An Effective Reference-Point-Set (RPS) Based Bi-Directional Frequent Itemset Generation

Ambily Balaram
Department of Computer Science and Engineering,
Coimbatore Institute of Technology, Anna University, India
ambilybalaram@gmail.com

Nedunchezhian Raju
Department of Computer Science and Engineering,
Coimbatore Institute of Technology, Anna University, India
nedunchezhian@cit.edu.in

Abstract: *Data Mining (DM) is a combination of several fields that effectively extracts hidden patterns from vast amounts of historical data. One of the DM activities used to produce association rules is Association Rule Mining (ARM). To significantly reduce time and space complexities, the proposed method utilizes an effective bi-directional frequent itemset generation approach. The dataset is explicitly bifurcated into dense and sparse regions in the process of mining frequent itemset. One more feature is proposed in this paper which sensibly predetermines a candidate subset called, Reference-Points-Set (RPS), to reduce the complexities associated with mining of frequent itemsets. The RPS helps to reduce the number of scans over the actual dataset. The novelty is to look at possible candidates during the initial database scans, which can cut down on the number of additional database scans that are required. According to experimental data, the average scan count of the proposed method is respectively, 24% and 65%, lower than that of Dynamic Itemset Counting (DIC) and M-Apriori, across different support counts. The proposed method typically results in a 10% reduction in execution time over DIC and is three times more efficient than M-Apriori. These results significantly outperform those of their predecessors, which strongly supports the proposed approach when creating frequent itemsets from large datasets.*

Keywords: *Reference point, reference-point-set, transactional buckets, sparse region, dense region.*

Received October 20, 2022; accepted May 29, 2023
https://doi.org/10.34028/iajit/20/6/6

1. Introduction

In today's rapidly evolving technological and business world, the power of data, information and knowledge inferred out of the data are well understood and the same is applied in business and scientific activities. Due to the technological advancements, a huge amount of data is generated, collected, stored and analyzed in real world applications. Statistics, Artificial Intelligence (AI), Machine Learning (ML), database technologies, and some other technologies are used in the field of Data Mining (DM). DM functionalities like Association Rule Mining (ARM) [1, 2, 6], clustering [11], classification [11], regression, and outlier detection [11] can be used to extract the hidden knowledge from a large amount of data. Typically, an association rule finds the affinity between data points. Association rule generation comprises of two phases:

1. Mining of the itemset that are frequent according to the user-specified minimum threshold
2. The process of deriving strong rules from the frequent itemsets discovered in the first step of ARM.

The benchmark ARM algorithms are Apriori [2], Dynamic Itemset Counting (DIC) [4], and FP- Growth method [12]. The Apriori algorithm is the sentinel algorithm that follows the bottom-up method to mine frequent itemsets through a generate-and-test approach from a large amount of transactional data. The Apriori Algorithm scans the dataset iteratively as the frequent itemsets are mined incrementally. The generation of candidates and counting of their support are really time and space consuming processes. Many methods have been proposed to address these difficulties, including many Apriori variants. Some of the methods include hash-based itemset counting [18], partitioning [20], and sampling [23]. DIC is an extension of Apriori algorithm that permits addition and omission of candidate itemsets at multiple stages of the database scans with an objective to reduce the time complexity. In order to eventually organize the itemsets into frequent or infrequent, it uses four distinct data structures.

The paper proposes an effective frequent itemset generation approach using Transactional Bucket (TB) and Reference-Point-Set (RPS) to find frequent itemsets with minimum number of data source scans of the transactional data source. A set of reference point candidates, RPS, produced from Reference Point (RP), is used to guide the mining process further. These RPS is derived using the weightage of certain buckets in the TB. The length of RP, derived through the combination formula, nCr, is used to generate RPS from every itemset in the TB. The method bifurcates the data into dense and sparse regions. The suggested methodology is combined with DIC, to make a hybrid approach that better demonstrates the benefits of the proposed approach.

The novelty in the proposed method is that in the early runs of the database scan itself it looks at the possible potential candidates of varying sizes. The process of TB

construction and RPS formation cuts down the requirement for additional database scans. The hybridization is accomplished by enhancing DIC with the following features:

1. Buckets are additionally created.
2. RPS are generated anticipating the reduction in the number of database scans.

The effectiveness of the method is theoretically and experimentally proved, and the results show that a significant increase in performance over its predecessors.

The paper is organized as follows: Section 2 presents the relevant background work. In section 3, the definition and problem statement are explained. In section 4, the proposed method is explained. Section 5 deals with results and discussion. Section 6 concludes the paper.

2. Literature Review

ARM [1, 4] is applied on a dataset to detect the affinity, association or correlation among variables of the dataset that essentially focuses on market-basket analysis. Retailing, clustering, classification, cross marketing, and catalog design are some of the uses of ARM. The Market-basket analysis approach involves observing and researching consumer shopping habits, thus identifying correspondence between items in their baskets. Over the years, several methods for discovering strong association rules [13, 19, 24, 28] have been proposed. A few benchmark papers are explained in this section.

