A genetic algorithm coupled with tree-based pruning for mining closed association rules

Jashma Suresh Ponmudiyan Poovan[1], Dinesh Acharya Udupi[1], Nandanavana Veerappareddy Subba Reddy[2]
[1]Department of Computer Science and Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education, Manipal, Karnataka, India
[2]Department of Information Technology, Manipal Institute of Technology Bengaluru, Manipal Academy of Higher Education, Bengaluru, Karnataka, India

Article Info

Article history:

Received Jun 23, 2022
Revised Aug 1, 2022
Accepted Aug 18, 2022

Keywords:

Closed itemsets
Crossover
Mutation
Offspring
Genetic algorithm

ABSTRACT

Due to the voluminous amount of itemsets that are generated, the association rules extracted from these itemsets contain redundancy, and designing an effective approach to address this issue is of paramount importance. Although multiple algorithms were proposed in recent years for mining closed association rules most of them underperform in terms of run time or memory. Another issue that remains challenging is the nature of the dataset. While some of the existing algorithms perform well on dense datasets others perform well on sparse datasets. This paper aims to handle these drawbacks by using a genetic algorithm for mining closed association rules. Recent studies have shown that genetic algorithms perform better than conventional algorithms due to their bitwise operations of crossover and mutation. Bitwise operations are predominantly faster than conventional approaches and bits consume lesser memory thereby improving the overall performance of the algorithm. To address the redundancy in the mined association rules a tree-based pruning algorithm has been designed here. This works on the principle of minimal antecedent and maximal consequent. Experiments have shown that the proposed approach works well on both dense and sparse datasets while surpassing existing techniques with regard to run time and memory.

Corresponding Author:

Dinesh Acharya Udupi
Department of Computer Science Engineering, Manipal Institute of Technology, Manipal Academy of Higher Education
Manipal 576104, Karnataka, India
Email: dinesh.acharya@manipal.edu

1. INTRODUCTION

Frequent itemset involves extracting itemsets that frequently occur in a dataset. Different techniques were proposed in recent years for extracting such itemsets but limited efforts were made to address the redundancy of itemsets. This paper emphasizes addressing redundancy by extracting closed itemsets. Most of the algorithms designed in this regard are either based on tree, list, or lattice-based approaches. While some of these algorithms underperform in terms of run time, others underperform in terms of memory. There is thus a dire need to devise an approach that handles these drawbacks. Recent studies have shown that genetic algorithms have better performance than conventional algorithms for mining frequent item sets. The operations such as crossover and mutation that rely on bitwise operations are found to enhance the efficacy of the extraction of itemsets. Once the itemsets are extracted redundancy continues to prevail in the association rules that are mined. Removing this redundancy will help in improving the conciseness of the rules that are mined. A tree-based pruning approach has been designed here that uses the concept of minimal antecedent and maximal consequent. This helped in producing a concise and lossless set of association rules.

The main contributions of the proposed work are as follows: i) The redundancy of itemsets is addressed by mining closed itemsets. A genetic algorithm is proposed to mine these itemsets; ii) Itemsets are represented in bitwise form and the rank selection technique is used to select the population of itemsets. Itemsets are sorted according to the fitness of the itemsets and a rank is assigned to them; iii) To find a k-itemset a cross-over point of 'k' is set. Progenies are generated by interchanging the bits of the parents until the crossover point is attained. These are then added to the population. Some genes of the offspring obtained can undergo mutation with a low random probability. This is achieved by flipping a few of the bits in the bit string of the itemset; iv) A closure checking operation is carried out to generate a set of itemsets $CFI=X \mid \neg\exists\, Y$ *wherein* $Y \supsetneq X$ *and support(X)=support (Y)*; v) A hash-based approach is used to generate the resultant collection of closed frequent itemsets; and vi) Association rules are mined from the extracted itemsets. Redundancy amongst these rules is addressed by designing a tree-based algorithm that uses the concept of minimal antecedent and maximal consequent. Section 1 represents the Introduction. The background, problem statement, and proposed solution have been explained here. Section 2 describes the proposed methodology, section 3 shows results and discussion, and finally, section 4 represents the conclusion part and future work.

To avoid repeated scanning of databases an efficient tree data structure called improved single scan pattern tree (ISSP)-tree [1] was designed to retain all database transactions regardless of the frequencies of the itemsets. This method was also adaptive to incremental and interactive mining. Uncertain frequent patterns were proposed by employing the concept of upper bound [2]. This resulted in reducing the number of false positives. Recurrent itemsets were identified using a combination of unstructured data and time series machine learning [3]. An incremental utility-based pattern mining algorithm was put forth here. To extract data in big datasets a Binary based Technique was designed [4]. Threads were collaborated to generate frequent itemsets in a big data environment. A compressed fuzzy list structure was designed to extract frequent itemsets [5]. This helped in avoiding repeated screening of the dataset.

Due to the relevance of mining frequent itemsets in varied domains, a voluminous amount of itemsets are generated in most cases. However, redundancy among these itemsets continues to be a pressing issue. Several approaches have been adopted in recent years to address this. Some of the commonly used approaches involve mining closed itemsets. These itemsets are concise and lossless. This means that these representations result in a concise collection of itemsets and the remaining itemsets may be generated from the existing ones by using standard rules of deduction. This involved construction of a CFI-Tree to check for supersets. An algorithm that was based on reflexivity and transitivity was developed by Li *et al.* [6]. A topology transaction tree was constructed to represent the associative relationship among the items of a transaction. A frequent pattern tree is constructed to extract frequent item sets in an incremental manner [7]. A multiscale approach has been designed for this purpose. This avoids the need to repeatedly scan the database and adjust the tree during the mining process. Yamamoto *et al.* [8] put forth a technique that is a fusion of parameter and resource-constrained mining of closed itemsets. While parameter constrained mining controls the maximum error rate, the resource-constrained approach has the advantage of setting a maximum memory consumption for the resources. A technique that focuses on mining local itemsets based on support size and cost, was presented to extract closed itemsets [9]. A hybrid approach was framed for mining frequent, sequence, and graph mining that makes use of ASP to generate frequent patterns. To minimize the search area a combination of dynamic bit vector and dynamic superset bit vector was designed [10].

Recent studies have shown that the extrication of closed sequences is more optimal than producing the entire collection of frequent items. An algorithm called CLoFS-DBV was put forward by Tran *et al.* [11] that relied on the usage of dynamic bit vectors. This resulted in the generation of a concise representation thereby minimizing the runtime and memory used in the process. This structure was further enhanced to extract a weighted closed itemset [12]. An early pruning strategy is applied here to eliminate unsatisfactory candidates.

To mine closed frequent sequences, Fabio *et al.* [13] put forth a CloFAST algorithm. The support of the itemsets is counted by merging the id's of sparse and vertical lists. This is followed by the extraction of new sequences in the next step. However, this approach has not taken care to remove noise from the datasets. To prune the search space a tree-based algorithm was presented by Rodríguez-González *et al.* [14]. The proposed closed frequent similar pattern algorithm (CFSP)-miner algorithm was found to perform better than the state-of-the-art approaches.

A study on the association between closed and generator itemsets was presented by Ledmi *et al.* [15]. A generator-based algorithm called GrAFCI was designed here, and its efficacy was investigated in terms of scalability and interestingness. Experiments showed that the proposed algorithm outperformed contemporary algorithms in terms of run time. Closed itemsets were extracted from a data stream using a compressed SlideTree data structure [16]. The method proposed here removes tuples that get updated from the SlideTree thereby avoiding the need to visit the entire previous slide.

To handle the top rank k-closed frequent pattern mining problem Abed *et al.* [17] presented two approaches. The first was based on Boolean satisfiability where an efficient algorithm was designed to improve the encoding of the problem. The second approach was based on constraint programming where a constraint programming model is exploited to mine top rank-k-closed frequent patterns from transactional datasets. Frequent itemsets were extracted by using a combination of set theory and class equivalence [18]. Each class is an independent group of itemsets. The support of individual itemsets is calculated by using Diffsets. Three algorithms namely, fast mining frequent itemsets using Nodesets (FIN), PrePost, and PrePost+ have been customized [19] to extract top rank-k frequent patterns. A dynamic minimum support strategy has been applied to enhance its efficiency.

Redundancy in the itemsets has been handled by designing an algorithm ExCover where each pattern covers at least one positive transaction [20]. Best covering patterns are extracted using a branch and bound technique. Recent studies have shown that genetic algorithms can increase the overall performance of extracting itemsets. Bagui and Stanley [21] proposed an approach for mining frequent item sets from a streaming environment by combining the techniques of concept drift and genetic algorithm. The genes are represented in binary form. The fitness of these genes is calculated and accordingly a cross-over or mutation is carried out to find the best fit individuals. Toman *et al.* [22] proposed a technique that used this algorithm along with k-means clustering for effective retrieval of information from datasets. A framework based on a bio-inspired framework was proposed [23] for mining frequent itemsets. Two new approaches were put forward in this paper-one based on the genetic algorithm and the other based on particle swarm optimization (PSO). To address the redundancy of itemsets Maulik *et al.* [24] proposed a computational rule mining framework. The first step involves applying a statistical strategy to eliminate redundant genes by selecting only those that satisfy the data distribution property. This is followed by discretization and post-discretization of the data consecutively. Rao *et al.* [25] optimized the traditional Apriori algorithm using the concepts of genetic algorithm for mining frequent itemsets. The proposed approach reduces the scanning time and also reduces the number of candidate itemsets at each step.

