International Journal of Engineering and Advanced Technology (IJEAT)
ISSN: 2249-8958 (Online), Volume-8 Issue-6, August, 2019

Partition based Single Scan Method for Mining Frequent Item Sets

U. Mohan Srinivas, E. Srinivasa Reddy

Abstract*: *Frequent Itemset mining (FIM) concept and limitations are explored in this paper, for the purpose of extracting unknown hidden patterns as itemsets from the transactional database. Since candidate generation and support calculations are the major tasks in FIM, the major limitations of FIM are tackled, (i) huge possible frequent itemsets are generated as candidates at each pass (ii) Data base scan at each pass to calculate the support of the generated itemsets (iii) generated itemsets are highly sensitive to the minimum support threshold. SS-FIM a single scan algorithm is to deal with the above limitations. However, several unnecessary itemsets are being hashed in the buckets. To overcome the limitations, a partition based approach is proposed in this paper. The proposed approach, PSSFIM, takes single scan of the database to identify frequent itemsets. The unique feature of PSSFIM allow to generate size of candidate itemsets independent on the minimum support. It allows the candidates in hash that are possible for frequent, which intuitively reduces the cost in terms of verifying the support of generated candidates. It is compared with SS-FIM and Apriori with the standard datasets. The results show that the PSSFIM is good at the comparison of SS-FIM and Apriori.

Keywords* : *Apriori, Frequent Itemsets, Run time, Support.

I. INTRODUCTION

Frequent Itemset Mining (FIM) is a popular data mining technique in extracting hidden and important itemsets from a given database. Let TDB be a Transactional Data Base recorded with a set of tuples [1] {T1, T2, T3, …., Tm}, each transaction is recorded with a set of distinct {I1, I2, I3, …., In}. An itemset Y is set of items where Y≤I. The support count of an itemset is computed as the occurrence of itemset Y in the transactions. And the actual support is from its occurrence and total number of transactions in the database. An itemset Y is frequent if its support is higher or equal than the given user threshold value. So, the aim of FIM is to derive frequent itemsets from the TDB.

Investigation of FIM is started with two categories of approaches. In the first category, they generate all possible itemsets length K at K-levels and then test their occurrence in the TDB to determine frequent itemsets. They use Apriori heuristic [1] to reduce unnecessary candidate itemsets. The second category approaches relies on FP Growth heuristic [2]. They maintain compressed tree format of TDB in main memory. And visit in recursive manner to find frequent itemsets. Although this category avoids candidate generation, these approaches consumes more memory when dealing with large databases with less minimum support.

Several approaches have been carried out for deriving frequent itemsets on the basis of the above two approaches. Although they are effective in less candidate generation and passes over the database. They consume more storage space and execution time in-terms. Hence it is motivated researchers to propose single scan approaches for deriving frequent itemsets.

One of the approaches is SS-FIM proposed by Youcef et al. [17]. They generate all the candidates for each transaction and then stores them into hash table to maintain support of each itemset. While generating each itemset, they check itemset presence in the hash, then inserts into hash if it is not already stored in the hash table. Otherwise it increments the support of itemset. After the candidate generation, itemsets hash table are examined to decide frequent itemsets. Although it is single scan, it consumes huge storage space when the transaction length is high.

This paper propose a Novel approach called Partition Based Single Scan Approach (PSS-FIM) for mining frequent itemsets. It is aimed at the improvement of SS-FIM with lesser number of candidates. Here, initially, all the transactions are divided into equal partitions. Candidate itemsets are generated from each transaction and maintained in hash table, if it is already not hashed, otherwise counter incremented by one.

When half of the partitions are visited, the candidates are tested before storing them into the hash table. At the end, the cumulated support count of each itemset in the hash table is considered to determine frequent itemsets. The contribution of PSS-FIM is to avoid unnecessary candidate itemsets which are going to be infrequent. The PSS-FIM approach has been investigated on various datasets against the Apriori, SS-FIM in result section. The results show that PSS-FIM outperforms other approaches when the minimum support is low.

The organization of the paper is presented as follows. The very next section reviews the FIM algorithms. In the next section, Apriori and SS-FIM are presented and followed by PSS-FIM algorithm. The result analysis of the PSS-FIM is presented in next section. Conclusion is presented in end section.

