# AN APRIORI BASED ALGORITHM TO MINE ASSOCIATION RULES WITH INTER ITEMSET DISTANCE

Pankaj Kumar Deva Sarma[1] and Anjana Kakoti Mahanta[2]

[1]Associate Professor, Department of Computer Science,
Assam University, Silchar, Assam, India, PIN-780011
[2]Professor, Department of Computer Science,
Gauhati University, Guwahati, Assam, India, PIN- 781014

***ABSTRACT***

*Association rules discovered from transaction databases can be large in number. Reduction of association rules is an issue in recent times. Conventionally by varying support and confidence number of rules can be increased and decreased. By combining additional constraint with support number of frequent itemsets can be reduced and it leads to generation of less number of rules. Average inter itemset distance(IID) or Spread, which is the intervening separation of itemsets in the transactions has been used as a measure of interestingness for association rules with a view to reduce the number of association rules. In this paper by using average Inter Itemset Distance a complete algorithm based on the apriori is designed and implemented with a view to reduce the number of frequent itemsets and the association rules and also to find the distribution pattern of the association rules in terms of the number of transactions of non occurrences of the frequent itemsets. Further the apriori algorithm is also implemented and results are compared. The theoretical concepts related to inter itemset distance are also put forward.*

***KEYWORDS***

*Association rules, frequent itemsets, support, confidence, inter itemset distance, spread, data mining.*

## 1. INTRODUCTION

Association rule mining, put forward by *Agrawal, Imielinsky & Swami* **[1]** is a technique for rule discovery from frequent patterns. Let $I = \{i_1, i_2, i_3, \ldots \ldots \ldots i_m\}$ be a set of literals called items or attributes over the binary domain {0,1} and D be a database of transactions, where each transaction T is a set of items such that $T \subseteq I$. A set of items $X \subseteq I$ is called an *itemset*. Given an itemset $X \subseteq I$, a transaction T contains X if and only if $X \subseteq T$. Each transaction T is a tuple of the database D and is represented by identifying the attributes with value 1. A unique identifier called TID is associated with each transaction. Thus a transaction T contains an itemset X if $X \subseteq T$.

An association rule is an implication of the form X => Y, where X and Y are itemsets such that $X \subseteq I$, $Y \subseteq I$ and $X \cap Y = \Phi$. The rule X=>Y holds in the transaction database D with confidence c if

DOI : 10.5121/ijdkp.2013.3605 

c% of the transactions in D that contain X also contain Y. The rule X=>Y has support s in the transaction database D if s% of transactions in D contains X UY. The support of the rule is denoted by $\sigma(X \cup Y)$ and its confidence is given by $\sigma(X \cup Y)/\sigma(X)$. In other words confidence of the association rule X=>Y is the conditional probability that a transaction contains Y, given that it contains X. A rule is said to be confident if its confidence is not less than user specified minimum confidence (*minconf)*. Confidence denotes the strength of implication and support indicates the frequencies of the occurring patterns in the rule. In this definition of association rules, negation or missing items are not considered to be of interest. Given a set of transactions D, it is required to generate all association rules that satisfy certain user specified minimum thresholds for support (*minsup)* and confidence (*minconf)*. An itemset with k-items is called a *k-itemset*. Support count, denoted by σ(X) is the frequency of occurrence of an itemset X in the transactions of the database. Support of an itemset X is also expressed as percentage of transactions in the database D.

An itemset X is said to be *large* or *frequent* if its support is not less than *minsup*. That is an itemset is called *frequent* if it occurs at least in some pre specified number of transactions of the database called *minimum support (minsup)*. A frequent itemset is said to be a **maximal frequent itemset** if it is a frequent itemset and no superset of this is a frequent itemset. Discovery of frequent itemsets from large transaction database has been a central component for mining association rules [2] [3]. Apriori [2] is one of the prominent algorithms for mining association rules. The task of mining association rules consists of two sub problems [1] [4]:

**Step 1:** Find all the large or frequent itemsets with respect to a pre specified minimum support. This step is computationally and I/O intensive. Given m items, there can be potentially $2^m$ itemsets, m $\epsilon$ Integer and m > 0. Efficient methods are needed to traverse this exponential itemset search space to enumerate all the frequent itemsets.

**Step 2:** Generate the association rules which are confident from the large itemsets discovered. This step is relatively straight forward. Rules of the form X\Y=>Y are generated for all the frequent itemsets X, where Y $\subseteq$ X subject to the condition that the rules have at least minimum confidence.

The performance of the mining association rules is determined by the efficiency of the method to solve the first problem in step 1 [2]. Research issues related to association rules include measures of interestingness and reduction of huge number of discovered rules. *Support* and *confidence* are two widely used measures of interestingness. Other parameters include *correlation* (or lift or interest) and *conviction* [5]**.**

With *average inter itemset distance* association rules are reinforced with additional meaning. The constraints support, confidence and average inter itemset distance are used conjunctively to reduce the number of association rules. In this paper a detail algorithm based on apriori algorithm is designed and implemented to calculate average inter itemset distance and discover the association rules with thresholds on average inter itemset distance along with support and confidence without scanning the database further. The apriori algorithm is also implemented and the results of both the methods are compared. Moreover, the necessary theoretical foundation for the calculation of average inter itemset distance is also put forward. The proposed algorithm applies a level wise approach of scanning like the apriori algorithm in which frequent n – itemsets become the seeds for generating the candidates for the next i. e. $(n+1)^{th}$ pass of the database and so on until no higher order itemset are found.

### 1.1 Inter Itemset Distance (IID)

*Inter itemset distance (IID) of an itemset* is the length of separation or gap within the *lifespan of occurrence of an itemset* in terms of the number of intervening transactions of non occurrence of the itemset between two successive occurrences of the same itemset in the transactions of the database. For an itemset with a solitary occurrence in the whole database, inter itemset distance cannot be defined. The minimum value of inter itemset distance of an itemset is zero if the itemset has occurred in two consecutive transactions and otherwise it is non zero. Since the occurrences of an itemset in the transactions of the database is random therefore, the lengths of various gaps between occurrences (i.e. the inter itemset distances) of an itemset are not identical between every pair of its occurrences in its lifespan. Therefore, the total inter itemset distance of an itemset is calculated. The average inter itemset distance of an itemset is calculated by dividing the total inter itemset distance of an itemset by the number of gaps of non occurrences of the itemset in its lifespan.