Association rules that were mined also had redundancy amongst them. To handle this problem a genetic algorithm [26] was put forth. The advantage of using genetic algorithms is that they help in reducing the time complexity of the algorithm while extracting frequent items [27]. Association rules were mined using a single scan of the database. In addition, this was also found to increase the security and confidentiality of itemsets [28]. An effective evaluation function is made use of to handle the drawbacks of missing costs and failures. Derouiche *et al.* [29] presented an improved genetic algorithm for extracting non-redundant association rules. Duplicate and redundant rules have been pruned by using a filter rules procedure. A lattice-based structure was employed to mine minimal non-redundant association rules [30]. To mine temporal association rules from a multi-attribute graph sequence, an algorithm has been proposed by Du and Yu [31]. This involves finding temporal association rules and then finding the credibility of these rules.

A scalable association rule learning heuristic was put forth by Li and Sheu [32] to learn the gene-disease association rules from large-scale microarray datasets. The proposed approach uses the divide and conquers approach along with graph theory to enhance the process of association rule learning. The best association rules were extracted using the *Élimination Et Choix Traduisant La Realité* (Elimination and Choice Translating Reality/ELECTRE) method [33]. A modified frequent pattern (FP)-growth algorithm is put forth by Shawkat *et al.* [34] to overcome the performance gap while handling large datasets. The relationship between skill and job was analyzed using this algorithm [35]. FP-growth and Apriori algorithms were employed to find the relation between weather data variables [36]. Hidden patterns were evaluated and related rules were extracted here [37]. A combination of Apriori, rough set, and fuzzy techniques were employed to detect fraudulent credit card transactions [38]. To extract association rules from sparse datasets Mouakher *et al.* [39] presented a technique that relied on the extraction of literal sets. An algorithm FasterIE has been designed that minimizes the number of nodes visited in the search space thereby improving the efficiency of mining itemsets. Vives-Mestres *et al.* [40] proposed a technique based on compositional data analysis to produce novel insights from association rules. The relationship between itemsets, dependency, strength, and direction has been analyzed here. To facilitate incremental mining i-Eclat was put forth by Bakar *et al.* [41]. Thi *et al.* [42] proposed an approach for constructing a classification model that employed multi-objective optimization and association rule mining techniques. The rules are selected based on interestingness measures and multi-objective optimization. Table 1 shows a comparison of a few of the existing approaches highlighting the advantages and disadvantages of each of them.

The disadvantages of existing approaches are that most of the existing techniques for mining association rules involve building and spanning a tree data structure which increases the time consumed by the algorithm. The itemsets and rules extracted using the traditional techniques also had redundancy and infrequent itemsets in them. The use of other meta-heuristic algorithms like PSO was found to have a low convergence rate which brought down the efficiency of the algorithm.

Table 1. Comparison of the existing approaches

Reference	Methodology and Advantages	Drawbacks
HPGA [22]	- Uses a k-means cluster with two levels of genetic algorithm for information retrieval - Obtained higher precision and better-quality information retrieval - There was a reduction in irrelevant results	Using multiple techniques can add to the time complexity of the algorithm Outliers may sometimes become a separate cluster instead of being ignored
MCDA-ELECTRE [33]	- Apply the ELECTRE method to find the best association rule - Measures are taken as attributes and association rules as alternatives and a decision matrix is constructed - Desired rules are obtained with maximum interestingness	Redundancy of the rules is not addressed
FP-Growth [35]	- Uses Association Rule Mining to analyze skill-job relationships - Jobs are analyzed using the FP-Growth algorithm - able to retrieve and process large amounts of data posted online	- Construction of FP-tree is difficult - The memory may be insufficient if the database is large
TAR [36]	- Uses Apriori and FP-growth to generate itemsets - Generates association rules to find the relation between weather data variables accompanied by time scale parameters.	- Apriori algorithm generates a large number of candidates and scans the database repeatedly thereby slowing down the process of generating itemsets - FP-Tree construction is difficult and consumes more memory.
EPDA [37]	Evaluate hidden patterns and produce related rules from the datasets centralize all algorithm execution in the mappers/nodes and integrate all effective Apriori algorithms.	Generates a large number of candidates. Lacks organization in the interaction between mapper and nodes
CCFD [38]	- Uses a combination of Apriori, rough set, and fuzzy techniques to detect fraudulent credit card transactions - Generates association rules to improve the detection of fraud in credit card transactions - Has the advantage of being able to work in real-time - Shows considerable reduction in run time	- Apriori algorithm uses a lot of resources thereby reducing the efficiency of the framework - Fuzzy sets may produce inaccurate results
i-Eclat [41]	- Proposed an incremental version of the Eclat Algorithm - Uses MySQL database to reduce the preprocessing stage of the database.	- Infrequent patterns are not handled effectively
IPOC [43]	- Implements an efficient IPOC. A frequent item set generation algorithm was applied to this tree. - New data items were linked to previous nodes in the tree by increasing their support count	Construction and traversal of the tree are time-consuming
FACO [44]	- The ant colony is used to calculate the shortest cycle - The path is recognized using the quickest shortest cycle. - Outperformed traditional ant colony systems	Performs well mostly for discrete problems
PSO [45]	- Software agents called particles move in the search space of an optimized problem - Has higher throughput	Has a low convergence rate
Alternative_3 [46]	- Uses a combination of trie and multi-thread approach to generate frequent itemsets. - The algorithm that uses multi-thread performs better than the algorithm implemented in single-thread in calculating hash-node	the processing time needed is longer if the experiment uses real data

Abbrev.: Hierarchical parallel genetic algorithms approach (HPGA), multi-criteria decision analysis-ELECTRE (MCDA-ELECTRE), frequent pattern Growth (FP-Growth), temporal association rule (TAR), enhanced parallel and distributed apriori (EPDA), credit card fraud detection (CCFD), incremental pre-ordered coded tree (IPOC), and fuzzy logic ant colony optimization (FACO)

From the literature, it can be concluded that although several techniques were put forth in recent years for mining frequent Itemsets, limited efforts were made to address the redundancy in the itemsets. This paper plans to address redundancy by mining closed itemsets. However, the association rules extracted from these itemsets also contain redundancy and designing an effective approach to address this issue is of paramount importance. Although multiple algorithms were proposed in recent years for extracting closed itemsets and their association rules most of them underperform in terms of run time or memory. Another issue that remains challenging is the nature of the dataset. While some of the existing algorithms perform well on dense datasets others perform well on sparse datasets. There have been very few efforts made to design an algorithm that works well on both datasets. Retrieving the resultant collection of itemsets in minimal time is also of pressing concern. Conventional approaches mostly use an Apriori-based approach or a trie data structure. Both of these consume more time thereby bringing down the performance of the algorithm. Thus, there is a requirement to design an approach that can handle all these drawbacks.

This paper aims to handle the above-mentioned drawbacks by using a genetic algorithm for mining closed itemsets. Recent studies have shown that genetic algorithms perform better than conventional algorithms due to their operations such as crossover and mutation that rely on bits. Bitwise operations are predominantly faster than conventional approaches and bits consume lesser memory thereby improving the overall performance of the algorithm. In addition to retrieving the resultant collection of closed itemsets a hash-based approach has been employed here. This brings down the retrieval time to O (log n) consequently improving the run time of the algorithm. To address the redundancy in the mined association rules a tree-based pruning algorithm has been designed here. This works on the principle of minimal antecedent and maximal consequent. Experiments have shown that the proposed approach works well on both dense and sparse datasets while surpassing existing techniques with regard to run time and memory.

There are four advantages of using a genetic algorithm. First, the genetic algorithm allows a deeper mechanism of understanding the fitness of individual items by using a multi-objective function. A multi-objective function is designed: Maximize $F(X) = max\{(supp(X), Closure(X))\}$. This allows for finding interesting solutions in a single run. Second, an exhaustive search is performed of the itemsets during the population initialization phase. Third, genetic algorithms are very resilient because they can be used to handle a variety of item-set mining problems, such as distributed learning approaches, dynamic databases, data streams, negative association rules, and multi-objective approaches. Fourth, the genetic algorithm uses operations such as crossover and mutation that uses bits. Most commonly used approaches either rely on traversal of tree or list structure to mine frequent itemsets. In comparison to traversal-based approaches, crossover and mutation are faster since these rely on bits and bitwise operations are faster.

There are also four advantages of employing hashing. First, hash tables have the advantage of speed. Since the proposed approach uses hashing for retrieving the resultant collection of itemsets, the time complexity is brought down to $O(log\ n)$ which considerably improvises the efficiency of extracting closed itemsets. Therefore, accessing itemsets becomes faster. Second, commonly used existing approaches insert the candidates into a frequent itemset tree (also called a set enumeration tree). Frequent Itemsets are mined by traversing this tree. To eliminate infrequent itemsets a pruning technique is employed while traversing the tree. While comparing this with the proposed Hash-based approach, the proposed approach is better in terms of run time because the time required for search operation through hashing is typically $O(1)$ and $O(x)$ in the worst case. On the other hand, the time required for traversing a set enumeration tree is $O(x + y)$ where *x* is the number of nodes and y is the number of edges in the worst case. Third, the itemsets that satisfy the multi-objective functions are mapped into a Hash table while the rest are not considered. This ensures that the hash table generated is more compact consequently improving memory consumption as well. Fourth, open addressing is employed to handle collisions. Since Java 8 is used for coding if the count of the items in the hash table crosses a threshold Java's HashMap will transform it into a balanced tree. This reduces the time complexity to $O(log\ n)$ which is better than the traditional set enumeration tree that had a time complexity of $O(2\text{^}\{nit\})$. This enhanced the performance of the proposed approach. Here $nt = |DB|$ and $nit = |I|$.