Revised Manuscript Received on October 30, 2019.
* Correspondence Author
U. Mohan Srinivas*, Research Scholar, Department of Computer Science & Engineering, Acharya Nagarjuna University, College of Engineering, Guntur, India.
E. Srinivasa Reddy, Department of Computer Science & Engineering, Acharya Nagarjuna University, College of Engineering, Guntur, India.

II. RELATED WORK

Frequent Itemset Mining is one of the traditional data mining technique. One of the reasons for FIM is to derive hidden correlation between itemsets in a strategic manner to avoid the difficulties of candidate generate and test strategy, where all the candidates are generated and then tested. To overcome these issues, level-wise candidate generates and test approach is proposed, where candidates are generated and tested at level wise (K-size) and k-length candidate itemsets are generated from (k-1) size frequent itemsets. Hence some of the unnecessary computation is not done. However, too many candidates are needed to test with many passes.

In line to the above issues Agrawal et al. [1] proposed Apriori concept which is a level wise candidate generate and test approach. It follows the property stating that if any subsets of frequent itemsets must be frequent. Hence the candidate itemsets are not considered whose subsets are not frequent. In addition to that one more property is included, Anti-monotonic property states that an itemset is not frequent then all of its supersets are not frequent. Though the above properties effective in-terms of unnecessary candidates, it is not efficient in-terms of passes. Many techniques are based on Apriori.

Table 1 shows the transaction database, which is recorded with 8 transactions and items {a, b, c, d, and e}. Illustration of Apriori is presented in Figure 1, where the list of candidates itemsets and frequent itemsets are visualized at various levels with respect to the minimum support threshold 50% (count=4).

Candidate itemset length 1 is tested against the TDB to cumulative the support to compare with the minimum support and it is visualized in Figure 1. And also, candidate length-2 and 3 are generated from frequent length-1 and length-2 by joining C1 with C1 and C2 with C2 when they share common k-1 length. Since no possibility of C3 candidates, it is terminated. After the step, frequent itemsets, when minimum support is 4 = {{a}, {b}, {c}, {a b}, {b c}}.

Table 1: Illustration of Transactional Database

TID	Purchased Items
1	a,b,e
2	b,d
3	b,c
4	a,b,d
5	b,c
6	a,c
7	a,b,c,e
8	a,b,c

Limitations of Apriori is motivated researchers to enhance the approach, which are (a) too many data base scans (b) too many candidates when the minimum threshold is low. One of the approaches is DIC (Dynamic Itemset Counting) proposed by Brinet et al. [3]. Here, TDB is divided into equal parts, classified them as strong, suspected small and large in first partition. In subsequent partitions to validate candidate itemsets and pushed into the candidate itemsets from suspected to large. Though it is efficient it is suitable when the TDB is fitting into the main memory.

Zaki et al. proposed variation of FIM is Eclat [5]. The database is represented as vertical list of transactions tidlist. It is based on disjoint equivalence classes, where itemsets are categorized into subsets based on its common k-1 prefixes. Next level candidate itemsets K-lenght are generated from its previous length (k-1) equivalence classes. The support of candidate itemsets are determined from the intersection of tidlist. And also proposed [6] another approach which is a variant of Eclat. It was able to derive regular patterns. Another variation of Apriori is proposed by Muller et al. [4]. It is a tree kind of approach which is depends on prefix structure. Thus allows faster performance. However, this kind of structure usually requires large space for both candidates and frequent itemsets.