**Average Inter Itemset Distance (IID) or Spread** indicates how closely or sparsely an itemset occurs in the database within its lifespan. It gives insight about the distribution pattern of occurrence of an itemset across the database. Frequent itemset discovery algorithms have not considered this and the lifespan of occurrence of an itemset while counting support. The rules are generated without any information about such pattern of occurrences. Based on preliminary level of experimentation, this approach helped to reduce the number of association rules but a detail algorithm is not incorporated in **[6]**. Itemsets with identical support may not have the same Average Inter Itemset Distance. Together with support, Average Inter Itemset Distance or Spread is used as another measure for quality for association rules. The smaller the threshold for Average Inter Itemset Distance the closer will be the spacing between the successive occurrences of an itemset.

### 1.2 Outline of the Paper

The paper is divided into five sections excluding the introduction section in which the concept of association rule mining and average inter itemset distance is presented. In the "Related Works" section leading algorithms for association rule mining particularly the apriori algorithm and its improvisations are presented. Further, various recent works and techniques concerning reduction of association rules are studied and discussed. In section, "Theoretical Formulation of Inter Itemset Distance" the concepts related to average inter itemset distance or spread are defined and their mathematical formalisms are presented. The minimum and the maximum limits of the Average Inter itemset Distance (IID) are derived in this section. In the next section the detail algorithm is presented and explained for the mining association rules with average inter itemset distance along with support and confidence. In Implementation section, the results of the implementation of the proposed algorithm and the apriori algorithm are described and compared. The paper ends with concluding remarks.

## 2. RELATED WORKS

The **Apriori** algorithm **[2]** proceeds in a level wise manner in the itemset lattice with a candidate generation technique in which only the frequent itemsets found at a level are used to construct candidate itemsets for the next level. Over the last decade and a half many improvisations and incremental development of the apriori algorithm including parallel algorithms have been

reported in the literature with a view to improve the computational performance. The **partition algorithm [7]** is one such algorithm which minimizes the database scans to only ***two*** by partitioning the database into small partitions so that each can be accommodated in the main memory. In the **FP-growth algorithm [8]** a pattern-growth approach for mining frequent itemsets without candidate generation was proposed. Tan, Kumar, & Srivastava discussed about selecting right interesting measures for association rules in **[9]**. Hilderman & Hamilton described various interesting measures in **[10]**. Alternative interesting measures for mining association rules are proposed in **[11]**. Various quality measures of data mining and related issues are discussed in [12].

Reduction of huge number of discovered association rules has been undertaken in various recent works. The number of association rules is reduced by two standard methods namely by increasing the minimum confidence parameter and by increasing the minimum antecedent support parameter. Recent research focused on finding other methods to reduce the rules either by adjusting the rule induction algorithm or by pruning the rule set **[13] [14] [15].** In **[16]**, a technique for removing overlapping rules for reducing the size of the rule sets with an extension of the apriori algorithm is presented.

***Vo & Le*** **[17]**, proposed a method for mining minimal non – redundant association rules. A concept of δ – tolerance ARs is used to eliminate redundancy in the set of association rules and to obtain concise mining results *in* **[18]**. In [19] a concise representation called reliable approximate basis for representing non redundant association rules is given for the purpose of reducing the number of association rules. Use of visualization techniques is discussed in **[20]** to deal with large number of discovered association rules. The notion of goodness of a rule set is quantified for reduction in the size of the association rule collection *in* **[21]**. In **[22]** an improved apriori algorithm is proposed to minimize the number of candidate sets based on evaluation of quantitative information associated with each attribute in a transaction. Occurrence of false positives and errors in pattern of enumeration of a large number of association rules are controlled through multiple testing correlation approaches in **[23]**. An approach to prune and filter enormous number of discovered association rules is proposed based on ontological relational weights by Radhika & Vidya in **[24]**. In **[25]** the problem of usefulness of association rules due to huge number of discovered association rules is analyzed and an interactive approach is proposed to prune and filter discovered rules using ontologies. A two step pruning method is proposed in **[26]** for reducing uninteresting spatial association rules.

Techniques for mining compressed sets of frequent patterns are developed to reduce the huge set of frequent patterns generated. Mining closed patterns can be viewed as lossless compression of frequent patterns. Lossy compression of patterns includes studies of top-*k* patterns **[27]**. In **[28]** *k* – itemsets are used to cover a collection of frequent itemsets. A profile based approach **[29]** and a clustering-based approach **[30]** are proposed for frequent itemset compression. Bayesian network is used to identify subjectively interesting patterns **[30]**. In **[32]** association rules are mined with non uniform minimum support. Method for selective generation of association rules is developed in **[33].**

**A** class of tough constraints called Loose anti – monotone constraints is introduced as a super class of convertible anti monotone constraints and used in a level wise apriori like computation by means of a data reduction technique. **[34]**

# 3. THEORETICAL FORMULATION FOR INTER ITEMSET DISTANCE

In this section, a theoretical formulation for mining association rules with average inter itemset distance or spread is developed.

## 3.1 Basic Definitions

Some basic definitions in connection with average inter itemset distance for frequent itemsets are given below.

**Definition1: Inter Itemset Distance (IID) of an itemset:** The Inter Itemset Distance (IID) or Spread of an itemset is defined as the number of intervening transactions in which the itemset is not present between two successive occurrences of the itemset.

**Definition2: Total Inter Itemset Distance (IID) of an itemset:** The Total Inter Itemset Distance (IID) of an itemset among all successive occurrences of an itemset is defined as the sum of the number of all the intervening transactions in which the itemset is not present between every two successive occurrences of the itemset within the lifespan of the itemset.

**Definition3: Lifespan of an itemset ($l_s$):** The life span ($l_s$) of an itemset is the number of transactions, starting with the first occurrence ($TID_{first}$) to the last occurrence of the itemset ($TID_{last}$) (inclusive of both the transactions of first and the last occurrences). It is assumed that the TIDs are numbered serially without any break. Thus,

$$l_s = TID_{last} - TID_{first} + 1$$
$$\text{i.e. } l_s = n_l - n_i + 1,\ n_l > n_i \qquad (1)$$

Where for an itemset, $n_i$ is the TID of its first occurrence and $n_l$ is the TID of its last occurrence.

**Ex.** Let for an itemset $n_i = 51$, $n_l = 87$
then, $l_s = 87 - 51 + 1 = 37$**.**

**Definition4: Average Inter Itemset Distance (IID) or Spread of an itemset:** The **Average Inter Itemset Distance (IID) or Spread of an itemset** among all successive occurrences of an itemset is defined as the sum of the number of all the intervening transactions in which the itemset is not present between every successive occurrences of the itemset within the lifespan of the itemset divided by the number of gaps of non occurrences of the itemset.