Zaki [26] proposed hybrid search strategies that used both BFS and DFS. Using lattice traversal technique, frequent itemsets and their subsets were generated. The issue has been broken down into sub-issues, such as prefix-based and maximum clique-based partitions. With the decrease in the support count, which accounts for the time complexity, large number of frequent itemsets was generated.

In [21], the Index-BitTableFI algorithm used bit - wise operations and it exploited the BitTable vertically and horizontally. A combination of BFS and DFS was used for the generation of frequent itemsets with index array support. The BitTable-based algorithm cannot perform well in the event of large datasets as there is a massive memory requirement to retain the BitTable representation.

Chen and Xiao [7] have proposed an algorithm for Bitmap Itemset Support Counting (BISC) consisting of three techniques, namely BISC1, BISC2 and technology for projection. All frequent DFS-based element sets are contained in BISC1. In BISC2, the database consisted of two elements, namely the prefix and the suffix.

Wang *et al*. [24] investigated the significant issue of extracting frequent item sets from a large uncertain database using the Possible World Semantics (PWS) concept. Also, two incremental mining algorithms has been proposed for discovering Probabilistic Frequent Itemset (PFIs), to avoid having to run the entire mining algorithm over on a new database.

To get over the drawbacks of the traditional Apriori algorithm, Maolegi and Arkok [17] introduced an upgraded Apriori algorithm, called M-Apriori. The algorithm's primary goal was to shorten the amount of time needed to scan the entire database in finding frequently itemsets. When compared to the original Apriori, the, the M-Apriori utilized 67.38% less time, making it more effective and less time-consuming. A detailed comparative study in the performance of the proposed method and M-Apriori are made in the results and discussions section.

Webb and Vreeken [25] proposed the branch-and-bound OPUS Miner algorithm that used a DFS search strategy and pruning mechanisms to generate self-sufficient item sets that summarized the key associations in the data.

Leeuwen and Galbrun [13] proposed an approach where attributes were divided into two sets that could provide two separate views in order to define the data structure of the same object set. The translation tables were used to include both unidirectional and bidirectional rules with a translation of the two views. To pick the translation tables, a score based Minimum Definition Length (MDL) concept was suggested along with the translator algorithm to find good models.

Utilizing a variety of cooccurrences and kernel item occurrences in at least one transaction, Phan and Le [19] presented a NOV-FI technique to swiftly find common itemsets from transactional databases. The proposed method addressed the problem of discovering frequent itemsets on search space items with at least a minimum support and without reusing for mining the succeeding time.

In the study [5], the paper proposed a method of Recent Maximum Frequent Itemsets Mining (RMFIsM) that worked with two matrixes to store each transaction's data and frequent 1-itemsets respectively. To achieve successful results, the authors performed a series of experiments

Zhang *et al*. [27] presented HashEclat, an approximation Eclat method based on MinHash that could quickly estimate the size of the intersection set and change the parameters to address the tradeoff between mining accuracy and execution time.

The study in [3] explored numerous connections between sliding window size, genetic algorithm limitations, and notion drift and described a method for mining common itemsets from streaming transaction data using genetic algorithms.

In [22], a novel Fast Incremental Updating Frequent Pattern growth (FIUFP-growth) based on Incremental Conditional Pattern tree (ICP-tree) was proposed. By mining frequent sets gradually, the tree produced incremental association rule mining. Additionally, the

plan reduced the amount of time and resources needed to build their sub-trees.

In their study, Zhao *et al*. [28] provided a methodical, in-depth, and thorough investigation on DM technologies. The proposed mining method reduced the noise in data, and also the precision is kept very high, indicating the greatest accuracy of the method.

The study in the paper [8] offered a comprehensive framework that used Generic Itemset Mining based on Reinforcement Learning (GIM-RL) to train agents to extract itemsets. Since the investigation was iterative, it was possible that GIM-RL recovered the necessary itemsets from datasets. When changes were made to existing item sets, either additions or deletions, the proposed environment informed agents about the appropriate itemsets for the target kinds. With the aid of successive trial-and-error phases and rewards for different actions, agents learned how to maximize cumulative benefits. As a consequence, the most necessary item sets were generated.

Li *et al.* [15] proposed a matrix-based strong association rule extraction approach to process and represent information, setting the minimal support to 10%, the minimum confidence to 80%, and obtaining all frequent item sets from the data set.

Magdy *et al.* [16] proposed the Closed Candidates-based Incremental Frequent Itemset Mining technique (CC-IFIM) to reduce candidate generation and boost the reliability of the obtained global frequent itemsets.

The proposed methodology is combined with DIC, one of the traditional frequent itemset methods, to make a hybrid approach that better demonstrates the benefits of the proposed approach. The following subsection 2.1 of the DIC algorithm provides an explanation of its basic operation.