2. PROOF OF CORRECTNESS OF THE ALGORITHM

The proof of correctness of this algorithm is as follow:.

Let $I = \{I_1, I_2, \dots\ I_N\}$ be a set of items and $D = \{T_1, T_2, \dots T_N\}$ be a set of transactions

If item $I_i \in T_i$ then $I_i \leftarrow 1$ else $I_i \leftarrow 0$

If sup[I_i]≥minsup then I_i is frequent and FI ← I_i

Define the $CrossOverPoint \leftarrow length$ of the k-itemset that is being generated

For each item, $I_i \in I$ till *CrossOverPoint* has reached $Fittest[I] \leftrightarrow SecondFittest[I]$

Define the $MutationPoint \leftarrow\ length$ of the k-itemset that is being generated

If *Fittest.I*[*mutationPoint*]*←0* then *fittest. I*[*mutationPoint*]*←1* else *Fittest.I*[*mutationPoint*]*←0*;

If *SecondFittest.I*[*mutationPoint*]*←0* then *SecondFittest.I*[*mutationPoint]←1* else *SecondFittest.I*[*mutationPoint*] *←0*;

CFI={ I | ¬∃ I′ wherein I′⊋I and support(I′)=support (I)}

Extract minimal non-redundant rules. *An association rule r: $R_1 \rightarrow R_2$ is a minimal non-redundant association rule ↔ ∄ an association rule r': $R_1' \rightarrow R_2'$ with support*(*r*) = *support*(*r′*), *confidence*(*r*) = *confidence*(*r′*), *$R_1' \sqsubset R_1$* and *$R_2' \sqsubset R_2$*

3. METHOD

The proposed method aims to address redundancy in itemsets by mining closed itemsets. A genetic algorithm has been designed for the same. In addition, hashing has been employed to mine the resultant collection of closed frequent itemsets. The association rules generated from the discovered itemsets also contain redundancy and a tree-based algorithm that works on the concept of minimal antecedent and maximal consequent has been designed here. The methodology begins with the formulation of a multi-objective function. Maximize $F(X) = max\{(supp(X), Closure(X))\}$. The pseudocode for extracting the closed frequent itemsets is shown in Algorithm 1. This procedure begins its execution by invoking the $Pop\_Init(\ )$ for initializing the population of itemsets as shown in Algorithm 2. This procedure represents the itemsets in bitwise form. The population is selected for the genetic algorithm using rank selection. Rank selection sorts the population first according to fitness value and ranks them. In this case, the fitness value is the minimum support. Then every itemset is allocated selection probability in terms of its rank. Individuals are selected as per their selection probability. All itemsets that meet the minimum support constraint are selected. This forms the set of frequent-1 itemsets. The next step involves carrying out a cross-over operation, as shown in Algorithm 3. This paper uses a single-point cross-over technique to generate the required itemsets. For instance, to find out frequent-2 itemsets, a cross-over point of 2 is fixed. In general, it can be stated that to find a k-itemset a cross-over point of ‘k’ has to be set. Once the cross-over operation is done, a check is made to see if the obtained bitset is a representation of the required itemset. If not, a mutation operation is carried out as in Algorithm 4. This implies that some of the bits in the bit string can be flipped. The work proposed in this paper applies single-point mutation to generate the itemsets. This process repeats itself till all frequent-2 item sets are obtained. While applying cross-over if an itemset is found to be infrequent it is not added to the resultant set and the algorithm proceeds to find the next possible frequent-k itemset. Once the set of frequent-2 itemsets is obtained closure checking operation is applied to mine the final set of closed frequent items. The algorithm then proceeds to repeat the above process for all frequent-k itemsets until the complete set of closed frequent itemsets has been obtained. The next step involves finding non-redundant closed association rules. A tree-based algorithm NR_Prune, as in Algorithm 5, is invoked. This involves the construction of a tree data structure. The set of closed frequent-1 itemsets forms the first level of the tree. The set of association rules that have one antecedent and one consequent form the second level of the tree. Each node has the following fields- antecedent, consequent, support, and confidence. The third level of the tree has 2 antecedents or 2 consequents. The remaining levels increase subsequently. From the second level onwards there is either a left expansion or a right expansion. Left expansion is where the number of antecedents increases by one and right expansion is where the number of consequents increases by 1. The left expansion forms the left sub-tree and the right expansion forms the right sub-tree. Once the tree is constructed each branch is traversed in a depth-wise manner. Redundant rules are pruned out by using the concept of minimal antecedent and maximal consequent. An association rule $A \epsilon R$ is non-redundant and minimal if there is no other association rule $A' \epsilon R$ having the same support and the same confidence, of which the antecedent is a subset of the antecedent of *A* and the consequent is a superset of the consequent of *A*. The general methodology is shown in Figure 1. All experiments were performed on a 2.5 GHz processor with an Intel Pentium i5 core and a RAM of 4.0 GB that uses a Windows operating system of 64 bits. The software used is Java Eclipse IDE. The experimental setup is shown in Figure 2.

Algorithm 1. CFI_GA
```
Input: Dataset D, minimum support min_sup, the maximum number of iterations max_iter
Output: Set of Closed Frequent Itemsets
 1.  Pop_Init( );
 2.  times =1; CFI =∅;
 3.  while times < max_iter do
 4.  for each item X do
 5.  if s(X) ≥ min_sup & X = ∉ FI then
 6.  FI=X;
 7.  Apply Crossover and Mutation
 8.  CFI=X|¬∃ Y wherein Y⊋X and support(X) =support (Y)
 9.  end if
 10. end for
 11. times++;
 12. end while
 13. Output all CFIs.
 14. NR_Prune
```

Algorithm 2. Pop_Init
```
Input: D=[T1, T2,….TN]
Output: Bitmap representation of the individuals.
 1.for each individual item in the transaction till the length of the individual is reached
 2.If (individuals[i] ∈ Ti)
 3.individuals[i]=1;
 4.else
 5.individuals[i]=0;
 6.end if;
 7.end for
```

Algorithm 3. Crossover
```
 1.crossOverPoint = population.individuals[0].itemLength;
 2.for each item ‘i’ till crossOverPoint is reached do
 3.temp = fittest. item [i];
 4.fittest. item [i] = secondFittest. item [i];
 5.secondFittest. item [i] = temp;
 6.end for
```

Algorithm 4. Mutation
```
 1.  mutationPoint = population.individuals[0]. itemLength;
 2.  if (fittest. item[mutationPoint] == 0) then
 3.  fittest. item[mutationPoint] = 1;
 4.  else
 5.  fittest. item[mutationPoint] = 0;
 6.  end if
 7.  if (secondFittest.item[mutationPoint] == 0) then
 8.  secondFittest. item[mutationPoint] = 1;
 9.  else
 10. secondFittest. item[mutationPoint] = 0;
 11. end if
```

Algorithm 5. NR_Prune algorithm
```
Input: A set of CFI and a threshold min_sup.
Output: The set of all non-redundant association rules
 1.  For each node N in the NR-tree do
 2.  Append the NInfo of N into the Nodeset of item N.itemName;
 3.  End for
 4.  Create the node Root;
 5.  Root. Level = 0;
 6.  Root.ChildrenList = ∅
 7.  Root.ItemName = ∅;
 8.  Root. Itemset = ∅;
 9.  Root. rules = ∅;
 10. For each item x ∈ CFI do:
 11. Create Parent P. Itemset={x};
 12. Child x .level = Root. Level + 1;
 13. Child x .rules = ({i}→{j});
 14. Child x. rsup=sup({i}→{j}) / |T|.
 15. Child x. rconf= sup({i}→{j}) / sup({i}
 16. End For
 17. For ∃ node x of the form {i}→{j} AND x.rsup ≥ minsup AND x.rconf ≥ minconf
 18. FoR each item c ϵ i,j do
 19. if sup (i→j∪{c})| / |T| ≥ minsup AND sup (i→j∪{c})/ sup(i) ≥ minconf
 20. Create node R:= R∪{I→J∪{c}}.
 21. Else if sup (i∪{c}→j)| / |T| ≥ minsup AND sup (i∪{c}→j)/ sup(i) ≥ minconf
 22. Create node L:= L∪{I∪{c}→J}.
 23. If R.sup =x.rsup and R.conf=x.rconf
 24. Output R
 25. Else if L.sup=x.rsup and L.conf=x.rconf
 26. Output L
 27. Else NR_Prune
 28. End if
 29. End if
 30. End For
 31. End For
```