**Definition5: Gap of non occurrences of an itemset: A gap** of non occurrence consists of all the transactions in which the itemset has not occurred between any two of its successive occurrences.

The **length of a gap** of non occurrence of an itemset is 0 (zero) when the itemset has occurred in two consecutive transactions and it is non zero when there is at least one transaction in which the itemset has not occurred between two successive occurrences of the itemset in its life span. **A** gap of non occurrence cannot be defined for an itemset whose support count is either 0 or 1. The total number of gaps of non occurrences of an itemset is one less than the support count of the itemset.

## 3.2 Range of Values for Average Inter Itemset Distance (IID) or Spread of an Itemset

**Let n =** |D| be the number of transactions in D and σ, the Specified support threshold in percentage. The support count (c) of an itemset is converted to percentage support (σ) by doing (c/|D|) * 100 i.e. (c/n) * 100. Therefore, range for percentage support (σ) is $0 \leq \sigma \leq 100$

Similarly the range of Average IID or spread of an itemset X, denoted by **Average IID (X) or spread (X)** is given by

**$0 \leq$ Average IID (X) $\leq$ (n – nσ)/(nσ – 1)** **(2)**

The **lower limit** on average IID or spread of an itemset is zero.

i. e. **(average IID or spread)$_{min}$ = 0** **(3)**

This happens when all the occurrences of an itemset are in consecutive transactions in its life span. The **upper limit** on average IID or spread of an itemset is defined based on the maximum life span and the input percentage support threshold (σ). In a transaction database of size | D| = n, the maximum lifespan of an item set is from TID = 1 to TID=n.Thus

$$(\textbf{average IID or spread})_{max} = \frac{(\text{Total number of transactions in D i.e. } |D| - \text{Input support count threshold i.e. } \sigma|D|)}{(\text{Input support count threshold (i.e. } \sigma|D|) - 1)}$$

$$\text{i.e. } (\textbf{average IID or spread})_{max} = \frac{(n - n\sigma)}{(n\sigma - 1)} \quad \textbf{(4)}$$

**The range of average IID for an itemset with input percentage support threshold σ is [0, (n – nσ)/(nσ – 1)].**

The range for an average IID is specifiable at the beginning of the algorithm and based on this, the user input threshold for average IID or spread can be specified. At first all the frequent 1-itemsets having support higher than the input support threshold and which are also closely spaced are discovered.

**Case1.** For constant | D|=n as **σ** increases, **(n – nσ) decreases** much **faster in comparison** to **(nσ – 1).** Thus for higher value of percentage support **(σ), (nσ – 1) is** approximately equal to **nσ.**

**i.e. (average IID or spread)$_{max}$**
**= (n – nσ)/nσ = (1/σ) – 1** **(5)**

**Ex.1. If σ = 50%** then **(average IID or spread)$_{max}$ = (1/50%) – 1 = 1.** This is true, since σ = 50% **means** average IID = 1 always under the full life span of an itemset. (i.e. full life span = n). If calculated exclusively, then for σ = 50%, **(average IID or spread)$_{max}$ = (n – nσ)/(nσ – 1) =** 1/ (1– 2/n). **If n** is large, then 2/n → 0 and hence **(average IID or spread)$_{max}$** = 1. Thus the proposition is consistent. This proposition is based on the fact that if an itemset has support = 50%, then the itemset is not present in 50% of the transactions. Calculating exclusively for such an itemset, we get the same result. (Average IID or spread)$_{max}$ = **(n – nσ)/(nσ – 1)** = (n – n/2)/ (n/2 – 1) = (n/2)/ (n/2 – 1). For n = 100, (average IID/average spread)$_{max}$ = (100/2)/ (100/2 – 1) = 1.024 ≈ 1.

**Ex.2. If** σ = 0%, i.e. when there is no presence of an itemset in the database, then (average IID or spread)$_{max}$ = **(n – nσ)/(nσ – 1)** = –n. This is undefined since –n does not have any significance and hence the concept of inter itemset distance IID does not arise.

**Ex.3. If** σ = 100% [σ = 100% =1], that is when the itemset is present in all the transactions of the database, then the value of (average IID or spread)$_{max}$ is zero since there is no gap among the occurrences of the itemset. Thus (average IID or spread)$_{max}$ = **(n – nσ)/(nσ – 1)** = (n – n)/ (n – 1) = 0.

**Average IID/Average spread in terms of lifespan of an itemset:** For an itemset with given percentage support (**σ**) in a database of n transactions

**Average IID** or **spread**
= ((| $TID_{last}$| - | $TID_{first}$| +1) – nσ)/( nσ -1)
= (LifeSpan – nσ)/( nσ -1)
**= ($l_s$ – nσ)/( nσ -1)** **(6)**

Here, nσ is the support count.

**Ex.4**. Let $n_l$ = 67, $n_i$ = 11 and σ = 40%, n = 100. Average IID or spread = ($l_s$ – nσ)/(nσ -1) = 17/39 = 0.45.

## 3.3 Closely Spaced – n Itemsets and Closely Spaced Frequent or Large – n Itemsets

The itemsets which satisfy input threshold for both support and Average Inter Itemset Distance or Spread are called ***Closely Spaced Frequent Itemsets (CSFI) or Closely Spaced Large Itemsets (CSLI)***. The itemsets of cardinality n which satisfy the input threshold for Average Inter Itemset Distance or Spread are called **closely spaced n - itemsets**. For Closely spaced n – itemsets the average inter itemset distance is less than or equal to the specified threshold.

**Definition 6: Closely Spaced –n Itemset**:

Let I = {$i_1$, $i_2$, $i_3$, … …. …. $i_m$} be a set of literals called items and D be a database of transactions, where each transaction T is a set of items such that T ⊆I. Given an itemset X ⊆I, a transaction T contains X if and only if X⊆T. An itemset X of cardinality n (n is an integer) is said to be closely spaced n – itemset in its lifespan if its average IID or spread is less than or equal to the user specified threshold value for maximum average IID or spread (d). Thus for a closely spaced n – itemset

**Average IID (X) or Spread (X) ≤ d** **(7)**

Where, d is the user specified threshold value for maximum average IID or spread for an itemset.