2.1. Dynamic Itemset Counting (DIC)

The DIC [4] approach is used to produce frequent itemsets with fewer transaction databases scans. It eliminates some of the drawbacks faced by Apriori. When each transaction is read from the dataset, at various stages of a scan, the itemsets are added and pruned dynamically. Like Apriori, two user specified measures are used. When all its subsets are frequent and its support count is greater or equal to the threshold, minsupp, the candidate itemset is considered as frequent itemset. The DIC introduces multiple stopping points, M, which indicates the number of virtual partitions in the dataset. In the respective counters, the support of each item collection is counted and retained.

The method follows an incremental approach in adding and deleting the itemset in the above-mentioned stopping points. The DIC method follows the anti-monotone property stating that all of a set's subgroups must be frequent in order for a set of items to be frequent. The general Algorithm (1) of DIC is presented below [10].

Algorithm 1: Dynamic Itemset Counting

Input: transactions from database M, Stopping Points Threshold, minimum support threshold

Output: c, frequent itemsets

```
1: SS=ϕ // solid square (frequent)
2: SC=ϕ // solid circle (infrequent)
3: DS = ϕ // dashed square (suspected frequent)
4: DC= {all 1-itemsets} // dashed circle (suspected infrequent)
5: While (DS! =0) or (DC! =0) do begin
6:    read M transactions from database into T
7:    for all transactions t ϵ T do begin
         //increment the respective counters of the itemsets
         marked with dash
8:       for each itemset c in DS or DC do begin
9:          if (c ϵ t) then
10:            c. counter++;
11:      for each itemset c in DC
12:         if (c. counter ≥ threshold) then
13:            move c from DC to DS;
14:               if (any immediate superset sc of c has all of its
subsets in SS or DS) then
15:                  add a new itemsets c in DC;
16:    end
17:    for each itemset c in DS
18:       if (c has been counted through all transactions) then
19:          move it into SS;
20:    for each itemset c in DC
21:       if (c has been counted through all transactions) then
22:          move it into SC;
23:     end
24: end
25: Answer = {c ϵ SS};
```

Normally, the number of scans over the transaction databases, the number of candidates itemsets produced, the support-confidence threshold settings and the time and space complexities are the major issues in most ARM algorithms. To overcome these issues the paper proposes an enhanced RPS based bi-directional scanning method.

3. Definitions and Problem Statement

The formal description of some of the significant terms used in this paper and the problem definition are given in this section.

3.1. Definitions

Let D be the transactional data source and I be the item domain, I={i1, i2, …. in}. Let T be the set of transactions, T={T1, T2,……Tn}. An association rule is an expression of implication of the type $A \Rightarrow B$, where $A \subseteq B$, $B \subseteq I$, and $A \cap B = \phi$. To generate association rules, first the frequent itemsets have to be mined from the data source D. Table 1 shows an example of the transactional data Source, D, as the

running example to illustrate the concepts of the method clearly [11].

Table 1. Transactional date source, D [11].

TID	List of items	TID	List of items
T_1	{I_1, I_2, I_5}	T_7	{I_1, I_3}
T_2	{I_2, I_4, I_6}	T_8	{I_1, I_2, I_3, I_5}
T_3	{I_2, I_3}	T_9	{I_1, I_2, I_3}
T_4	{I_1, I_2, I_4}	T_{10}	{I_1, I_2, I_4, I_6}
T_5	{I_1, I_3}	T_{11}	{I_5, I_6}
T_6	{I_2, I_3}	T_{12}	{I_3, I_4, I_5}

The above table contains twelve transactions and each transaction is a subset of I is {I1, I2, I3, I4, I5, I6} and the Transactional Identifier, TID ={T1, T2, T3, T4, T5, T6, T7, T8, T9, T10, T11, T12}. Assume that minsupp is 2.

- Support is a measure that provides the occurrence frequency of an itemset in the table, T. Support (A→B) =P (A∪B), defines the joint probability of the itemset being present in the transactions. For example, itemset {I1} is in T1, T4, T5, T7, T8, T9, and T10, therefore, support ({I1}) = 7. Itemset {I1, I2} is in T1, T4, T8, T9 and T10, therefore, support ({I1, I2})=5.
- Confidence measures the strength of an Assertion Rule, i.e., Confidence (A→B)=P (B|A), defines the conditional probability of itemsets in the transactions.
- Frequent Itemsets: an itemset is frequent if it satisfies the defined minimum support threshold. i.e., itemsets' support is greater than or equal to the predefined minimum support threshold value, minsupp. For example, itemset {I2, I3} is in T3, T6, T8, and T9. support ({I2, I3}) is 4>2, therefore, {I2, I3} is a frequent itemset.
- Infrequent Itemsets: an itemset is infrequent if itemsets' support is less than the predefined minimum support threshold value, minsupp. For example, itemset {I5, I6} is in T11. Support ({I5, I6}) is 1<2, therefore, {I5, I6} is an infrequent itemset.