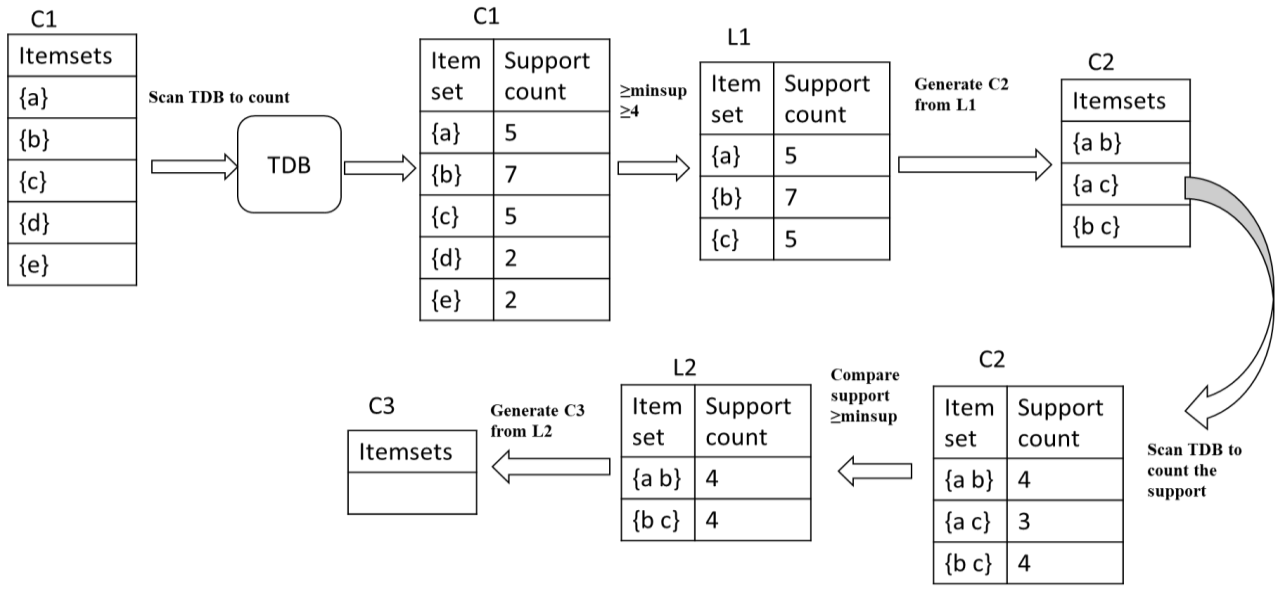

Figure 1: Apriori Illustration of Table 1

To avoid the limitations of Apriori based approaches, FP-Growth kind of approaches are proposed, where the database is represented in compressed tree FP. It is composed into phases. The first phase- FP Tree construction, where each transaction is inserted as a path. The second phase, frequent itemsets identification- frequent itemsets are derived by visiting each node in a recursive manner. Due to the structure, they are fast in response. However, it is limited to the small database. To improve FP-growth, Cerf et al. [8] proposed NFPgrowth algorithm. Frequent patterns are maintained in independent table to speed up mining process. To handle uncertain databases, modified FP-growth [7] was proposed. In further reducing the tree traversing time, variant FP array techniques [9,10] were proposed. It was extended to derive maximal, closed and categorical itemsets.

The above approaches show their abilities when the input database size is moderate. They need to alter for the large databases whose size starts from millions [11]. The result of the above investigation, to reduce the database competency, bio inspired approaches BSO-ARM [12], PeARM [14], and PGARM [13] proposed. However, they are not suitable for FI's extensions.

To overcome the difficulties of FIM, Youcef et al. [17] proposed SS-FIM (Single Scan Frequent Itemset Mining) approach, where all the possible itemsets are considered as candidates that are obtained from each transaction and stored in hash table. For a new itemset, if an instance is in table, then its counter is incremented. Otherwise a new entry is created with count one. At the end, counts of all entries are compared against minimum support and determines a frequent itemsets. The above procedure is done with a only one scan on the given database. However, too many candidates are generated when the minimum support is low and dense databases. SS-FIM approach is illustrated for the table 1 is presented in Figure 2.

Limitations: Similar to Apriori, it is very sensitive to minimum support. It generates huge itemsets when it is varying from high to low.

III. PARTITION BASED SINGLE SCAN FREQUENT ITEMSET MINING (PSS-FIM)

This section discusses the proposed approach PSS-FIM, which is an extension of SS-FIM. Then theoretical analysis and result analysis w.r.t Apriori and SS-FIM.

The purpose of PSS-FIM is to reduce search space size while visiting transactions. It relies on the divide and mine approach, divide the transactions into equal partitions M where |M| is (minsup*|TDB|/2), and generates all the candidates for each transaction. It increments the support of itemset in hash table, If it is already indexed. Otherwise, it creates a new entry with the itemset name in hash table initializes counter value with one. This process is repeated for all the partitions. One of the goal is to avoid the itemsets which are not going to become frequent. After visiting half of the partitions, if I is newly generated itemset and not indexed in the hash table then it is discarded, because of the partition concept, it is not going to be frequent. Such kind of itemsets are not stored in hash table. Hence the computation for such kind of unnecessary itemsets is achieved with no information loss.