**Definition 7: Closely Spaced Frequent or Large n – Itemset**:

Let I = {$i_1$, $i_2$, $i_3$, … …. …. $i_m$ } be a set of literals called items and D be a database of transactions, where each transaction T is a set of items such that T ⊆I. Given an itemset X ⊆I, a transaction T contains X if and only if X⊆T. An itemset X of cardinality n (n is an integer) is said to be *Closely Spaced Frequent or Large n – Itemset* in its lifespan if it is closely spaced with respect to specified threshold for average inter itemset distance or spread and also frequent or

large at the same time with respect to specified threshold for support. Thus for a Closely Spaced Frequent or Large n – Itemset X

$$\textbf{Supp(X)} \geq \boldsymbol{\sigma}$$
$$\textbf{and}$$
$$\textbf{Average IID or Spread (X)} \leq \textbf{d} \qquad \textbf{(8)}$$

Where, σ is the threshold for minimum support and d is the threshold value for maximum average IID or spread for the itemset X. The range of d is given by (2) above.

When average inter itemset distance or spread is used as a measure of interestingness along with support for the discovery of closely spaced frequent itemsets, then threshold values must be provided as input for both the parameters. An itemset X of the database D may be closely spaced based on the user specified value of average inter itemset distance d without being frequent. Such closely spaced itemsets has other significance. Such an itemset though not frequent in the context of the whole database but occurs due to sudden event related to the concerned domain of the database. However, this problem needs to be studied differently. Discovering all closely spaced itemsets along with their average inter itemset distances and all the closely spaced frequent itemsets along with their supports and average inter itemset distances is a non trivial problem if the cardinality of I, the set of all the items of the database of transactions D is large. The problem is to identify which of the subsets of I are frequent and closely spaced.

## 3.4 Calculation of Average Inter itemset Distance (IID) or Spread

Average Inter itemset Distance (IID) or Spread of an itemset is the average separation of the occurrences of the same itemset in its lifespan. If an itemset occurs in consecutive transactions then for each such pair of occurrence the length of the gap is zero. Thus each occurrence of every itemset in the database has to be kept track off and their separation in terms of the number of intervening transactions has to be calculated and stored and the same has to be progressively updated till the end of scanning the last transaction in the database. Thus,

$$\text{Average Inter Itemset Distance or spread (d)} = \frac{\text{Sum of the lengths of all the gaps of occurrences of an itemset within its lifespan in terms of the number of transactions of non occurrence}}{(\text{Support of the itemset} - 1)}$$

$$\textit{i.e.} \quad \mathbf{d} = \frac{\sum_{i=1}^{m-1} d_{i,i+1}}{(s-1)} \qquad \textbf{(9)}$$

Where, m is the TID of the transaction in which the itemset has its last appearance. The apriori algorithm is modified for the calculation of d and $d_{i,\ i+1}$. The lifespan of a frequent and closely spaced itemset is discovered without making any additional scan of the database. It is also observed that itemsets with same support and same size does not necessarily have the same average inter itemset distance or spread.

# 4. AN ALGORITHM FOR MINING ASSOCIATION RULES WITH AVERAGE INTER ITEMSET DISTANCE

Mining association rules with average inter itemset distance, support and confidence further refines the association rules discovered with support and confidence. An algorithm is designed based on the level wise approach of the apriori algorithm and is described below. We call the association rules which satisfy the pre specified values of support, confidence and average inter itemset distance as the ***closely spaced association rules*** to distinguish them from the conventional association rules.

## 4.1 Problem Decomposition

The problem of mining association rules with average inter itemset distance, support and confidence is decomposed into three broad steps:

**(i) Step 1:** Find all the frequent itemsets having support greater than or equal to the user specified minimum support threshold σ.

**(ii) Step 2:** Find the average inter itemset distance or spread (d) for each of the frequent itemsets discovered in step 1.

The actions of these two steps are performed in the same pass of the algorithm for each scan of the database. This process is continued till all the frequent n – itemsets and all the closely spaced frequent n – itemsets are discovered. This takes n scans over the database, the same as the number of scans in the apriori algorithm. The frequent n – itemsets and closely spaced n – itemsets are stored along with their support and average inter itemset distance.

**(iii) Step 3:** Use the frequent and closely spaced itemsets to generate the association rules with respect to the pre specified threshold values. An algorithm by modifying the apriori algorithm is proposed below.

## 4.2 Proposed Algorithm

Based on the above problem decomposition, the proposed algorithm has the following segments:

**(i)** Mining Closely Spaced Large – 1 Itemsets ($SL_1$).
**(ii)** Mining Closely Spaced Large – k Itemsets ($SL_k$).
**(iii)** Generating Candidate k – Itemsets ($SC_k$) from the large (k – 1) - itemsets ($L_{k-1}$) discovered in every previous pass using the function Generate Candidate Itemsets ($L_{k-1}$).
**(iv)** Prune Candidate k – Itemsets ($SC_k$)
**(v)** Mining Closely Spaced Large Itemsets (SL) of all the sizes.
**(vi)** Generate closely spaced association rules from closely spaced large Itemsets (SL).
These segments are presented below.

**(i) Mining Closely Spaced Large – 1 Itemsets ($SL_1$): Modified Algorithm to compute the Large 1-Itemsets ($L_1$) and Closely Spaced 1-Itemsets ($S_1$) and then to compute Closely Spaced Large 1 – Itemsets ($CSLI_1$) by the operation $L_1 \cap S_1$.**

Let us denote Closely Spaced Large 1 – Itemsets ($CSLI_1$) by $SL_1$ and Closely Spaced Large k – Itemsets ($CSLI_k$) by $SL_k$. Thus $SL_1 = L_1 \cap S_1$ and $SL_k = L_k \cap S_k$. In the following, a method to scan the database to count the support (σ) and the *Inter Itemset Distance* (IID) denoted by *d* for the Closely Spaced Candidate 1 – Itemsets is given. By using the specified input values for support and *Inter Itemset Distance* in the method the Large 1 – Itemsets ($L_1$) and the Closely Spaced 1 – itemsets ($S_1$) are found. Now, the Closely Spaced Large 1 – Itemsets ($CSLI_1$) denoted by $SL_1$ are found by using $SL_1 = L_1 \cap S_1$ (In general, $SL_n = L_n \cap S_n$). **Let** $C_1$ = Candidate 1 – itemsets. Initially, $C_1$ includes all the single element subsets of the set of items I. **And** $SC_1$ = Closely Spaced Candidate – 1 Itemsets. Initially, this also consists of all the single element subsets of the set of items I. Since, before the beginning of the scan, support (σ) and the average Inter Itemset Distance (IID) d is not calculated and therefore, all the single element subsets of I are potentially frequent and also closely spaced. The algorithm is stated below.