3.2. Problem Statement

The major challenges encountered by the existing methods are:

1. Huge number of candidates itemset generation.
2. Multiple scans over transactional database.
3. Tedious support computations for each candidate in the candidate set.

The problem is to propose an enhanced ARM method to effectively generate frequent itemsets where it applies the construction of RPS that encompasses the current and anticipated future candidates in order to reduce unnecessary scans over the input dataset. The number of candidates for which the frequency needs to be examined is also reduced.

4. The Proposed Method

In the proposed method, the total transactions in the dataset, are bifurcated into sparse and dense regions based on buckets generated, where a TB is data structure which stores the transaction according to the transaction length. Let D be the dataset and DR and SR be the bifurcated regions of dense and sparse respectively. Also, a set called, RPS, is constructed logically from the estimated RP. RP is the mid-length of items. The RPS includes the combinations of all items of length RP. As a result, the total candidates, C_{total} includes the complete collection of RPS and the higher length candidates generated in k^{th} Pass. For instance, if the candidate set for examination of support generated in pass 1 is C_k, and the RP length candidates is *RPS*, where RPS>k. So, the total number of candidates to be examined is

$$C_{total}=|C_k|+|RPS| \quad (1)$$

The proposed method is illustrated in Figure 1.

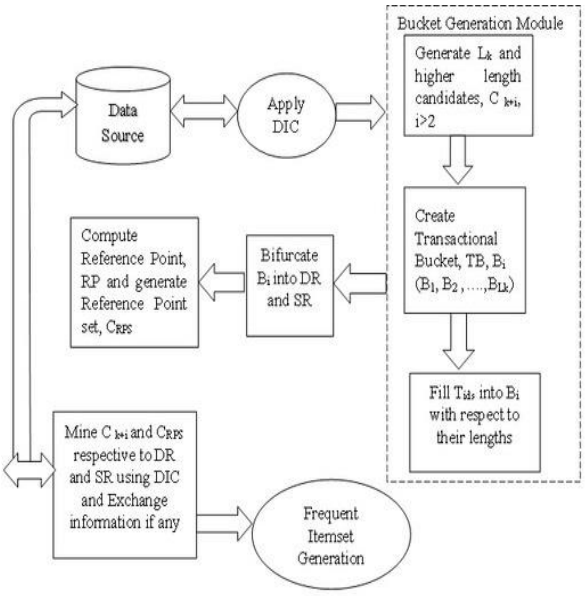

Figure 1. The proposed method.

4.1. Reference-Point- Set (RPS) Based Bi-Directional Frequent Itemset Generation

Using TB and RPS, to decrease the number of transactional dataset scans and hence the search space, the paper suggests an innovative approach. The major goal is to evaluate probable candidates during the initial database scan passes, which can cut down on the number of additional database scans that are required. Using Greedy approach, the potential candidates are accumulated in each iteration and in every stage of each pass. The proposed method includes the following steps:

1. Creation and filling of TB according to the transaction length.
2. Reference point computation from the defined item

domain and RPS generation.
3. Bifurcation of the transactions database into dense and sparse regions, using TB, respectively.
4. Parallel application of DIC on higher length candidates and reference point candidate set respective to DR and SR.

4.1.1. Transactional Bucket (TB) Creation and Filling

Initially, the data Source DS is scanned to identify L1, 1-length frequent itemsets. Next, |L1| buckets, B1, B2,……., B|L1| are created and filled with TIDs appropriately. For instance, the bucket B1 keeps TIDs of transaction having one item alone where Bn keeps TIDs of transaction having n items. Each TBi is a vector. The size of the vector differs from 0 to |L1|.

4.1.2. Reference Point Computation and Reference-Point-Set (RPS) Generation

Subsequently, RPS is coined with the reference to RP. The RP is estimated as

$$RP = \text{ceil}\left(\sum_{id=1}^{max}(B_{id} * W_{id})\right) \quad (2)$$

Where,

$$W_{id} = (B_{id} * |TB_{id}|) / \sum_{id=1}^{max}(B_{id} * TB_{id}) \quad (3)$$

W_{id} is the WeightID, the weightage of particular bucket *ID* from the total *TB* and max denotes the maximum count of transactional buckets. Bid refers to the Transactional Bucket *ID* and |*TBid*| denotes the number of transactions in the transactional bucket *ID*, Bid.

4.1.3. Bifurcation of the Transactions Database into DR and SR

From the information derived from TBs, the dataset D is logically organized into DR and SR. Here standard deviation is applied for dividing the D into DR and SR.

4.1.4. Parallel Application of DIC

Next, the typical DIC method is applied on DR, SR, and RPS for finding the frequent itemsets of higher length. The information obtained from DIC is exchanged between DR and SR to maintain consistency in the mining process. The entire mining process will be terminated when no information exists to share between DR and SR. The proposed method applies DIC as the first step to generate 1-length frequent sets, L1, which in turn provide the basis for constructing the data structure namely Transactional Buckets. The structure of the data structures is depicted in Figure 2. Once the first scan of transactional data base is completed, the RP is computed using Equation (2). For instance, for the transactional Data Source, D, the RP obtained is 3.