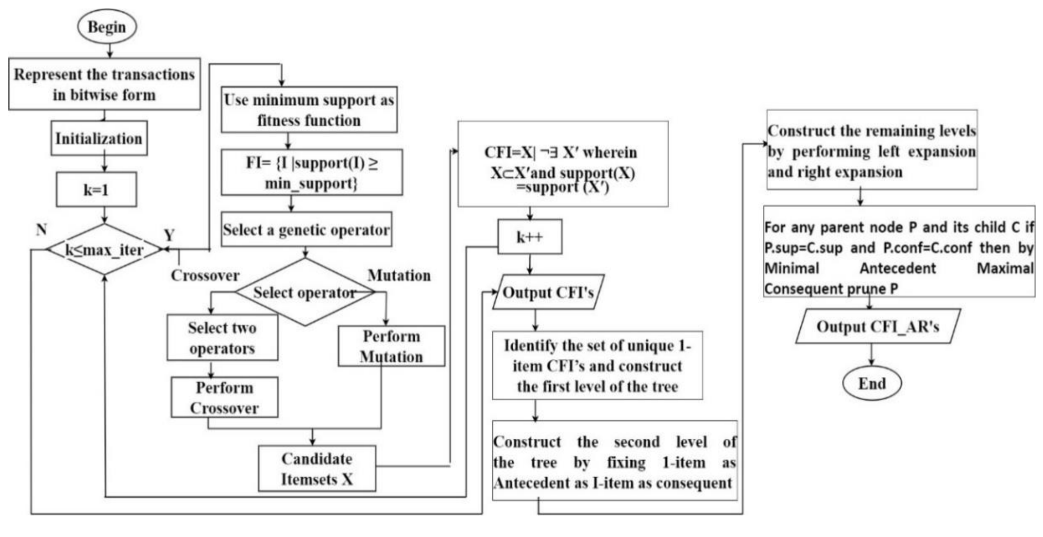
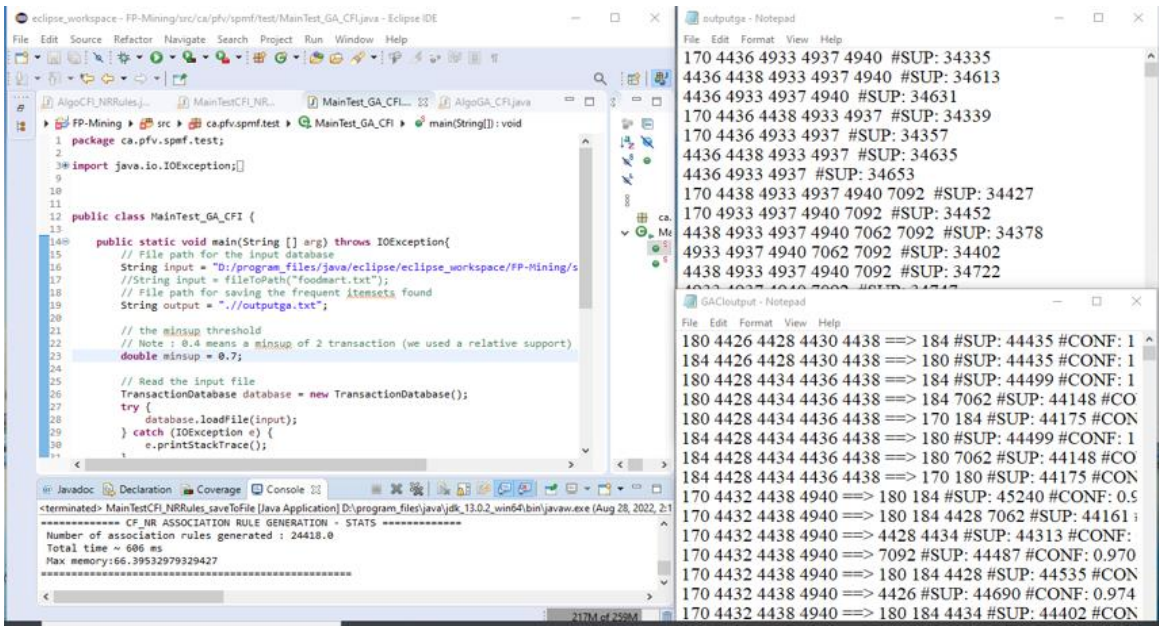

Begin
Represent the transactions in bitwise form
Initialization
k=1
N
k≤max_iter
Y
Use minimum support as fitness function
FI= {I |support(I) ≥ min_support}
Select a genetic operator
Crossover
Select operator
Mutation
Select two operators
Perform Crossover
Perform Mutation
Candidate Itemsets X
CFI=X| ¬∃ X′ wherein X⊂X′and support(X) =support (X′)
k++
Output CFI's
Identify the set of unique 1-item CFI's and construct the first level of the tree
Construct the second level of the tree by fixing 1-item as Antecedent as I-item as consequent
Construct the remaining levels by performing left expansion and right expansion
For any parent node P and its child C if P.sup=C.sup and P.conf=C.conf then by Minimal Antecedent Maximal Consequent prune P
Output CFI_AR's
End

Figure 1. General methodology for extracting closed association rules using genetic algorithm and tree-based pruning


eclipse_workspace - FP-Mining/src/ca/pfv/spmf/test/MainTest_GA_CFI.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
AlgoCFI_NRRules.j... MainTestCFI_NR... MainTest_GA_CFI... AlgoGA_CFI.java
FP-Mining src ca.pfv.spmf.test MainTest_GA_CFI main(String[]) : void
Javadoc Declaration Coverage Console
outputga - Notepad
GACIoutput - Notepad
File Edit Format View Help
217M of 256M

```
package ca.pfv.spmf.test;

import java.io.IOException;


public class MainTest_GA_CFI {

    public static void main(String [] arg) throws IOException{
        // File path for the input database
        String input = "D:/program_files/java/eclipse/eclipse_workspace/FP-Mining/s
        //String input = fileToPath("foodmart.txt");
        // File path for saving the frequent itemsets found
        String output = ".//outputga.txt";

        // the minsup threshold
        // Note : 0.4 means a minsup of 2 transaction (we used a relative support)
        double minsup = 0.7;

        // Read the input file
        TransactionDatabase database = new TransactionDatabase();
        try {
            database.loadFile(input);
        } catch (IOException e) {
            e.printStackTrace();
```

```
<terminated> MainTestCFI_NRRules_saveToFile [Java Application] D:\program_files\java\jdk_13.0.2_win64\bin\javaw.exe (Aug 28, 2022, 2:1
============= CF_NR ASSOCIATION RULE GENERATION - STATS =============
 Number of association rules generated : 24418.0
 Total time ~ 606 ms
 Max memory:66.39532979329427
======================================================
```

```
170 4436 4933 4937 4940  #SUP: 34335
4436 4438 4933 4937 4940  #SUP: 34613
4436 4933 4937 4940  #SUP: 34631
170 4436 4438 4933 4937  #SUP: 34339
170 4436 4933 4937  #SUP: 34357
4436 4438 4933 4937  #SUP: 34635
4436 4933 4937  #SUP: 34653
170 4438 4933 4937 4940 7092  #SUP: 34427
170 4933 4937 4940 7092  #SUP: 34452
4438 4933 4937 4940 7062 7092  #SUP: 34378
4933 4937 4940 7062 7092  #SUP: 34402
4438 4933 4937 4940 7092  #SUP: 34722
```

```
180 4426 4428 4430 4438 ==> 184 #SUP: 44435 #CONF: 1
184 4426 4428 4430 4438 ==> 180 #SUP: 44435 #CONF: 1
180 4428 4434 4436 4438 ==> 184 #SUP: 44499 #CONF: 1
180 4428 4434 4436 4438 ==> 184 7062 #SUP: 44148 #CO
180 4428 4434 4436 4438 ==> 170 184 #SUP: 44175 #CON
184 4428 4434 4436 4438 ==> 180 #SUP: 44499 #CONF: 1
184 4428 4434 4436 4438 ==> 180 7062 #SUP: 44148 #CO
184 4428 4434 4436 4438 ==> 170 180 #SUP: 44175 #CON
170 4432 4438 4940 ==> 180 184 #SUP: 45240 #CONF: 0.9
170 4432 4438 4940 ==> 180 184 4428 7062 #SUP: 44161
170 4432 4438 4940 ==> 4428 4434 #SUP: 44313 #CONF:
170 4432 4438 4940 ==> 7092 #SUP: 44487 #CONF: 0.970
170 4432 4438 4940 ==> 180 184 4428 #SUP: 44535 #CON
170 4432 4438 4940 ==> 4426 #SUP: 44690 #CONF: 0.974
170 4432 4438 4940 ==> 180 184 4434 #SUP: 44402 #CON
```

Figure 2. Experimental set up

3.1. Illustration with example

For the dataset shown in Figure 3(a) the set of frequent itemsets the sets of closed frequent itemsets are as shown in Figures 3(b) and 3(c) respectively. The following steps are carried out till the resultant collection of closed frequent itemsets is obtained.