**[1] Algorithm:** GenerateCloselySpacedLargeOneItemset()

**Inputs:**

| | |
|---|---|
| D | // Database of Transactions |
| I | // Items: Set of all Items |
| σ | // Input threshold for Support in percentage |
| l | // Input threshold for Average Inter Itemset Distance (IID) |

**Outputs:**

| | |
|---|---|
| $SC_1$ | // Closely Spaced Candidate 1– Itemsets |
| $L_1$ | // Large 1 – Itemsets |
| $S_1$ | // Closely Spaced 1 – Itemsets |
| $SL_1$ | // Closely Spaced Large 1 – Itemsets |

**Steps:**

**1. Start**
2. Initialize k= 0;
3. int n; // n is the total number of transactions in the database
4. int LastTID = n;
5. $L_1 = \Phi$; // Initially $L_1$ is empty
6. $S_1 = \Phi$; // Initially $S_1$ is empty
7. $SL_1 = \Phi$; // Initially $SL_1$ is empty
8. $SC_1$ = I; // The itemset I is assigned to the Closely Spaced Candidate //1– itemsets $SC_1$.
9. Input σ; // σ is the input support threshold in percentage
10. float l =Average IID threshold; // Input average IID threshold as pre specified value;
11. **for (i =1; i ≤ m; i++)** // m is the total number of elements in $SC_1$
12.**{** **// Loop for initialization**.
13. int $c_i = 0$; // Initially the support count for each element of $SC_1$ is zero.
14. int $d_i = 0$; // Initially the IID = 0 for each element. d == IID.
15. int $t_i = 0$; // $t_i$ stores the TID of recent occurrence of element i.
16. int $T_i = 0$; // $T_i$ stores the TID of recent nonoccurrence of element i.
17. float $IID_i = 0$; // $IID_i$ == IID value for each element of $SC_1$.
18. **};** **// End 'for', the Loop for initialization**.
19. **for each transaction $t_j \in D$ do** // j: Transaction No. j = 1, 2, ……., n
20. {
21. **for each $ic_i \in SC_1$ do** //$ic_i$ represents the single element itemsets of $SC_1$ i =1,2,….., m.
22. {
23. if **$ic_i \in t_j$** then
24. {

```
25.                 ci := ci + 1;   // Increment the support count by 1. ci: support count of 1- itemset.
26.                 ti = tj;              // stores the TID of the recent occurrences of the element ici in ti.
27.             }
28.             if ((ci > 0) and (ici  ϵ  tj)) then
29.             {
30.                 di := di + 1;                    // increment the IID value by 1.
31.                 Ti = tj;          // stores the TID of the recent non occurrences of the element ici in Ti.
32.             }
33.          if (Ti ≠ LastTID) then
34.                 {IIDi = di – (Ti - ti);}
35.          else
36.                 {IIDi = di;}
37.          if (ci > 1) then
38.                   Average IIDi = IIDi/(ci – 1);
39.          else
40.                   Average IIDi = 1 +1;
41.        }                                                    // End inner 'for'
42.}                                                           // End outer 'for' i.e. 2 level nesting ends here.
43.  for each ici ϵ SC1 do
44. {
45.        if ((ici ≥ (σ X | D|)) then
46.                 {L1 = L1 U ici;}                    // Large 1-Itemsets
47.        if ((Average IID) ≤ l) then
48.                 {S1 = S1 U ici;}                    // Closely Spaced 1–Itemsets
49.}                                                   // End for
50.  SL1 = L1∩S1                                  // Closely Spaced Large 1–Itemsets
51. Output L1;                                          // Large 1–Itemsets
52. Output S1;                                          // Closely Spaced 1–Itemsets
53. Output SL1;
54. End.
```

## (ii) Mining Closely Spaced Large – k Itemsets ($SL_k$)

**The** modified apriori algorithm to mine Closely Spaced Large k – Itemsets from transaction databases and to generate corresponding association rules with respect to thresholds for support, Average Inter Itemset Distance (IID) and Confidence from the discovered Closely Spaced Large Itemsets is given below. It contains algorithms to mine Closely Spaced Large k – Itemsets, Closely Spaced Large 1 – Itemsets [as in (i) above], to generate Candidate k – itemsets from Large (k-1) – itemsets, to prune Candidate-k itemsets (same as in apriori), and then to generate the Closely Spaced Large k – itemsets and finally to generate the association rules from the Closely Spaced Large k – itemsets.

**[2] Modified Apriori Algorithm:**

//The modified a priori algorithm for mining Closely Spaced Large/Frequent k – itemsets based on average Inter //Itemset Distance (IID), support and confidence.

**1. Algorithm: Modified a priori**

// Initialize k = 1; int k;

**Inputs:**

**D** // Data base of Transactions
T // Set of all Transactions
I // Set of all Items

σ // Input threshold for Support
c // Input threshold for Confidence
l // Input threshold for Average Inter Itemset Distance (IID)
$SC_1$ // Closely Spaced Candidate – 1 Itemsets
$SL_1$ // Closely Spaced Large– 1 Itemsets

**Outputs:**

$SC_k$ // Closely Spaced Candidate k – Itemsets ($k \geq 2$)
$SL_k$ // Closely Spaced Large k – Itemsets ($k \geq 1$)
SL // Closely Spaced Large Itemsets of all sizes
$L_k$ // Large k–Itemsets
$S_k$ // Closely Spaced k–Itemsets
L // Set of all Large Itemsets
S // Set of all Closely Spaced Itemsets