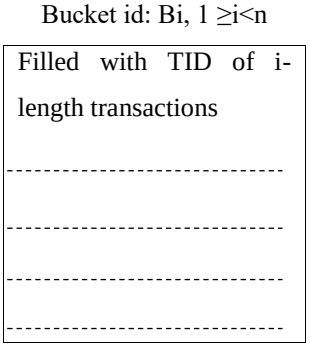

Figure 2. Transactional bucket structure.

So, for all 6 items, I1 through I6, items of length three is computed using combination formula, nCr. As a result, 120, candidate itemsets are generated. In Pass 2, according to the transactional buckets created in Pass 1, the entire data source is logically bifurcated into Dense Regions (DR) and Sparse Regions (SR). Here standard deviation is applied on the TB for division of SR and DR. Before the data source is bifurcated into regions, the transactional buckets are arranged in descending order of the transactional Bucket IDs (Bid). The details of TBs generated for the given dataset is given in the Table 2.

Table 2. Transactional buckets specifications.

Bid	Tid	Count	Region
B6		0	DR
B5		0	
B4	T8, T10	2	
B3	T1, T2, T4, T9, T12	5	
B2	T3, T5, T6, T7, T11	5	SR
B1		0	

Similar to DIC, Stopping Points (SPs) are defined on the dense and sparse regions. The typical DIC method is applied over blocks of DR, DR-SP1, DR-SP2, and so on in dense region, and so forth where each block is marked with a stopping point.

The partial higher length candidate itemsets generated during the first pass on DS and RPS, generated with the help of RP are considered for the generation of frequent itemsets. At the end of pass 2, L2 and partial higher length candidates C3 are obtained from both DR and SR regions. Simultaneously, higher length frequent itemsets are also obtained from RP computations. The entire process continues with higher passes to generate higher length itemsets until no further information exists to exchange between regions. The higher length frequent itemsets produced using RP computation can be further processed until no itemsets satisfy the minimum support threshold.

5. Results and Discussions

5.1. Experimental Setup

The experiment was run on a Windows 10 PC with an Intel(R) Core (TM) i3-8130U CPU running at

2.20GHz, 2201 MHz, with 2 cores and 4 logical processors. Python was used as the Integrated Development Environment (IDE) to implement the proposed method. Three benchmark datasets from frequent itemset mining dataset repository, known as UCI repository, used to experiment the performance of the proposed method [9], are Accidents, T1014D100K and Retail. Table 3 shows the summary of the datasets taken for performance evaluation.

Table 3. Dataset summary.

Datasets	|T|	Size	No. of attributes
Accidents	340183	33.8MB	573
T10I4D100K	100000	3.83MB	250
Retail	88162	3.97MB	200

The performance of the proposed method is verified against the performance of the benchmark methods, M-Apriori and DIC.

The performance metrices applied are time of execution measured in seconds and total number of database scans count with datasets that differs in number of transactions, their size and the number of attributes.

The study was evaluated for five different minimum support threshold values ranging from 10 to 50 for each dataset. The stopping point, M, is set to 50000, 25000, and 20000 for the datasets Accidents, T10I4D100K and Retail, respectively.

Time and scan count were used as the foundation for an analysis of how well the suggested method performed. The total number of database scans performed to complete frequent itemset creation and pattern matching of the candidate itemsets are used to calculate the scan count. The time performance analysis is measured in seconds by the time it takes to generate frequent item sets.

Table 4. Performance evaluations in terms of time and scan count for accidents.

Datasets	Minsupp	Performance of					
		Proposed Method		DIC		M-Apriori	
		Scan Count	Time (sec.)	Scan Count	Time (sec.)	Scan Count	Time (sec.)
Accidents (340183*573) M=50000	0	7	8452.37	15.5	10006.25	25	31245.53
	20	5.5	6469.27	12	8552.33	22	24568.83
	30	4	2997.19	9.5	4989.17	18.5	19457.57
	40	3.5	1715.28	8	2251.39	16	14586.24
	50	3.5	997.21	8	1587.74	15	9887.89

Table 5. Performance evaluations in terms of time and scan count for T10I4D100K.

Datasets	Minsupp	Performance of					
		Proposed Method		DIC		M-Apriori	
		Scan Count	Time (sec.)	Scan Count	Time (sec.)	Scan Count	Time (sec.)
T10I4D100K (100000*250) M=25000	10	5.5	7129.69	11	9002.65	21	15782.47
	20	5	5126.10	8.5	6732.287	18	11003.69
	30	4.5	1988.23	7.5	2446.212	15	9881.324
	40	4	957.89	8	1380.078	14.5	7881.26
	50	3	968.47	8	1087.5	14	5669.36

With various support measures, the performance of association rule mining is analyzed. It demonstrates that there has been a considerable decrease in the number of scans across the three data sources. Similarly, there is considerable amount of reduction in time taken by proposed method when compared to DIC and M-Apriori. Table 4, 5, and 6 provide the performance evaluations in terms of time and scan count for three datasets, Accidents, T10I4D100K and Retail, respectively.

