tree and NCFP-tree store the same number of frequent itemsets in compact form. Therefore, both trees represent the same table 4. Similarly, FPclose and CFP-tree closed give same number of closed itemsetes which is shown in table 3.From these tables, we observe that the number of compact form frequent itemsets is lesser than all frequent itemsets but, it is slightly larger than the number of closed itemsets.

The CFI-tree corresponds to FPclose is given in figure1 and CFP-tree / NCFP-tree corresponds to CFP-tree closed and NCFPGEN is given in figure 2.

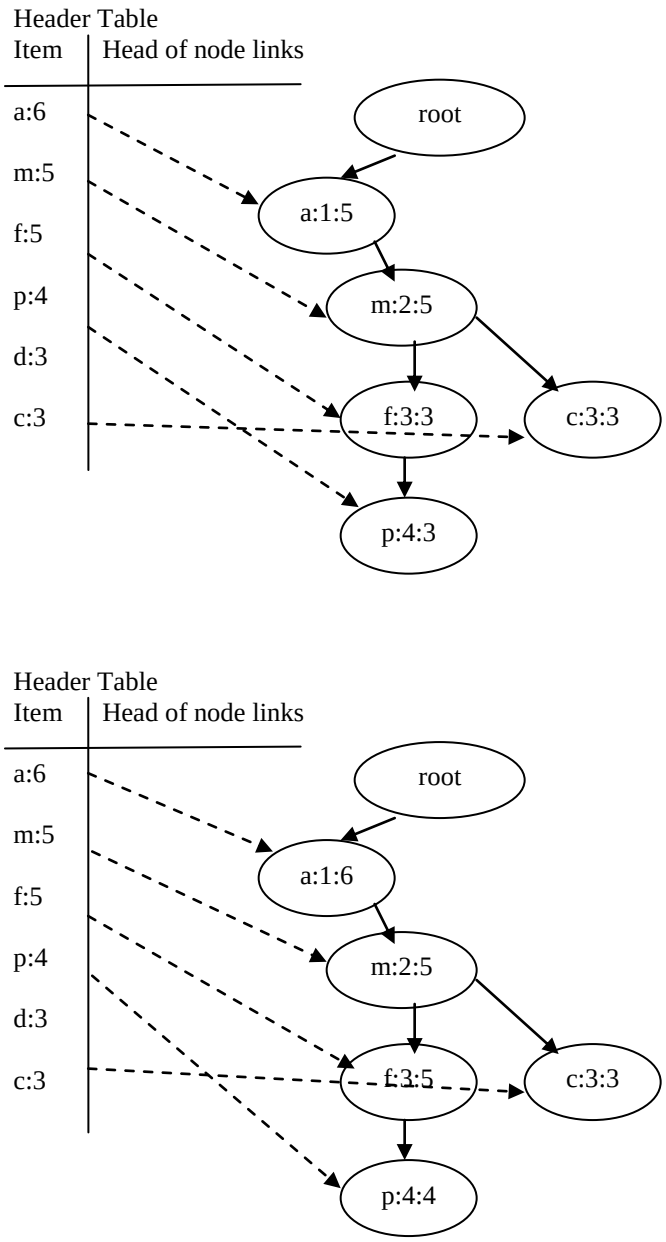

Fig. 1 (a) & (b) CFI-tree Construction

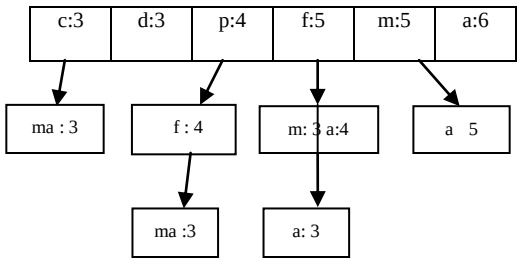

Fig. 2 CFP-tree / NCFP-tree Construction

In figure 1 (a), first we insert (a,m,f,,p) and (a,m,c) with count 3. Then, we insert (a,m) with count 5.Therefore, the counts for nodes a and m are both changed to be 5.In figure 1 (b), the remaining CFI's (a,f) :4, (f,p) :4, (f) :5 and (a) :6 are inserted. Now, the resultant CFI-tree contains all mentioned CFI's in table 3.