Dataset				
T1	1	2	4	5
T2	1	3		
T3	1	2	3	5
T4	2	3	5	
T5	1	2	3	5

(a)

Frequent Itemsets
1(4),2(4),3(4),5(4),(1,2)(3), {1,5}(3),{2,3}(3), {2,5}(4),{3,5}(3), {1,2,5}(3),{2,3,5}(3)

(b)

Closed Frequent Itemset
1(4), 2(4), 3(4), 5(4) {2,5}(4), {3,5} (3) {1,2,5}(3),{2,3,5}(3)

(c)

Figure 3. Comparing the number of (a) itemsets generated for dataset, (b) by mining frequent itemsets, and (c) closed frequent itemsets

a) A multi-objective function is designed here. Maximize $F(X) = max\{(supp(X), Closure(X))\}$.
b) Assume $Minsupp$ is set to a value of 3/5= 0.6. This means there should be at least 3 occurrences of an itemset.
c) Choose the initial population
d) Evaluate the fitness of each individual in that population (fitness here is in terms of minimum support)
e) All those items whose support surpasses the minimum support are chosen:
   1 = 1000:5
   2 = 0100:3
   3 = 0010:4
   4 = 0001:4
f) Repeat on this generation until termination: (time limit, and sufficient fitness achieved)
g) Breed new individuals through crossover and mutation operations to give birth to offspring
h) Applying cross-over. Setting cross-over point=2. Value of 2 is chosen since itemsets of size 2 are to be found now. Exchanging the bits in the first 2 places between '1' and '2' we get:
   Hence 12_1 = 1000
   12_2 = 0100. This is shown in Figure 4.
   Since the required representation of '12' is not obtained, we apply mutation on indices e1=2 and e2=1. Hence, we get

$12_1 = 1100$
$12_2 = 1100$. Hence 12 = 1100 and has a support of 3 as per the original dataset. This is because there are 3 cases in the table where ‘1’ and ‘2’ both have a value of 1, i.e. in transactions 1,3,4 respectively.

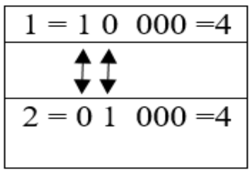
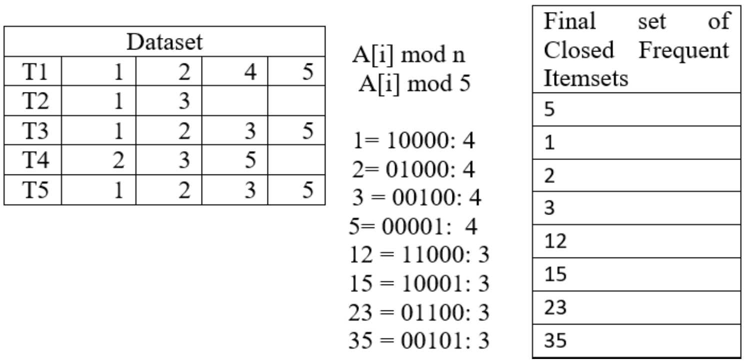


Figure 4. Crossover operation between ‘1’ and ‘2’

i) Next check for closure property. 1' and '2' have a support of 4. Its superset ‘12’ has a support of 3. Since the support of ‘1’ is not the same as that of its superset ‘12’ and ‘1’ is a closed itemset and added to the hash table. Likewise the support of '2' and its superset '12' is not the same so '2' is also a closed itemset and is added to the hash table.
j) A similar procedure is adopted for all itemsets till the complete set of Closed Frequent Itemsets is obtained. The resultant collection of closed frequent itemsets is inserted into the Hash table. This is illustrated in Figure 5.



Dataset				
T1	1	2	4	5
T2	1	3		
T3	1	2	3	5
T4	2	3	5	
T5	1	2	3	5

A[i] mod n
A[i] mod 5

1= 10000: 4
2= 01000: 4
3 = 00100: 4
5= 00001: 4
12 = 11000: 3
15 = 10001: 3
23 = 01100: 3
35 = 00101: 3

Final set of Closed Frequent Itemsets
5
1
2
3
12
15
23
35

Figure 5. Hashing of closed frequent itemsets

k) From the itemsets that are extracted association rules are generated. Redundancy amongst these rules is addressed by constructing a tree structure and employing a pruning strategy based on minimal antecedent and maximal consequent. Only those rules that meet the minimum support and confidence constraint are kept and the rest are eliminated. This is shown in Figure 6.
l) The set of closed frequent-1 itemsets from the first level of the tree. The set of association rules that have one antecedent and one consequent form the second level of the tree.
m) Each node has the following fields- antecedent, consequent, support, and confidence. The third level of the tree has 2 antecedents or 2 consequents and so on the levels increase. From the second level onwards there is either a left expansion or a right expansion. Left expansion is where the number of antecedents increases by one and right expansion is where the number of consequents increases by 1. The left expansion forms the left sub-tree and the right expansion forms the right sub-tree.
n) A Depth-First pruning approach is followed here. The algorithm starts by visiting node 1 first. This node has 3 children. Starting with the first child 1→2. This, in turn, has got 2 children.
o) The first child (1, 5) →2 has a support of 0.6 which is the same as its parent but the confidence is at a value of 1. Hence it is not a redundant node and is printed as output.
p) The next sibling of this node is 1→ (2, 5) which has the same support and confidence as that of its parent. Hence 1→2 is redundant with respect to 1→ (2, 5). But 1→ (2, 5) also has a parent 1→5 with the same support and confidence. Hence 1→5 is also redundant with respect to 1→ (2, 5).
q) Next, we visit node 1→3. This has no children and is thus printed as output. A similar procedure is followed on other nodes as well to generate the complete set of non-redundant association rules.

r) The nodes marked in red are pruned and the rest of the nodes are given as output to produce the final collection of non-redundant association rules.

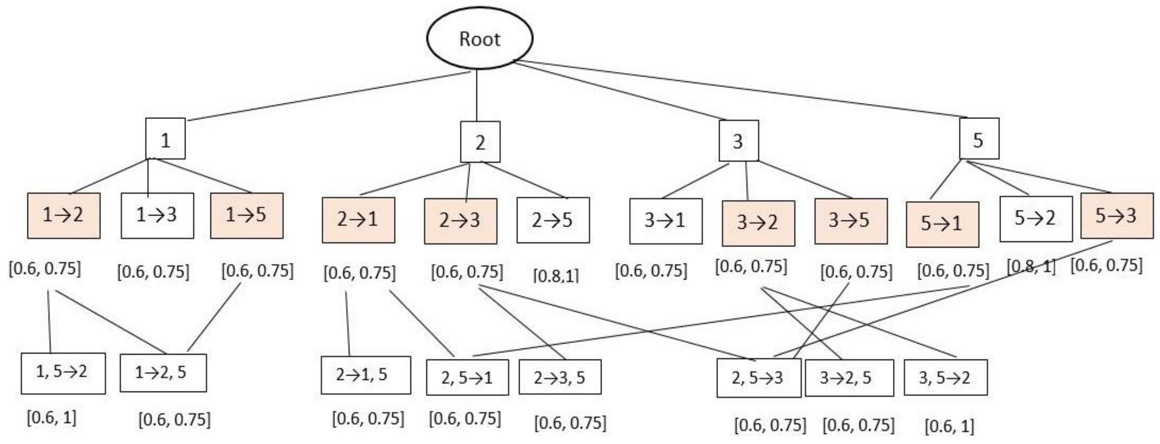


Figure 6. Tree-based pruning of non-redundant association rules

4. RESULTS AND DISCUSSION

This section describes the results obtained while generating closed itemsets and closed association rules. The datasets used here are, PUMSB [47], Susy [48], Connect [49], Retail [50], OnlineRetail [51], and BMS-2 [52]. The details are shown in Table 2.

Table 2. Description of the dataset

Name of the Dataset	Count of the Transactions	Count of the items	Average Item Count Per Transaction	Nature of Dataset
PUMSB	49,046	2113	74	Dense
Susy	5,000,000	190	19	Dense
connect	67,557	129	43	Dense
OnlineRetail	541,909	2,603	4.37	Sparse
Retail	88,162	16,470	10.30	Sparse
BMS-2	59,602	497	2.51	Sparse

4.1. Mining closed frequent itemsets using genetic algorithm

4.1.1. Analysis in terms of run time

The proposed CFI_GA algorithm has been compared with the existing FPClose [53], DCI_Closed [54], N-list-based algorithm for mining frequent closed patterns (NAFCP) [55], and NEclatClosed [56] algorithms. The graphical representation of run time on the different datasets is illustrated in Figure 7. The proposed approach was found to outperform list-based algorithms such as NEclatClosed and NAFCP since it involved the construction of a list data structure which was found to be time-consuming. DCI_Closed, on the other hand, relied on a lattice-based approach and mining itemsets from a lattice were found to involve more time than the methodology proposed in this paper. FPClose algorithm built an FP-Tree for extracting the itemsets. The setting up and navigation of this structure were found to be time-consuming. CFI_GA, on the other hand, relies on operations such as cross-over and mutation that do not involve building and traversing a tree or list data structure. In addition, it uses hashing for generating the final collection of CFI which has a retrieval time of $O(log\ n)$ thereby reducing the overall “time complexity”.

4.1.2. Comparative analysis of memory

An analysis of memory using different datasets for the proposed and existing approaches has been performed. The graphical representation of run time on different datasets is illustrated in Figure 8. The proposed approach was found to outperform existing algorithms since it uses bits to represent the itemsets and bits consume lesser memory space than storing the itemsets as such. Also, since CFI_GA does not involve the construction of a tree or list data structures unlike the existing algorithms, the space required to store these data structures is also avoided thereby enhancing the overall performance of the proposed approach.