**Steps:**

1. int k;
2. Initialize k =1;
3. int n;
4. int Last TID =n;
5. L = Φ;
6. S = Φ;
7. SL = Φ;
8. float s;
9. Input σ;
10. float l;
11. l = Average threshold IID;
12. int $c_i$ =0; //$c_i$: Support count for $i^{th}$ itemset
13. float $d_i$ = 0;
14. $SC_1$ = I;
//Initially all the itemsets of size -1 are candidate itemsets ($SC_1$). Scan the database of transactions D //to count the support and the Average Inter Itemset Distance (IID) of each element of $SC_1$ to //determine $L_1$, $S_1$ and $SL_1$. $L_1$ = Large/frequent 1–itemsets; $S_1$ = Closely Spaced 1–itemsets; $SL_1$= //Closely Spaced Large 1–itemsets. For these $L_1$, $S_1$ and $SL_1$ call Algorithm //GenerateCloselySpacedLargeOneItemsets ($SC_1$) with $SC_1$ = I, the set of all items.
15. (a) $L_1$= GenerateCloselySpacedLargeOneItemsets ($SC_1$);
// Store the set of Large 1–Itemsets generated by the function
// GenerateCloselySpacedLargeOneItemsets ($SC_1$) in $L_1$ as a file;
15. (b) $S_1$= GenerateCloselySpacedLargeOneItemsets ($SC_1$);
//Store the set of Closely Spaced 1- Itemsets generated by the function Generate //Closely Spaced LargeOneItemsets ($SC_1$) in $S_1$ // as a file;
15. (c) $SL_1$= GenerateCloselySpacedLargeOneItemsets ($SC_1$);
//Store the set of Closely Spaced Large 1- Itemsets generated by the function //Generate Closely Spaced LargeOneItemsets ($SC_1$) in $L_1$ as a file; Function Call: //the function: GenerateCloselySpacedLargeOneItemsets (I) is called with I = $SC_1$..
16. Initialize k: = 2; // k represents the pass number
17. **while ($L_{k-1} \neq \Phi$) do** **// begin while**
18. **{**
19. $L_k$ = Φ;
20. $S_k$ = Φ;
21. $SL_k$ = Φ;
22. $SC_k$ = GenerateCandidateItemset ($L_{k-1}$); //$SC_k$ = GenerateCandidateItemset with the $L_{k-1}$ //found in the previous pass i.e. $SC_2$ = //GenerateCandidateItemset($L_1$) in the first pass // of this loop.
23. $SC_k$ = Prune ($SC_k$);

```
24.     for each I_i ∈ SC_k do
25.     {
26.             c_i = 0;            //Small c_i = 0 i.e. Initial support counts for each itemset is zero
27.             d_i = 0;            //Initial IID = 0 for each itemsets.
28.     }                           //End for
```

//Now for all transactions $t_j \in T$ do: Increment the counts for all candidates in $SC_k$ that are contained //in transaction $t_i$ and increment the Inter Itemset Distance (IID) of all candidates by adding 1 to the //Inter Itemset Distance (IID) count of all candidates in $SC_k$ whenever the itemset is not contained // in $t_j$.

```
29.     for each t_j ∈ T do         // T: Transaction database
30.     {
31.             for each I_i ∈ SC_k do      // Here i is not any variable, I_i denotes each itemset of SC_k
32.             {
33.                     if I_i ∈ t_j then
34.                     {
35.                             c_i = c_i + 1;
36.                             t_is = t_j;         // Stores the TID of the recent occurrence in t_is,
                                                    // which is just a subscripted variable.
37.                     }                           // End if
38.             if ((c_i > 0) and (I_i ∈ t_j)) then
39.                     {
40.                     d_i = d_i +1;               // increment IID by 1
41.                             t_id = t_j;                 // Stores the TID of recent non occurrence in
                                                            // the subscripted variable t_id
42.                     }                           // End if
43.                     if (t_id ≠ Last TID) then
44.                     {IID = d_i – (t_id – t_is);}
45.                     else
46.                     {IID = d_i;}
47.             if (c_i > 1) then
48.                     Average IID = IID/(c_i – 1);
49.                     else
50.                             Average IID = 1 + 1;
51.             }                                   // end for (inner loop) began in line no.31
52.}                                        // end for (outer loop) began in line no.29
53.   for each I_i ∈ SC_k do
54.{
55.       if ((c_i ≥ (σ X | D|)) then               // X is multiplication
56.             L_k = L_k U I_i;                    // Large – k Itemsets
57.       if ((Average IID) ≤ (Threshold average IID)) then
58.             S_k = S_k U I_i;                    // Closely Spaced – k Itemsets
59.     }                                           // end for
60.     SL_k = L_k U S_k;                   // Closely Spaced Large – k Itemsets
61.     k: = k + 1;                         // Increment the pass number
62. }                                       // End while loop began on line no. 17
63.     L= L_1 U L_k;                       // Set of all large itemsets
64.     S=S_1 U S_k;                        // Set of all closely spaced itemsets
65.     SL = SL_1 U SL_k;                   // Set of all closely spaced large itemsets
66.     Output L;
67.     Output S;
68.     Output SL;
69. End.
```

### (iii) Generating Candidate k–Itemsets ($SC_k$) from the large (k – 1) – itemsets ($L_{k-1}$) using the function GenerateCandidateItemsets ($L_{k-1}$).

**[3] Algorithm: GenerateCandidateItemsets ($L_{k-1}$)**

**Input:** $L_{k-1}$, the Large (k – 1)–itemsets (for $k \geq 2$).

**Output**: $SC_k$, the closely spaced candidate sets of size k which are actually candidate itemsets of size k for becoming large itemsets of size k ($L_k$) and closely spaced itemsets of size k ($S_k$) ($k \geq 2$). From these candidate itemsets, the large k – itemsets $L_k$ based on input values of support ($\sigma$) and the closely spaced k – itemsets i.e. $S_k$ ($k \geq 2$) based on the input values of Average Inter Itemset Distance (d) are discovered. Therefore, these candidate sets are called Closely Spaced Candidate k – itemsets ($k \geq 2$) and denoted by $SC_k$.

**Steps:**
**1.** $SC_k = \Phi$;
**2. For each** $I \in L_{k-1}$ **do** // $k \geq 2$
**3. {**
4. **For each** $J \neq I \in L_{k-1}$ **do**
5. {
6. **if** (k – 2) of the elements in I and J are equal **then**
7. $SC_k = SC_k \cup (\{I \cup J\})$;
8. }
9. }
10. Return ($SC_k$);
11. END.

**(iv) Prune Candidate k – Itemsets ($SC_k$):** Algorithm to Prune Candidate Itemsets ($SC_k$) generated in [3] above is as follows.

**[4] Algorithm: prune ($SC_k$)**

**Input:** $SC_k$, ($k \geq 2$), the set of closely spaced candidate k itemsets
**Output**: $SC_k$, ($k \geq 2$), the pruned closely spaced candidate itemsets of size k

**Steps:**

1. For each $c \in SC_k$
2. {
3. For each (k – 1) subsets **d** of **c** do
4. {
5. if $d \in L_{k-1}$ then
6. $SC_k = SC_k - \{c\}$
7. }
8. }
9. Return ($SC_k$);

**(v) Generate closely spaced association rules from closely spaced large Itemsets (SL):** Algorithm to generate closely spaced association rules from closely spaced large Itemsets (SL).