Table 4 shows that when the support count rose from 10 to 50, the scan count significantly fell from 7 to 3.5, and the execution time decreased from 8452.37 to 997.21 for the dataset Accident with a higher number of transactions, attributes, and stopping points. Experimental evidence shows that the scan count in DIC is 60% higher and the scan count in M-Apriori is four times higher than the suggested technique for all support count ranges. In terms of execution time in seconds, DIC and M-Apriori, on average, show increases of 15% and four times higher, respectively, over the proposed method.

From Table 5, for dataset T10I4D100K with a smaller number of transactions, attributes and stopping point count than Accidents, the scan count decreased from 5.5 to 3 and execution time also decreased from 7129.69 to 968.47 for increasing support count ranges. Comparing the scan count of proposed method, DIC shows an increase of 42% and was four times greater for M-Apriori. For execution time expressed in seconds, DIC shows a 9 % increase than proposed method, on an average and M-Apriori shows an execution time of three times increase than proposed method.

From Table 6, for dataset Retail with a smaller number of transactions, attributes and stopping point count than T10I4D100K, the scan count decreased from 5 to 3 and execution time also decreased from 10140.2 to 981.344 for increasing support count ranges. Comparing the scan count of proposed method, DIC shows an increase of 46% and was four times greater in M-Apriori.

Table 6. Performance evaluations in terms of time and scan count for retail.

Datasets	Minsupp	Performance of					
		Proposed Method		DIC		M-Apriori	
		Scan Count	Time (sec.)	Scan Count	Time (sec.)	Scan Count	Time (sec.)
Retail (88162*200) M=20000	10	5	10140.2	10.5	10569.8	20.5	16258.23
	20	4	2998.2	8	3997.20	17	13558.37
	30	3.5	1128.69	8	1512.74	16.5	10008.82
	40	3	1001.58	7	1299.04	15	8479.19
	50	3	981.344	7	1059.30	14.5	8002.96

For execution time expressed in seconds, DIC shows a 5 % increase than proposed method on an average and M-Apriori shows an execution time of three times increase than proposed method.

Figures 3, 4, and 5 for the datasets Accidents, T10I4D100K, and Retail, respectively, show visually how the proposed technique performed against DIC and M-Apriori. Based on support in percent against time in seconds and scan count for each dataset, performance is assessed. The aforementioned graphs show how M-Apriori, DIC, and the proposed method performed. The proposed method performs better than DIC and M-Apriori because there are fewer candidate generations and earlier mining process convergence.

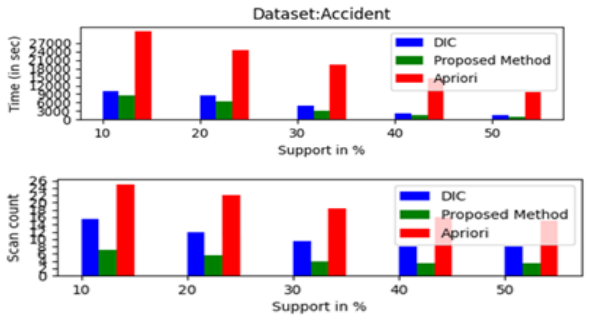

Figure 3. Performance evaluation on data source accident.

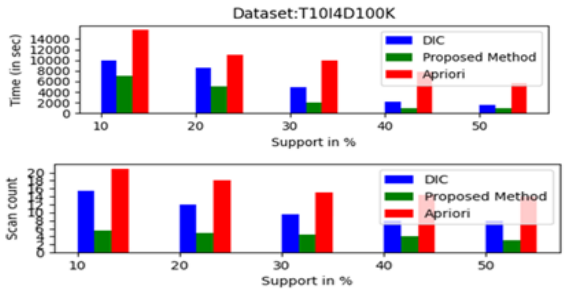

Figure 4. Performance evaluation on data source T1014D100K.

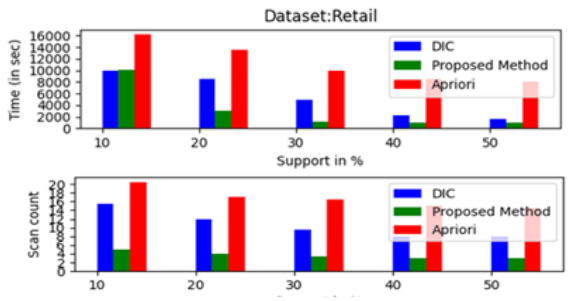

Figure 5. Performance evaluation on data source Retail.