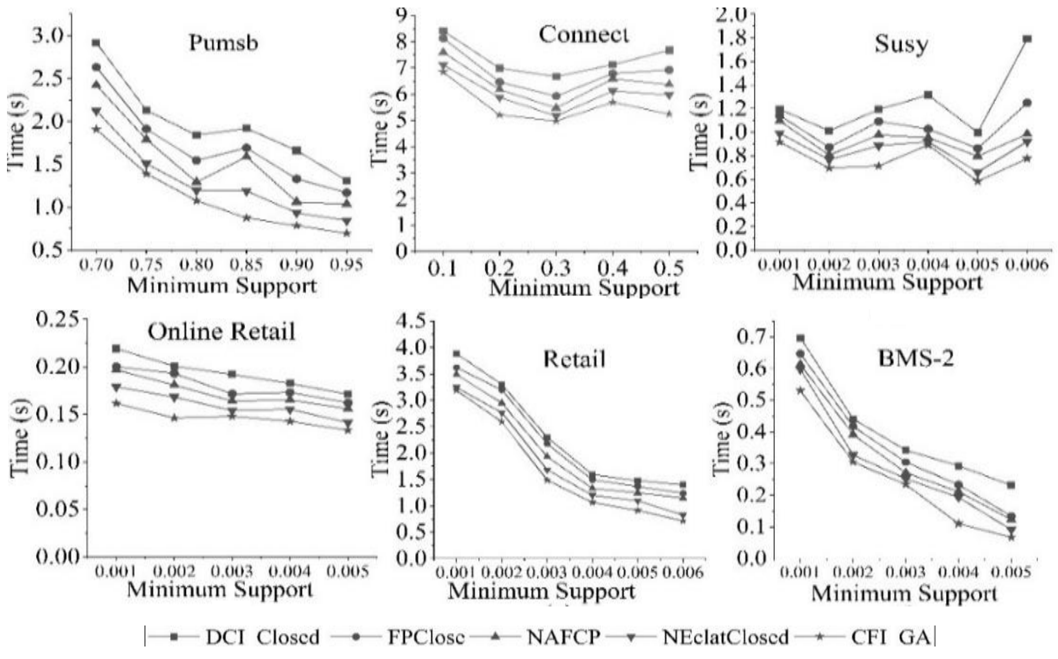


Figure 7. Comparison of the run time of the proposed CFI_GA algorithm with the existing algorithm on different datasets

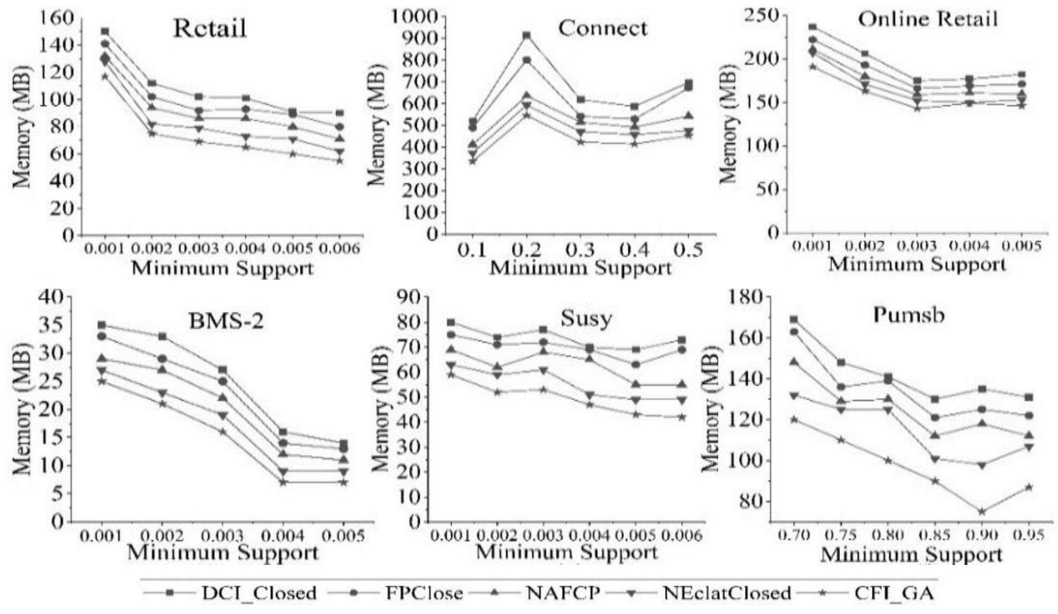


Figure 8. Comparison of the memory consumption of the proposed CFI_GA algorithm with the existing algorithm on different datasets

4.2. Mining closed association rules using tree-based pruning

4.2.1. Analysis in terms of run time

The proposed CFI_NR_ARM algorithm has been compared with the existing MGAR_FCIL [57], and RR generator [58] algorithms. The graphical representation of run time on the different datasets is illustrated in Figure 9. The proposed approach was found to outperform the existing algorithms MGAR_FCIL and RR generator. The existing algorithms employ a breadthwise approach for generating rules and this was found to consume more time. The proposed algorithm on the other hand employs a depth-wise approach by traversing a tree to generate the final collection of closed association rules which reduces the retrieval time considerably.

4.2.2. Comparative analysis of memory

An analysis of memory using different datasets for the proposed and existing approaches has been performed. The graphical representation of run time on different datasets is illustrated in Figure 10. From the graphs, it can be observed that the proposed algorithm outperforms existing algorithms MGAR_FCIL and RR generator in terms of memory. This is because the proposed approach uses bits to represent the itemsets and bits consume lesser memory space than storing the itemsets as such.

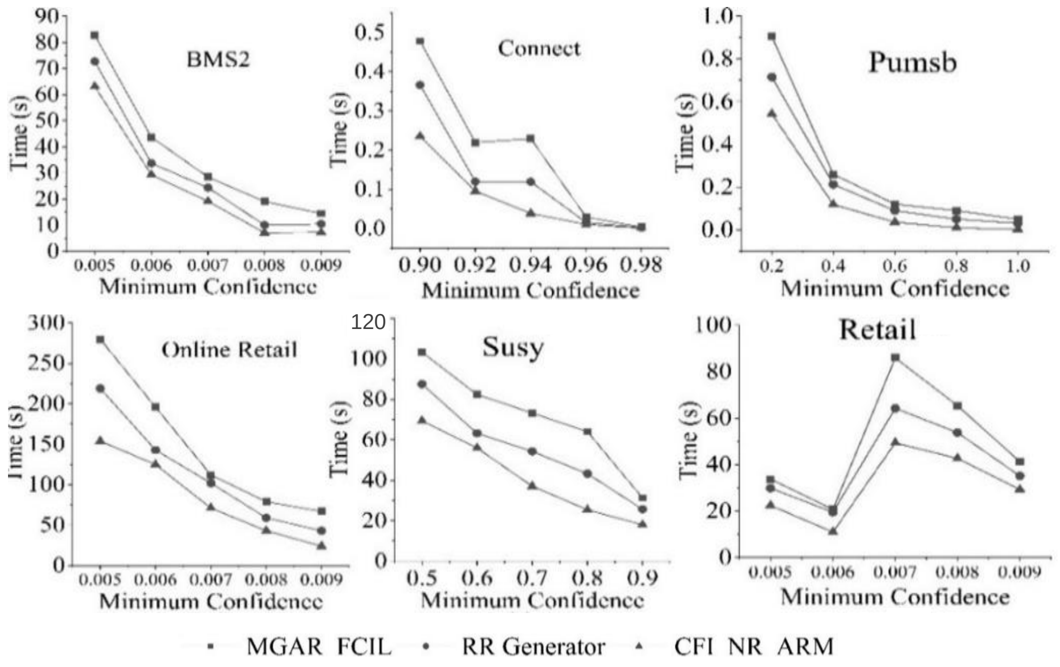


Figure 9. Comparison of the run time of the proposed CFI_NR_ARM algorithm with the existing algorithm on different datasets

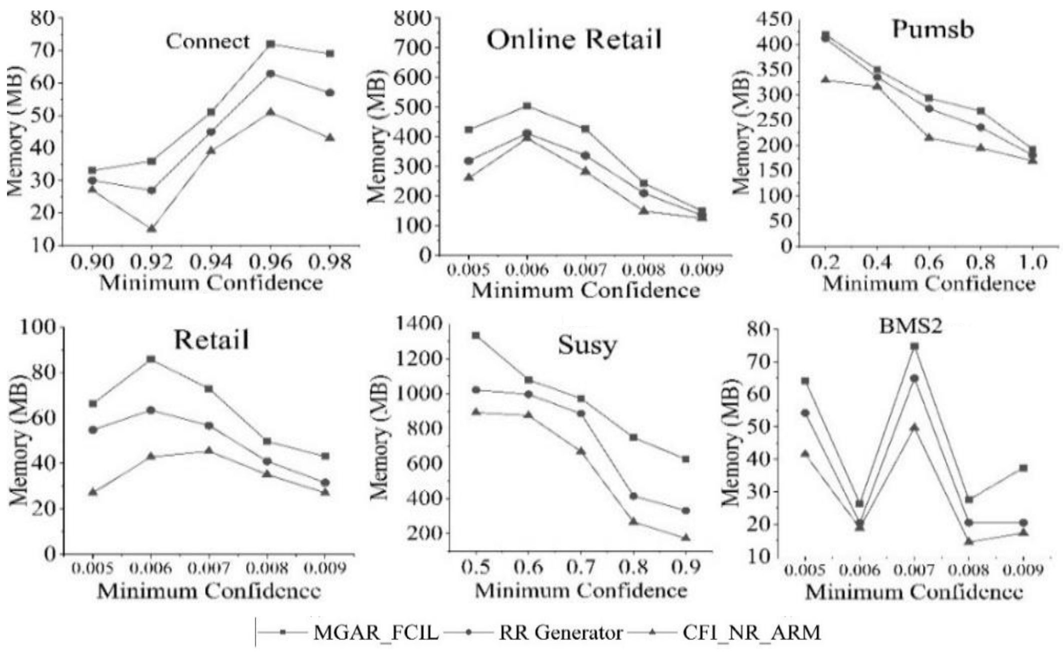


Figure 10. Comparison of the memory consumption of the proposed CFI_NR_ARM algorithm with the existing algorithm on different datasets

5. CONCLUSION

A genetic algorithm has been designed in this paper to mine closed frequent itemsets. To prune out the non-redundant association rules, a tree-based approach has been employed. The tree is constructed by using the concept of left and right expansion. The redundant rules are then pruned out by relying on the principles of minimal antecedent and maximal consequent. The proposed approach has been compared with the existing algorithms in terms of run time and memory. Experiments have shown that the proposed approach consumes significantly lesser memory than the existing approaches. This is because the genetic algorithm-based approach uses bits and bits consume lesser memory than traditional methods. The operations mutation and crossover that are employed here are faster than existing breadthwise approaches thereby improving the runtime as well. As a part of a future plan, this work may be extended to extract association rules in a parallel and distributed framework on big datasets.