**[5] Algorithm: AprioriRuleGeneration (SL)**

**Input:**

**D** // Data base of transactions
**I** // Set of Items
**SL** // Set of all closely Spaced Large Itemsets
σ // support
**c** // Confidence
**Average IID** // Average Inter Itemset Distance (IID) or Spread

**Output**:
R // SET of all Association rules satisfying σ, c and average IID or spread
// called closely spaced association rules.

**Steps:**
**1.** R = Φ; // Initially the set of rules R is empty
**2. for each** I ϵ SL **do**
**3. {**
4. **for each** x ϵ I such that x ≠ Φ do
5. {
6. If **(support (I)/support (x))** ≥ **c** then
7. R = R U {**x** → **(I – x)};**
8. }
9. }
10. OUTPUT (R)
11. END

## 4.3 Analysis of the Algorithm

The computational complexity of the proposed modified algorithm depends upon support threshold, number of items, number of transactions and the average width of the transactions in the dataset. The value of the Average Inter Itemset Distance threshold will not affect the computational complexity of the proposed algorithm since it is not required to make any extra pass of the dataset while counting the value of the Average Inter Itemset Distance of each candidate itemset.

**Time Complexity of the Proposed Algorithm**

**(a) Generation of Frequent -1 and Closely Spaced – 1 Itemsets:** These two tasks are performed in the same loop of the algorithm and hence no extra scan of the database is required to calculate the Average Inter Itemset Distances of the **Candidate – 1** itemsets. In this step the frequent -1 ($L_1$) and Closely Spaced – 1 ($S_1$) Itemsets are determined. Thereafter, the set of **Closely Spaced Frequent -1 Itemsets** are found by the intersection of $L_1$ and $S_1$. If w is the average transaction width and **n** is the total number of transactions in the database then this operation requires O(nw) time.

**(b) Candidate Generation:** To generate candidate k–itemsets, pairs of frequent (k – 1)–itemsets are merged to determine whether they have at least k – 2 common elements. Each merging operations requires at most k – 2 equality comparisons. In the best case, every merging step produces a viable candidate k–itemset. In the worst case scenario, the algorithm must merge every pair of frequent (k – 1)–itemsets found in the previous iteration. Therefore, the overall cost of merging frequent itemsets is

$$\sum^{w}_{k=2} (k-2)\,|C_k| < \text{Cost of Merging} < \sum^{w}_{k=2} (k-2)|F_{k-1}|^2$$

During candidate generation a hash tree is also constructed to store the candidate itemsets. The cost for populating the hash tree with candidate itemsets is $O( \sum^{w}_{k=2} k \ | C_k|)$, where k is the maximum depth of the tree.

During candidate pruning, we need to verify that the (k – 2) subsets of every candidate k – itemset are frequent. Since the cost for looking up a candidate in a hash tree is O(k), the candidate pruning step requires $O ( \sum^{w}_{k=2} k(k – 2) |C_k|)$ time.

**(c) Support Counting:** The number of itemsets of size k produced by a transaction of length |t| is ${}^{|t|}C_k$ and the number of hash tree traversals required for each transaction is also equal to ${}^{|t|}C_k$. If w is the maximum transaction width and $\sigma_k$ is the cost of updating the support count of a candidate k – itemset in the hash tree, then the cost of support counting is $O(N\sum_k ({}^{w}C_k \ \sigma_k))$. Since, for counting the Average Inter Itemset Distances of the itemstes no additional loop is employed and it is done in the same loop used for support counting, therefore the cost of calculating the Average Inter Itemset Distances of the itemstes is $O(N\sum_k ({}^{w}C_k \ d_k))$, where $d_k$ is the cost of updating the Average Inter Itemset Distance of a candidate k – itemset in the hash tree. Therefore, the total cost of support counting and calculating the Average Inter Itemset Distances is $O(N\sum_k ({}^{w}C_k (\sigma_k +d_k)))$.

**(d) Rule Generation:** A closely spaced large k – itemset can produce up to $(2^k – 2)$ association rules excluding the rules which have empty antecedents (Φ=>Y) and empty consequents (Y=> Φ). The calculation of confidence of a **closely spaced association rule** does not require additional scans of the transaction database since it can be calculated by using the supports of the itemsets (X U Y) and X of the rule X=>Y in the ratio *sup(X U Y)/sup(X)*.

## 5. IMPLEMENTATION AND RESULTS

The apriori and the modified apriori algorithms are implemented in Java with windows XP operating system in a PC with Intel Core2 Duo Processor and 512MB of RAM. The data set used is the *retail* dataset of size 4.2MB available in the UCI repositories. The significance of Average Inter Itemset Distance is: the lesser the value of Average Inter Itemset Distance of an itemset, the nearer are the occurrences of the itemset in the transactions of the dataset. A closely spaced large itemset has to fulfill the two threshold values viz. the minimum support threshold and the maximum Average Inter Itemset Distance threshold. As a result, the number of qualified itemsets for rule generation reduces and hence the number of generated rules also reduces with the added meaning obtained from the Average Inter Itemset Distance for the rule. The quantity of reduction of frequent itemsets as compared to the conventional a priori approach depends on the threshold values of both the parameters. In the case of high support and small value of Average Inter Itemset Distance, the number of frequent sets discovered will be less as compared to low support and high value of Average Inter Itemset Distance.

(1)The following (table1) shows that the number of discovered large itemsets and the corresponding association rules in the modified version of the algorithm is reduced.