The main contributions that lead to the proposed method's reduced scan count when compared to DIC and M-Apriori are the bucket creation, reference point computation, and parallel application of DIC on DR and SR. Additionally, the earlier generation of RPS make it simpler to compute frequent higher length itemsets more quickly. The bifurcated DR and SR regions provide a clear perspective of mining since it uses a Pincer-like idea [14] to mine candidates from either end.

6. Conclusions

Strong-ARM techniques are introduced with a common objective of generating frequent itemsets from large datasets with minimum complexities. They enhance the performance by reducing the search space in terms of reduction in number of dataset scans required and reduction in number of candidates as well. In the proposed method, a tradeoff between the time complexity and the space complexity is attempted and demonstrated. The proposed method, using TB and RPS, increases the mining efficiency by reducing the search space with the help of simple data structure such as TB. In turn, the space requirements are also reduced to a larger extend by RP computation that could produce higher length potential candidates in the early passes of the mining process. Experimental evidence shows that the average scan count in DIC and M-Apriori is respectively, 24% and 65%, higher than that of the proposed technique for different support counts. In terms of execution time in seconds, DIC on average shows a 10% increase while M-Apriori displays an execution time that is three times higher than that of the proposed technique.

The experimental results reveal the potential of the proposed approach through considerable reduction of scan count and time complexity. However, in the real-world dataset, with an increase in the item domain of the transactional database, the generation of frequent itemset out of numerous potential candidates may lead to memory space consumption. As a result, the proposed method is recommended for in-depth research for optimally utilizing the memory space while examining very large datasets.

Furthermore, in the future, the research could be extended in applying the proposed method on incremental mining whenever there is a need to update the dataset incrementally.

References

[1] Agrawal R., Imieliński T., and Swami A., "Mining Association Rules Between Sets of Items in Large Databases," *in Proceedings of the ACM SIGMOD International Conference on Management of Data*, Washington, pp. 207-216, 1993.

[2] Agrawal R. and Srikant R., "Fast Algorithms for Mining Association Rules," *in Proceedings of the 20th International Conference on Very Large Data Bases*, Chile, pp. 487-499, 1994. https://dl.acm.org/doi/10.5555/645920.672836

[3] Bagui S. and Stanley P., "Mining Frequent Itemsets from Streaming Transaction Data Using Genetic Algorithms," *Journal of Big Data*, vol. 54, no. 7, 2020. https://doi.org/10.1186/s40537-020-00330-9

[4] Brin S., Motwani R., Ullman J., and Tsur S., "Dynamic Itemset Counting and Implication Rules for Market Basket Data," *in Proceedings of the ACM SIGMOD International Conference on Management of Data*, Tucson, Arizona USA, pp. 255-264, 1997.

[5] Cai S., Hao S., Sun R., and Wu G., "Mining Recent Maximal Frequent Itemsets over Data Streams with Sliding Window," *The International Arab Journal of Information Technology*, vol. 16, no. 6, pp. 961-969, 2019. https://www.iajit.org/PDF/November%202019, %20No.%206/15400.pdf

[6] Ceglar A. and Roddick J., "Association Mining," *ACM Computing Surveys*, vol. 38, no. 2, pp. 1-42, 2006. https://doi.org/10.1145/1132956.1132958

[7] Chen J. and Xiao K., "BISC: A Bitmap Itemset Support Counting Approach for Efficient Frequent Itemset Mining," *ACM Transactions on Knowledge Discovery from data*, vol. 4, no. 3, pp. 1-37 2010. https://doi.org/10.1145/1839490.1839493

[8] Fujioka K. and Shirahama K., "Generic Itemset Mining Based on Reinforcement Learning," *IEEE Access*, vol. 10, pp. 5824-5841, 2022. https://doi.org/10.48550/arXiv.2105.07753

[9] Goethals B., "Frequent Itemset Mining Implementations Repository," http://fimi.uantwerpen.be/data, Last Visited, 2023.

[10] Hamilton H., "Dynamic Itemset Counting and Implication Rules for Market Basket Data" http://www2..uregina.ca/~dbd/cs831/notes/itemsets/DIC.html, Last Visited, 2023.

[11] Han J., Pei J., and Kamber M., *Data Mining: Concepts and Techniques*, Morgan Kaufmann, 2011. https://www.sciencedirect.com/book/9780123814791/data-mining-concepts-and-techniques.