In figure 2, we either create single entry node or multiple entries node depends on the count of the frequent itemsets. A single entry contains multiple items if it is the only child of its parent. The frequent itemset ma: 3 is entered as single entry and the frequent itemset m: 3 and a: 4 are entered as multiple entries while driving from their parent.

V. 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 algorithms such as FPclose, CFPtree-closed and NCFPGEN for creating NCFP-tree approaches are implemented in Java. The experiments are evaluated on two datasets namely mushroom and retail. The mushroom and retail are real datasets. They are relatively dense. 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. The retail dataset contains the retail market basket data from an anonymous Belgian retail store. It has 16,470 items and 88,162 transactions. The maximum length of its transaction is 77 and the average length of its transaction is10.31. Both are obtained from the UCI repository of machine learning databases. The algorithms are tested on the mushroom dataset with a support level of 10% to 50% in increments of 10%.They are also tested on the retail dataset with a support level of 0.01% to 0.05% in increments of 0.01%.

Running Time

Here, the experiment considers CPU time only for finding performance of the algorithms. The figure 1 shows the running time of FPclose, CFPtree-closed and NCFPGEN algorithms. G. Liu et al [20] showed that the CFPtree-closed algorithm is slightly better than FPclose even though it generates frequent closed itemsets using a post-processing strategy. They also showed that constructing a CFP-tree for storing all frequent itemsets is much more efficient than generating frequent closed itemsets especially on dense datasets such as mushroom and retail. The following figure shows that NCFPGEN algorithm is better for these datasets. The paper [23] discusses that NCFP-tree is better than CFP-tree for the datasets such as mushroom, retail and T10I4D100K. It also describes that NCFP-tree provides all features similar to CFP-tree. The figure 1 shows that NCFP-tree is comparable to both FPclose, CFPtree-closed algorithms.

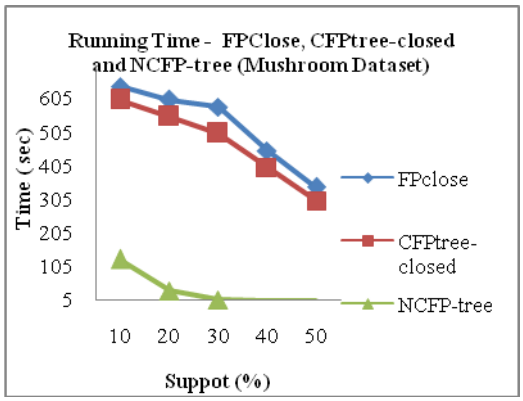

Fig. 1(a)

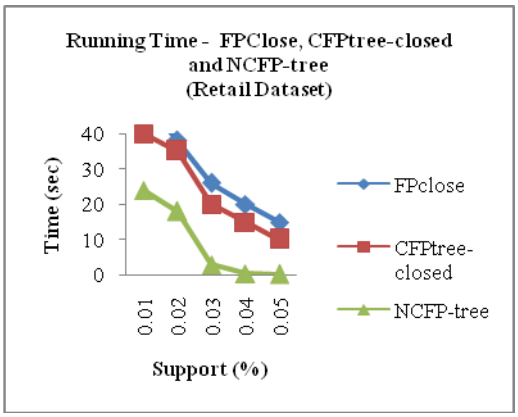

Fig. 1(b)
Fig. 1(a) – 1(b) Running Time

VI. RESULTS AND DISCUSSION

The algorithms FPclose, CFPtree-closed and NCFPGEN are based on similar FP-tree concept and they follow the depth-first search technique. Each algorithm has its own strengths and limitations. In FPclose algorithm, the construction of the CFI-tree is mostly similar to FP-tree when inserting new transactions. But, it does not support for finding other concise representations such as generators, disjunction-free sets, non-derivable frequent itemsets etc. In CFPtree-closed algorithm, the CFP-tree is utilized. The CFP-tree contains the complete set of frequent itemsets and it supports for producing concise representations of frequent itemsets using query processing algorithm. In NCFPGEN algorithm, a similar CFP-tree such as NCFP-tree is created. This NCFP-tree also supports for producing concise representations of frequent itemsets. The running time shows that NCFPGEN algorithm is comparable to both FPclose, CFPtree-closed algorithms

VII. CONCLUSION

.
We have compared three algorithms such as FPclose, CFPtree-closed and NCFPGEN with two datasets in this paper. The running time shows that NCFPGEN algorithm is comparable to both FPclose, CFPtree-closed algorithms.