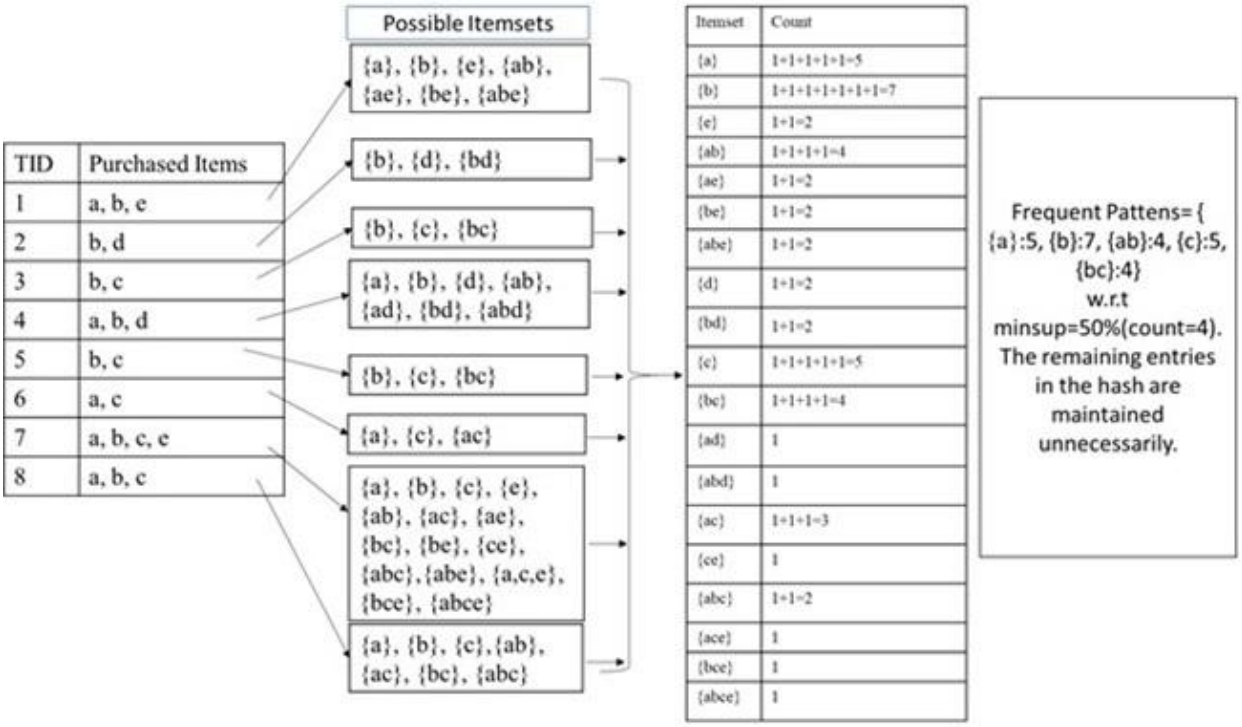

Figure 2: SS-FIM illustration of Table 1

PSS-FIM algorithm is complete, because it derives FI's directly from the candidate itemsets which are generated directly from the given TDB, whose support is ≥minsup. After visiting minsup of TDB, if the newly generated itemset is not indexed in the hash table, there is no chance of getting the frequency of minsup. Hence it is complete. Algorithm 1 describes the step by step activities of PSS-FIM.

Algorithm 1: PSS-FIM Algorithm

Algorithm: PSS-FIM Algorithm
Input: TDB Transactional DataBase, minsup: user minimum threshold.
Output: FI: set of Frequent Itemsets.