[12] Han J., Pei J., Yin Y., and Mao R., "Mining Frequent Patterns without Candidate Generation: a Frequent-Pattern Tree Approach," *Data Mining and Knowledge Discovery*, vol. 8, no. 1, pp. 53-87, 2004. https://doi.org/10.1023/B:DAMI.0000005258.31418.83

[13] Leeuwen M. and Galbrun E., "Association Discovery in Two-View Data," *IEEE Transactions on Knowledge and Data Engineering*, vol. 27, no. 12, pp. 3190-3202, 2015. 10.1109/TKDE.2015.2453159

[14] Lin D. and Kedem Z., "Pincer-Search: An Efficient Algorithm for Discovering the Maximum Frequent Set," *IEEE Transactions on Knowledge and Data Engineering*, vol. 14, no. 3, pp. 553-566, 2002. doi: 10.1109/TKDE.2002.1000342

[15] Li F., Meng C., Wang C., and Fan S., "Equipment Quality Information Mining Method Based on Improved Apriori Algorithm," *Journal of Sensors*, vol. 2023, 2023. https://doi.org/10.1155/2023/2155590

[16] Magdy M., Ghaleb F., Mohamed D., and Zakaria W., "CC-IFIM: An Efficient Approach for Incremental Frequent Itemset Mining Based on Closed Candidates," *The Journal of Supercomputing*, pp. 7877-7899, 2023. https://doi.org/10.1007/s11227-022-04976-5

[17] Maolegi M. and Arkok B., "An Improved Apriori Algorithm for Association Rules," *International Journal on Natural Language Computing*, vol. 3, no. 1, 2014. https://doi.org/10.48550/arXiv.1403.3948

[18] Park J., Chen, M., and Yu P., "An Effective Hash Based Algorithm for Mining Association Rules," *ACM SIGMOD Record*, vol. 24, no. 2, pp. 175-186, 1995. DOI:10.1145/568271.223813

[19] Phan H. and Le B., "A Novel Algorithm for Frequent Itemsets Mining in Transactional Databases," *Trends and Applications in Knowledge Discovery and Data Mining*, vol. 11154, pp. 243-255, 2018. DOI:10.1007/978-3-319-95786-9_21

[20] Savasere A., Omiecinski E., and Navathe S., "An Efficient Algorithm for Mining Association Rules in Large Databases," *in Proceedings of the 21st International Conference on Very Large Data Bases*, San Francisco, pp. 432-443, 1995.

[21] Song W., Yang B., and Xu Z., "Index-BitTableFI: An Improved Algorithm for Mining Frequent Itemsets," *Knowledge Based Systems*, vol. 21, no. 6, pp. 507-513, 2008. https://doi.org/10.1016/j.knosys.2008.03.011

[22] Thurachon W. and Kreesuradej W., "Incremental Association Rule Mining with a Fast Incremental Updating Frequent Pattern Growth Algorithm," *IEEE Access*, vol. 9, pp. 55726-55741, 2021. 10.1109/ACCESS.2021.3071777

[23] Toivonen H., "Sampling Large Databases for Association Rules," *in Proceedings of the 22nd International Conference on Very Large Data*

Bases, San Francisco, pp. 134-145, 1996. https://dl.acm.org/doi/10.5555/645922.673325

[24] Wang L., Cheung D., Cheng R., Lee S., and Yang X., "Efficient Mining of Frequent Itemsets on Large Uncertain Databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 24, no. 12, pp. 2170-2183, 2012. DOI: 10.1109/TKDE.2011.165.

[25] Webb G. and Vreeken J., "Efficient Discovery of the Most Interesting Associations," *ACM Transactions on Knowledge Discovery from Data*, vol. 8, no. 3, pp. 1-31, 2014. https://doi.org/10.1145/2601433

[26] Zaki M., "Scalable Algorithms for Association Mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 12, no. 3, pp. 372-390, 2000. DOI: 10.1109/69.846291

[27] Zhang C., Tian P., Zhang X., Liao Q., and Jiang Z., "HashEclat: An Efficient Frequent Itemset Algorithm," *International Journal of Machine Learn and Cyber*, vol. 10, pp. 3003-3016, 2019. https://doi.org/10.1007/s13042-018-00918-x

[28] Zhao Z., Zhou J., Gabu G., Alroobaea R., and Masud M., "An Improved Association Rule Mining Algorithm for Large Data," *Journal of Intelligent Systems*, vol. 30, no. 1, pp. 750-762, 2021. DOI:10.1515/jisys-2020-0121

Ambily Balaram is currently pursuing her Ph.D degree in the area of Data Mining, in the department of Computer Science and Engineering at Coimbatore Institute of Technology. She received the B.Tech and M.Tech degrees in Computer Science and Engineering from Government Engineering College, Kannur University and KMCT College of Engineering, Calicut University respectively. She has more than 5 years of teaching experience in professional colleges. Her research interests include Data Mining and Machine Learning.

Nedunchezhian Raju, PhD, is a Professor in the Department of Computer Science and Engineering, Coimbatore Institute of Technology, Coimbatore, India. He has served in various capacities as Head of the Department, Vice Principal, Head of the Institution, and Director of Research. His research interests include data data analytics and machine learning. He has more than 25 years of experience in teaching and research. He has guided 20 PhD scholars, and six more PhD scholars are currently doing their research under his supervision. To his credit, he has published more than 100 papers in refereed journals and international conferences. He has published a few books and book chapters also. He did his PhD, ME, and BE degrees all in computer science and engineering.