In future, NCFPGEN algorithm may be compared with other closed itemset algorithms in various aspects to find out the effectiveness of NCFPGEN algorithm because this algorithm supports for producing concise representations of frequent itemsets. We believe that NEFPGEN algorithm may contribute a good support to produce concise representations with further research efforts.

REFERENCES

[1] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation", In Proceedings of ACM SIGMOD'00, pp 1–12, May 2000.

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

[3] 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.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] 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.

[9] 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.

[10] 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.

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

[12] Gosta Grahne and Jianfei Zhu, "Fast Algorithms for Frequent Itemset Mining Using FP-Trees", IEEE Transactions on Knowledge and Data Engineering, vol. 17, no. 10, October 2005.

[13] Calders and B. Goethals., "Mining all non-derivable frequent itemsets", In Proc. of 2002 European Conf. on Principles of Data Mining and Knowledge Discovery(PKDD'02), pp 74–85, 2002.

[14] Y. Bastide, N. Pasquier, R. Taouil, G. Stumme, and L.Lakhal, "Mining minimal non-redundant association rules using frequent

closed itemsets," in Proc. 1st Int. Conf. CL, London, U.K., pp. 972–986, 2000.
[15] A.Bykowski and C. Rigotti, "A condensed representation to find frequent patterns," in Proc. PODS, New York, NY, USA, pp. 267-273, 2001.
[16] J.F. Boulicaut, A. Bykowski, and C. Rigotti, "Free-sets: A condensed representation of boolean data for the approximation of frequency queries," Data Mining Knowl. Discov., vol. 7, no. 1, pp.5–22, 2003.
[17] R. J. Bayardo, "Efficiently mining long patterns from databases,"in Proc. SIGMOD, New York, NY, USA, pp. 85–93, 1998.
[18] J. Wang, J. Han, Y. Lu, and P. Tzvetkov, "TFP: An efficient algorithm for mining top-k frequent closed itemsets," IEEE Trans. Knowl. Data Eng., vol. 17, no. 5, pp. 652–664, May 2005.
[19] 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.
[20] 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.
[21] R.Prabamanieswari, D.S.Mahendran, T.C. Raja Kumar, "A Survey on Concise and Lossless Representation of Frequent Pattern Sets", International Journal of Advanced Research in Computer and Communication Engineering, Vol. 4, Issue 9, September 2015.
[22] Caiyan Dai and Ling Chen, "An Algorithm for Mining Frequent Closed Itemsets with Density from Data Streams", International Journal of Computer Sciences and Engineering, Vol. 4(2), pp. 40-48, Feb 2016.
[23] 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.

Authors Profile

Mrs R.Prabamanieswari is currently pursuing Ph.D degree. She is working as Asssociate Professor in Computer Science, Govindammal Aditanar College for Women, Tiruchendur. She has attended National and International Seminars and Conferences and published few papers. Her area of research is Data Mining.

D.S.Mahendran is working as Associate Professor in Computer Science, Aditanar College, Tiruchendur. He received M.Sc. Physics and PBDCSA degrees from Madurai Kamaraj University , M.Phil and Ph.D degrees in Computer Science from Alagappa University, Karaikudi. His research interest includes Computer algorithms, Ad-Hoc Networks, Network Security and Cloud Computing.

Prof. T.C.RajaKumar has been working as Associate Professor of the Department of Computer Science in St. Xavier's College (Autonomous), Tirunelveli. He has completed his M.Sc. (Computer Science) in Bharathidasan University, Tiruchirappali, and M.Phil. (Computer Science) in Alagappa University, Karaikudi and Ph.D. (Computer Science) in Manonmaniam Sundaranar University, Tirunelveli. He is a member of IEEE and also a member of curriculum development committees of various Universities and Autonomous colleges of Tamilnadu. He has attended so many National and International Seminars, Conferences and published more than 20 scientific papers in National and International Journals. His area of research is Digital Image Processing and Data Mining.