REFERENCES

[1] S. A. Ahmed and B. Nath, "ISSP-tree: An improved fast algorithm for constructing a complete prefix tree using single database scan," *Expert Systems with Applications*, vol. 185, Dec. 2021, doi: 10.1016/j.eswa.2021.115603.

[2] R. Davashi, "UP-tree & UP-Mine: A fast method based on upper bound for frequent pattern mining from uncertain data," *Engineering Applications of Artificial Intelligence*, vol. 106, Nov. 2021, doi: 10.1016/j.engappai.2021.104477.

[3] B. Praveen Kumar and D. Paulraj, "Frequent mining analysis using pattern mining utility incremental algorithm based on relational query process," *Journal of Ambient Intelligence and Humanized Computing*, vol. 12, no. 5, pp. 4745–4755, May 2021, doi: 10.1007/s12652-020-01880-9.

[4] Z. Kuang, H. Zhou, D. Zhou, J. Zhou, and K. Yang, "A non-group parallel frequent pattern mining algorithm based on conditional patterns," *Frontiers of Information Technology & Electronic Engineering*, vol. 20, no. 9, pp. 1234–1245, Sep. 2019, doi: 10.1631/FITEE.1800467.

[5] J. C.-W. Lin, U. Ahmed, G. Srivastava, J. M.-T. Wu, T.-P. Hong, and Y. Djenouri, "Linguistic frequent pattern mining using a compressed structure," *Applied Intelligence*, vol. 51, no. 7, pp. 4806–4823, Jul. 2021, doi: 10.1007/s10489-020-02080-w.

[6] B. Li, Z. Pei, K. Qin, and M. Kong, "TT-miner: Topology-transaction miner for mining closed itemset," *IEEE Access*, vol. 7, pp. 10798–10810, 2019, doi: 10.1109/ACCESS.2018.2888627.

[7] Y. Xun, X. Cui, J. Zhang, and Q. Yin, "Incremental frequent itemsets mining based on frequent pattern tree and multi-scale," *Expert Systems with Applications*, vol. 163, Jan. 2021, doi: 10.1016/j.eswa.2020.113805.

[8] Y. Yamamoto, Y. Tabei, and K. Iwanuma, "PARASOL: a hybrid approximation approach for scalable frequent itemset mining in streaming data," *Journal of Intelligent Information Systems*, vol. 55, no. 1, pp. 119–147, Aug. 2020, doi: 10.1007/s10844-019-00590-9.

[9] S. Paramonov, D. Stepanova, and P. Miettinen, "Hybrid ASP-based approach to pattern mining," *Theory and Practice of Logic Programming*, vol. 19, no. 04, pp. 505–535, Jul. 2019, doi: 10.1017/S1471068418000467.

[10] T. Hashem, M. Rezaul Karim, M. Samiullah, and C. Farhan Ahmed, "An efficient dynamic superset bit-vector approach for mining frequent closed itemsets and their lattice structure," *Expert Systems with Applications*, vol. 67, pp. 252–271, Jan. 2017, doi: 10.1016/j.eswa.2016.09.023.

[11] M.-T. Tran, B. Le, and B. Vo, "Combination of dynamic bit vectors and transaction information for mining frequent closed sequences efficiently," *Engineering Applications of Artificial Intelligence*, vol. 38, pp. 183–189, Feb. 2015, doi: 10.1016/j.engappai.2014.10.021.

[12] H. Bui, B. Vo, T.-A. Nguyen-Hoang, and U. Yun, "Mining frequent weighted closed itemsets using the WN-list structure and an early pruning strategy," *Applied Intelligence*, vol. 51, no. 3, pp. 1439–1459, Mar. 2021, doi: 10.1007/s10489-020-01899-7.

[13] F. Fumarola, P. F. Lanotte, M. Ceci, and D. Malerba, "CloFAST: closed sequential pattern mining using sparse and vertical id-lists," *Knowledge and Information Systems*, vol. 48, no. 2, pp. 429–463, Aug. 2016, doi: 10.1007/s10115-015-0884-x.

[14] A. Y. Rodríguez-González, F. Lezama, C. A. Iglesias-Alvarez, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and E. M. de Cote, "Closed frequent similar pattern mining: Reducing the number of frequent similar patterns without information loss," *Expert Systems with Applications*, vol. 96, pp. 271–283, Apr. 2018, doi: 10.1016/j.eswa.2017.12.018.

[15] M. Ledmi, S. Zidat, and A. Hamdi-Cherif, "GrAFCI+ A fast generator-based algorithm for mining frequent closed itemsets," *Knowledge and Information Systems*, vol. 63, no. 7, pp. 1873–1908, Jul. 2021, doi: 10.1007/s10115-021-01575-3.

[16] B. Peddireddy, C. Anuradha, and P. S. R. C. Murthy, "Mining closed item sets from tuple-evolving data streams," *International Journal of Engineering and Advanced Technology*, vol. 8, no. 6, pp. 5010–5016, Aug. 2019, doi: 10.35940/ijeat.F9107.088619.

[17] S. Abed, A. A. Abdelaal, M. H. Al-Shayeji, and I. Ahmad, "SAT-based and CP-based declarative approaches for Top-Rank- K closed frequent itemset mining," *International Journal of Intelligent Systems*, vol. 36, no. 1, pp. 112–151, Jan. 2021, doi: 10.1002/int.22294.

[18] S. Iqbal, A. Shahid, M. Roman, Z. Khan, S. Al-Otaibi, and L. Yu, "TKFIM: Top-K frequent itemset mining technique based on equivalence classes," *PeerJ Computer Science*, vol. 7, Mar. 2021, doi: 10.7717/peerj-cs.385.

[19] A. A. Abdelaal, S. Abed, M. Al-Shayeji, and M. Allaho, "Customized frequent patterns mining algorithms for enhanced top-rank-K frequent pattern mining," *Expert Systems with Applications*, vol. 169, May 2021, doi: 10.1016/j.eswa.2020.114530.

[20] Y. Kameya, "An exhaustive covering approach to parameter-free mining of non-redundant discriminative itemsets," in *DaWaK 2016: Big Data Analytics and Knowledge Discovery*, 2016, pp. 143–159.

[21] S. Bagui and P. Stanley, "Mining frequent itemsets from streaming transaction data using genetic algorithms," *Journal of Big Data*, vol. 7, no. 1, Dec. 2020, doi: 10.1186/s40537-020-00330-9.

[22] S. Hussein Toman, M. H. Abed, and Z. H. Toman, "Cluster-based information retrieval by using (K-means)-hierarchical parallel genetic algorithms approach," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 19, no. 1, pp. 349–356, Feb. 2021, doi: 10.12928/telkomnika.v19i1.16734.

[23] Y. Djenouri and M. Comuzzi, "Combining Apriori heuristic and bio-inspired algorithms for solving the frequent itemsets mining problem," *Information Sciences*, vol. 420, pp. 1–15, Dec. 2017, doi: 10.1016/j.ins.2017.08.043.

[24] U. Maulik, S. Mallik, A. Mukhopadhyay, and S. Bandyopadhyay, "Analyzing large gene expression and methylation data profiles using StatBicRM: Statistical biclustering-based Rule mining," *PLOS ONE*, vol. 10, no. 4, Apr. 2015, doi: 10.1371/journal.pone.0119448.

[25] C. Babi, M. V. Rao, V. V. Rao, and B. Arketla, "Optimized partitioning based genetic algorithm for generating mining frequent patterns from big data sets," *International Journal of Innovative Technology and Exploring Engineering (IJITEE*, vol. 8, no. 7, pp. 289–295, 2019.

[26] M. M. Madbouly, E. A. El Reheem, and S. K. Guirguis, "Interval type-2 fuzzy logic using a genetic algorithm to reduce redundant association rules," *Journal of Theoretical and Applied Information Technology*, vol. 99, no. 2, pp. 316–329, 2021.

[27] S. Ghosh, S. Biswas, D. Sarkar, and P. Sarkar, "Mining frequent itemsets using genetic algorithm," *International Journal of Artificial Intelligence & Applications*, vol. 1, no. 4, pp. 133–143, Oct. 2010, doi: 10.5121/ijaia.2010.1411.