Partition the transactions of TDB into M equal partitions
M←$\frac{|TDB|\times(\frac{minsup}{2})}{100}$
for each $M_k \in$ M
  for each transaction Ti ∈ TDB do
    Cand ← PossibleItemsets(Ti)
      for each Itemset I ∈ Cand do
        if I ∈ h then
          H(t) ← h(t) +1
            else { if i > ($\frac{|TDB|}{2}$) h(t) ← 1} *// after visiting half of the partitions.*
        end if
      end for
  end for
 end for
FI ← ∅
for itemset I ∈ h do
if h(t) ≥ minsup then
 FI ← FI U I
end if
end for
return FI

PSS-FIM takes transactional database TDB as input, and minsup given by the user. It divides TDB into M partitions which are equally size shown as first statement. Size of each partition is the half of minsup of TDB. Also uses data structure hash table to maintain the candidate itemsets along with support value as occurrence in step 8 and 9. After visiting half of the partitions, the generated itemset is new to the hash table then it is discarded in step 9. At the end, it returns all the itemsets whose support value is ≥ minsup.

First, set of possible itemsets C is generated from each transaction Ti. For instance, consider transaction T1 {a,b,c}, then C={ { a }, { b}, {e}, {a b}, {ae}, {be}, {abe}}. After that, each itemset I ∈ C is stored in hash table h. Initially, h is empty, then itemsets are stored with count value 1, where index name is same as item name. After visiting all the transactions, supports of all the keys are compared with minsup to determine frequent itemsets.

Illustration: Figure 3 shows the pictorial representation of PSS-FIM over TDB of Table 1 for minsup=50% (0.5 or 4). It starts by partitioning TDB into 4 partitions. And then it starts by reading the transaction T1 {a, b, e} and extracts the possible itemsets are {a}, {b}, {e}, {ab}, {ae}, {be} and {abe}. The hash table h is empty, then all these itemsets are stored into the h and support is initialized to 1. It continues till the half of the partitions are visited, and then for new itemsets, if it is not indexed in h, then itemsets are discarded. For example, T6 of P3 is {a, c}, then C= {a, c, ac}. Since the availability of {a} and {c}, its counts are cumulated by 1. Itemset {ac} is discarded, since no entry is in h and not possible to become frequent.

*Theoretical Analysis: T*ime complexity of PSS-FIM is determined from the cost of (i) Possible Itemset generation (ii) Frequent Itemset identification. The cost for candidate generation of each transaction Ti is $2^{|Ti|}$-1, and for TDB is $\sum_{i=1}^{N} 2^{Ti} - 1$, where N is |TDB|. Partition heuristic of PSS-FIM discards some itemsets which are not frequent, denoted as I_X. Hence it is $\sum_{i=1}^{N}(2^{Ti} - 1 - I_X)$. the complexity of possible itemset generation is

$$O(N(2^p - 1 - I_X)), \quad \text{-- (1)}$$

Where p is possible items (maximum) for each transaction.

The second one, to find FI's, hash table need to look at each index for compute support, then operation is $N(2^p - 1 - I_X)$.

The total running cost of PSS-FIM is

$$O(2N(2^p - 1 - I_X)), \quad \text{-- (2)}$$

Whereas the Apriori complexity is $O(N \times M^2)$, and SS-FIM is

$$O(N(2^p - 1)). \quad \text{-- (3)}$$

Although Equation 1 and 3 has exponential form, while equation 1 is polynomial, 1 and 3 yields lower values compared to 2 for most TDB's, because p is usually lesser than N. Equation 1 outperforms Equation 3, because of I_X.

IV. RESULT ANALYSIS

To present the performance details of PS-FIM, experiments are conducted on datasets [16] on the system with the configuration of 4GB RAM and Intel P3.

One of the dataset is Bolts, It is represented with 2178 transactions. Each transaction is comprised of items 8-16. Other dataset where size is medium, comprised with transactions of 59000-100000. And each one is comprised of items 500-16000 with on average 2-10. The other kind of dataset is large database BMS POS. It has used 1660 items on average 2.5 in 500000 transactions.

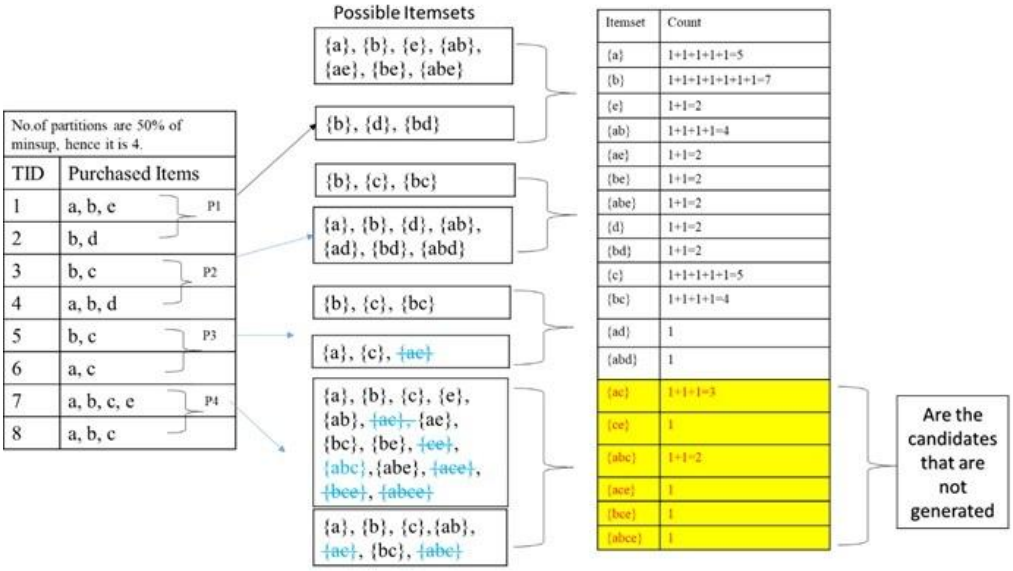

Figure 3: PSS-FIM illustration of Table 1

Table 2 is recorded with runtime comparison of PSS-FIM, SS-FIM and Apriori. It says that Apriori has better performance than SS-FIM and PSS-FIM. However, PSS-FIM exhibits double performance than others when the datasets are large and medium. Table 2 is reported that PSS-FIM shows good performance for dense databases.

Experimental result on BMS-POS dataset is presented in Figure 4. It shows that PSS-FIM takes less execution time compared to the other approaches while minsup decreases.

Table 2. Execution Time of PSS-FIM, SS-FIM and Apriori

Dataset	Apriori-Maximal	SS-FIM	PSS-FIM
Bolts	4	140	110
Sleep	6	110	80
Pollution	20	821	641
Basket ball	15	18	17
Quake	29	50	45
BMS-WbView-1	1002	45	34
BMS-WbView-2	3985	80	65
retail	4895	525	400
Connect	2600	1285	1021
BMS POS	9825	500	411

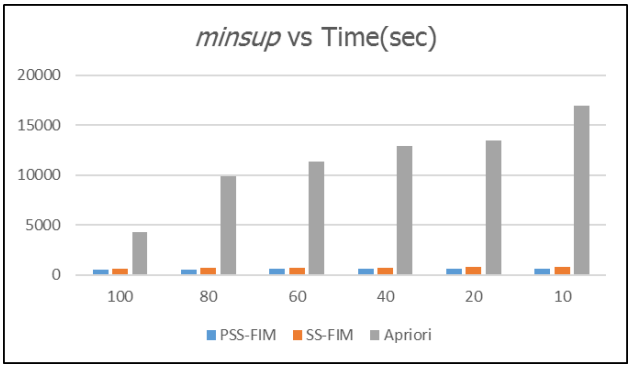

Figure 4: Runtime (sec) of PSS-FIM, SS-FIM and Apriori w.r.t minsup for BMS POS

V. CONCLUSION

This paper has proposed an intelligent FIM algorithm. It consumes less candidate itemsets and one scan to extract all frequent itemsets compared to Apriori and SS-FIM. Hash table is adopted to store the candidates that are generated in each partition unless it is not violated partition heuristic.

Theoretical and experimental results say that PSS-FIM performance is better than other approaches for large and dense databases. Results motivated us to look at the more heuristics so that performance can be improved and we plan to extend PSS-FIM to determine maximal and closed itemsets.