[28] S. Rathi, R. Soni, and V. S. Kushwah, "A new algorithm for privacy preservation in utility mining using genetic algorithm," in *Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies - ICTCS '16*, 2016, pp. 1–6, doi: 10.1145/2905055.2905231.

[29] A. Derouiche, A. Layeb, and Z. Habbas, "Mining interesting association rules with a modified genetic algorithm," in *MedPRAI 2020: Pattern Recognition and Artificial Intelligence*, 2021, pp. 274–285.

[30] B. Vo and B. Le, "Mining minimal non-redundant association rules using frequent itemsets lattice," *International Journal of Intelligent Systems Technologies and Applications*, vol. 10, no. 1, pp. 92–106, 2011, doi: 10.1504/IJISTA.2011.038265.

[31] X. Du and F. Yu, "A fast algorithm for mining temporal association rules in a multi-attributed graph sequence," *Expert Systems with Applications*, vol. 192, Apr. 2022, doi: 10.1016/j.eswa.2021.116390.

[32] H. Li and P. C.-Y. Sheu, "A scalable association rule learning and recommendation algorithm for large-scale microarray datasets," *Journal of Big Data*, vol. 9, no. 1, Dec. 2022, doi: 10.1186/s40537-022-00577-4.

[33] A. Dahbi, S. Jabri, Y. Balouki, and T. Gadi, "The selection of the relevant association rules using the ELECTRE method with multiple criteria," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 4, pp. 638–645, Dec. 2020, doi: 10.11591/ijai.v9.i4.pp638-645.

[34] M. Shawkat, M. Badawi, S. El-ghamrawy, R. Arnous, and A. El-desoky, "An optimized FP-growth algorithm for discovery of association rules," *The Journal of Supercomputing*, vol. 78, no. 4, pp. 5479–5506, Mar. 2022, doi: 10.1007/s11227-021-04066-y.

[35] F. F. Patacsil and M. Acosta, "Analyzing the relationship between information technology jobs advertised on-line and skills requirements using association rules," *Bulletin of Electrical Engineering and Informatics*, vol. 10, no. 5, pp. 2771–2779, Oct. 2021, doi: 10.11591/eei.v10i5.2590.

[36] I. Indrabayu, I. S. Areni, A. Bustamin, and R. Irianty, "A real-time data association of internet of things based for expert weather station system," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 2, pp. 432–439, Jun. 2022, doi: 10.11591/ijai.v11.i2.pp432-439.

[37] M. Sornalakshmi *et al.*, "An efficient apriori algorithm for frequent pattern mining using mapreduce in healthcare data," *Bulletin of Electrical Engineering and Informatics (BEEI)*, vol. 10, no. 1, pp. 390–403, Feb. 2021, doi: 10.11591/eei.v10i1.2096.

[38] I. Sadgali, N. Sael, and F. Benabbou, "Human behavior scoring in credit card fraud detection," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 3, pp. 698–706, Sep. 2021, doi: 10.11591/ijai.v10.i3.pp698-706.

[39] A. Mouakher, F. Hajjej, and S. Ayouni, "Efficient mining support-confidence based framework generalized association rules," *Mathematics*, vol. 10, no. 7, Apr. 2022, doi: 10.3390/math10071163.

[40] M. Vives-Mestres, R. S. Kenett, S. Thió-Henestrosa, and J. A. Martín-Fernández, "Measurement, selection, and visualization of association rules: A compositional data perspective," *Quality and Reliability Engineering International*, vol. 38, no. 3, pp. 1327–1339, Apr. 2022, doi: 10.1002/qre.2910.

[41] W. A. Wan Abu Bakar, M. Man, M. Man, and Z. Abdullah, "i-Eclat: performance enhancement of Eclat via incremental approach in frequent itemset mining," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 18, no. 1, pp. 562–570, Feb. 2020, doi: 10.12928/telkomnika.v18i1.13497.

[42] D. Bui-Thi, P. Meysman, and K. Laukens, "MoMAC: Multi-objective optimization to combine multiple association rules into an interpretable classification," *Applied Intelligence*, vol. 52, no. 3, pp. 3090–3102, Feb. 2022, doi: 10.1007/s10489-021-02595-w.

[43] P. Naresh and R. Suguna, "IPOC: an efficient approach for dynamic association rule generation using incremental data with updating supports," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 24, no. 2, pp. 1084–1090, Nov. 2021, doi: 10.11591/ijeecs.v24.i2.pp1084-1090.

[44] V. Sivasankarareddy and G. Sundari, "Survey on wireless sensor networks: energy efficient optimization routing algorithms," *Indonesian Journal of Electrical Engineering and Computer Science (IJEECS)*, vol. 19, no. 2, pp. 756–765, Aug. 2020, doi: 10.11591/ijeecs.v19.i2.pp756-765.

[45] E. Sun, "A survey on clustering routing protocols based on PSO in WSN," *TELKOMNIKA (Indonesian Journal of Electrical Engineering)*, vol. 12, no. 7, pp. 5484–5490, Jul. 2014, doi: 10.11591/telkomnika.v12i7.4763.

[46] A. Hodijah and U. T. Setijohatmo, "Analysis of frequent itemset generation based on trie data structure in apriori algorithm," *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, vol. 19, no. 5, pp. 1553–1564, Oct. 2021, doi: 10.12928/telkomnika.v19i5.19273.

[47] R. Bayardo, "PUMSB," *UCI Repository*, 1996. https://archive.ics.uci.edu/ml/datasets/census+income (accessed Jan. 04, 2022).

[48] D. Whiteson, "Susy," *UCI repositor*, 2014. https://archive-beta.ics.uci.edu/ml/datasets/susy (accessed Jul. 31, 2022).

[49] J. Tromp, "Connect-4," *UCI repository*, 1995. https://archive-beta.ics.uci.edu/ml/datasets/connect+4 (accessed Jul. 31, 2021).

[50] T. Brijs, "Retail market basket data set," *Frequent Itemset Mining Repository*, 2003. http://fimi.uantwerpen.be/data/ (accessed Jul. 31, 2021).

[51] D. D. Chen, "Online retail data set," 2015. https://archive.ics.uci.edu/ml/datasets/online+retail (accessed Dec. 18, 2021).

[52] R. Kohavi, C. E. Brodley, B. Frasca, L. Mason, and Z. Zheng, "BMS_WebView_2," *Frequent Itemset Mining repository*. https://www.philippe-fournier-viger.com/spmf/datasets/BMS2_itemset_mining.txt (accessed Jul. 31, 2022).

[53] G. Grahne and J. Zhu, "Fast algorithms for frequent itemset mining using FP-trees," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 10, pp. 1347–1362, Oct. 2005, doi: 10.1109/TKDE.2005.166.

[54] C. Lucchese, S. Orlando, and R. Perego, "DCI Closed: a fast and memory efficient algorithm to mine frequent closed itemsets," *Freq. Itemset Min*, pp. 1–9, 2004.

[55] T. Le and B. Vo, "An N-list-based algorithm for mining frequent closed patterns," *Expert Systems with Applications*, vol. 42, no. 19, pp. 6648–6657, Nov. 2015, doi: 10.1016/j.eswa.2015.04.048.

[56] N. Aryabarzan and B. Minaei-Bidgoli, "NEclatClosed: A vertical algorithm for mining frequent closed itemsets," *Expert Systems with Applications*, vol. 174, Jul. 2021, doi: 10.1016/j.eswa.2021.114738.

[57] B. Vo, T.-P. Hong, and B. Le, "A lattice-based approach for mining most generalization association rules," *Knowledge-Based Systems*, vol. 45, pp. 20–30, Jun. 2013, doi: 10.1016/j.knosys.2013.02.003.

[58] C. Tîrnăucă, J. L. Balcázar, and D. Gómez-Pérez, "Closed-set-based discovery of representative association rules," *International Journal of Foundations of Computer Science*, vol. 31, no. 01, pp. 143–156, Jan. 2020, doi: 10.1142/S0129054120400109.

BIOGRAPHIES OF AUTHORS

Jashma Suresh Ponmudiyan Poovan is currently working as an assistant professor in the Department of Computer Science and Engineering at Manipal Institute of Technology, Manipal, India, and has 8 years of experience to her credit. She is pursuing a Ph.D. from Manipal Institute of Technology, Manipal Academy of Higher Education in the area of data mining. She obtained her master's degree in computer science and engineering from Anna University (2013) and her bachelor's degree from Kannur University (2010). Her research interests include data mining, data analytics, and machine learning. She can be contacted at jashma.suresh@manipal.edu.

Dinesh Acharya Udupi is serving as a professor at Manipal Institute of Technology in the Department of Computer Science and Engineering. He completed his Ph.D. at the Manipal Institute of Technology in 2008. With over 30 years of experience, he has published papers in several journals and presented articles at conferences at both national and international levels. His area of interest is data mining, knowledge discovery, and data science. He can be contacted at dinesh.acharya@manipal.edu.

Nandanavana Veerappareddy Subba Reddy is currently serving as HOD in the Department of IT, MIT Bengaluru. With over 30 years of experience, he has published papers in several journals and presented articles at conferences at both national and international levels. His area of interest includes machine learning, data mining, pattern recognition, and image processing. He can be contacted at nvs.reddy@manipal.edu.