REFERENCES

1. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In proceedings of the ACM SIGMOD Record, vol. 22, no. 2, pp. 207–216. ACM, June 1993
2. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGMOD Record, vol. 29, no. 2, pp. 1–12. ACM, May 2000
3. Brin, S., Motwani, R., Ullman, J.D., Tsur, S.: Dynamic itemset counting and implication rules for market basket data. In: ACM SIGMOD Record, vol. 26, no. 2, pp. 255–264. ACM, June 1997
4. Mueller, A.: Fast sequential and parallel algorithms for association rule mining: a comparison. Technical report CS-TR-3515, University of Maryland, College Park, August 1995.
5. Zaki, M.J., Parthasarathy, S., Ogihara, M., Li, W.: New algorithms for fast discovery of association rules. In: Third International Conference Knowledge Discovery and Data Mining (1997).
6. Amphawan, K., Lenca, P., Surarerks, A.: Efficient mining top-k regular-frequent itemset using compressed tidsets. In: Cao, L., Huang, J.Z., Bailey, J., Koh, Y.S., Luo, J. (eds.) PAKDD 2011. LNCS (LNAI), vol. 7104, pp. 124–135. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28320-8 11.
7. Leung, C.K.-S., Mateo, M.A.F., Brajczuk, D.A.: A tree-based approach for frequent pattern mining from uncertain data. In Proc.PAKDD 2008. pp. 653–661. doi:10.1007/978-3-540-68125-0 61.
8. Cerf, L., Besson, J., Robardet, C., Boulicaut, J.F.: Closed patterns meet n-ary relations. ACM Transactions on Knowledge Discovery Data (TKDD), vol. 3 issue 1, Mar 2009.
9. Grahne, G., Zhu, J.: Fast algorithms for frequent itemset mining using FP-trees. IEEE Transactions on Knowledge and Data Engineering. Vol.17-issue 10, pp. 1347–1362, 2005. Doi: 10.1109/TKDE.2005.166
10. Borgelt, C.: Frequent item set mining. Wiley Interdisc. Rev.: Data Mining and Knowledge Discovery. 2(6), 437–456 (2012).
11. Djenouri, Y., Bendjoudi, A., Mehdi, M., Nouali-Taboudjemat, N., Habbas, Z.: GPU-based bees swarm optimization for association rules mining. The Journal of Supercomputing. 71(4), 1318–1344 (2015).
12. Djenouri, Y., Drias, H., Habbas, Z.: Bees swarm optimisation using multiple strategies for association rule mining. International Journal of Bio-Inspired Computation. Vol. 6, issue 4, pp. 239–249, sep-2014.
13. Gheraibia, Y., Moussaoui, A., Djenouri, Y., Kabir, S., Yin, P.Y.: Penguins search optimisation algorithm for association rules mining. Journal of Computing and Information Technology. Vol.24, issue 2, pp.165–179, 2016. doi: 10.20532/cit.2016.1002745
14. Luna, J.M., Pechenizkiy, M., Ventura, S.: Mining exceptional relationships with grammar-guided genetic programming. Knowledge and Information Systems. Vol. 47, issue 3, pp.571–594, 2016.
15. Hegland, M.: The Apriori Algorithm – A Tutorial. Mathematics and Computation in Imaging Science and Information Processing. Vol. 11, pp.209–262.2005. Doi:https://doi.org/10.1142/9789812709066_0006
16. Guvenir, H.A., Uysal, I.: Bilkent University function approximation repository (2000). http://funapp.cs.bilkent.edu.tr/DataSets.
17. Youcef D, Marco C, Djamel: SS-FIM: single scan for frequent itemsets mining in transactional databases. Pacific-Asia Conference on Knowledge Discovery and Data Mining, part II, LNAI 10235, pp. 644-654, 2017.

AUTHORS PROFILE

U. Mohan Srinivas received the B.Tech. Engineering Degree from Acharya Nagarjuna University, Guntur, India in 1991, M.Tech. in Computer Science and Engineering from Jawaharlal Nehru Technological University, Kakinada, India in 2004. He is pursuing Ph.D. from CSE Dept., ANUCET at Acharya Nagarjuna University under the guidance of Prof. E. Sreenivas Reddy. He is a member of CSI and IAENG. His research interests include Data Mining, Artificial Intelligence and Pattern recognition.

E. Sreenivas Reddy got B.Tech (ECE) Degree from ANU, Guntur, India in 1988, M.S. Degree from BITS, India in 1997, M.Tech (CS) from Visveswaraiah Technological University, India in 2000 and Ph.D in Computer Science from Acharya Nagarjuna Univeristy, India in 2008. Currently he is guiding many scholars pursuing Ph.D. from different universities. He is the senior member of IEEE and presented many papers in international conferences and published papers in several national and international journals. His research interest includes Data Mining, Image Processing, Biometrics and Pattern recognition.