**Table1:** Number of Rules with apriori algorithm (min_ sup (2%) and min_conf (20%)) and modified apriori algorithm (min_ sup (2%) and min_conf (20%) and AverageIID=20.0).

| Size | No. of Transactions | Support count at min_ sup (2%) | Apriori Algorithm min_ sup (2%), min_conf (20%) | | | Modified Apriori Algorithm min_ sup (2%), min_conf (20%), AverageIID=20.0 | | |
|---|---|---|---|---|---|---|---|---|
| | | | LI: No. of Large Itemsets | No. of Rules | Execution Time (Sec) | No. of Closely Spaced Large Itemsets | No. of Rules | Execution Time (Sec) |
| 100 kB | 1804 | 36 | 110 | 140 | 87.69 | 24 | 44 | 85.27 |
| 200 kB | 4077 | 81 | 78 | 100 | 217.82 | 18 | 34 | 217.96 |
| 300 kB | 6283 | 125 | 69 | 85 | 388.23 | 18 | 32 | 379.25 |
| 400 kB | 8459 | 169 | 67 | 84 | 593.83 | 18 | 32 | 665.29 |
| 500 kB | 10786 | 215 | 65 | 87 | 709.99 | 18 | 32 | 800.31 |
| 600 kB | 13064 | 261 | 63 | 85 | 951.009 | 18 | 34 | 1018.04 |
| 700 kB | 15745 | 314 | 60 | 82 | 1179.191 | 19 | 34 | 1254.274 |
| 800 kB | 17441 | 348 | 59 | 83 | 1455.389 | 18 | 32 | 1422.941 |
| 900 kB | 20009 | 400 | 59 | 89 | 1863.744 | 18 | 34 | 1724.895 |
| 1000kB | 21857 | 437 | 60 | 89 | 1856.57 | 18 | 34 | 1810.383 |

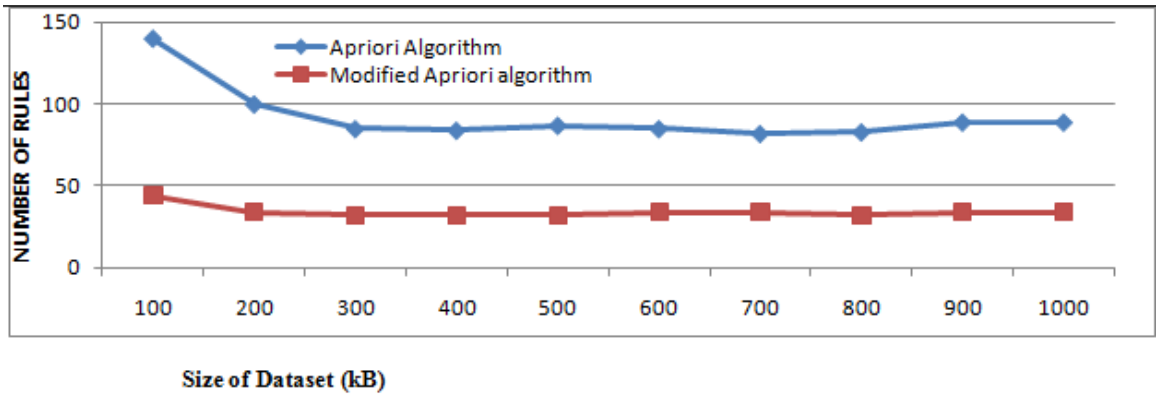


**Figure1:** The number of association rules discovered from the apriori algorithm and the modified apriori algorithm with support threshold = 2%, confidence threshold = 20% and Average IID threshold = 20.0 and by varying the dataset sizes.

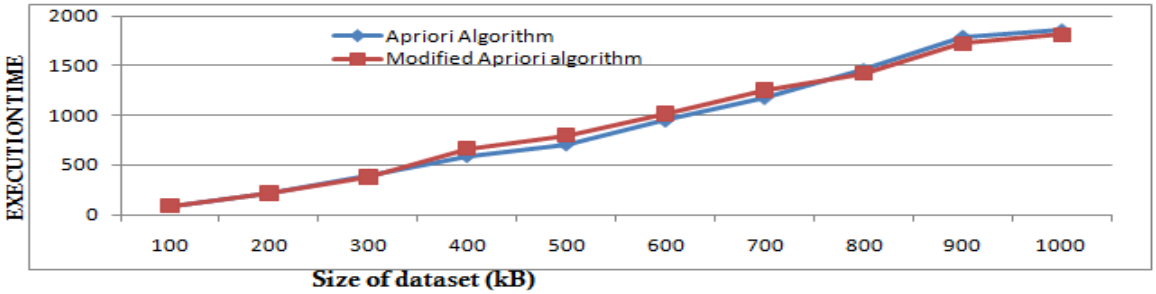


**Figure2:** Comparison of Execution Time required for the apriori algorithm and the modified algorithm with support threshold of 2%, confidence threshold of 20% and Average IID threshold of 20.0 and by varying the size of the dataset.

As shown in the graph the modified algorithm efficiently reduces the number of association rules and the execution time required are comparable.

2. Behaviour with variation in minimum confidence threshold at constant minimum support and AverageIID: The following results are obtained as a result of comparison of the effect of varying minimum confidence on both the apriori and the modified apriori algorithms with fixed minimum support and fixed averageIID for a dataset of size 100kB(Table2).

**Table 2:** comparison of the effect of varying minimum confidence on both the apriori and the modified apriori algorithms with fixed minimum support (1%) and fixed averageIID=25.0 for a dataset of size 100kB.

| Minimum confidence (min_conf) (%) | Support Count at Minimum support (1%) (min_sup) | Apriori Algorithm: min_sup (1%) (dataset of 100kB,1804 Transactions) | | | Modified Apriori Algorithm: min_sup (1%) and AverageIID=25.0 (dataset of 100kB, 1804 Transactions) | | |
|---|---|---|---|---|---|---|---|
| | | No. of LI: Large Itemsets | No. of Rules | Execution Time (Sec) | No. of Closely Spaced Large itemsets(CSLI) | No. of Rules | Execution Time (Sec) |
| 0% | 18 | 440 | 1242 | 337.46 | 35 | 92 | 213.907 |
| 10% | 18 | 440 | 737 | 335.401 | 35 | 86 | 213.408 |
| 20% | 18 | 440 | 646 | 339.067 | 35 | 64 | 214.001 |
| 30% | 18 | 440 | 567 | 339.737 | 35 | 48 | 212.956 |
| 40% | 18 | 440 | 502 | 343.024 | 35 | 41 | 215.64 |
| 50% | 18 | 440 | 408 | 345.821 | 35 | 33 | 215.596 |
| 60% | 18 | 440 | 303 | 341.0484 | 35 | 22 | 214.446 |
| 70% | 18 | 440 | 170 | 342.681 | 35 | 08 | 214.048 |
| 80% | 18 | 440 | 84 | 339.082 | 35 | 04 | 214.953 |
| 90% | 18 | 440 | 25 | 338.146 | 35 | 0 | 213.533 |
| 100% | 18 | 440 | 0 | 333.777 | 35 | 0 | 213.019 |

It is observed that in the case of apriori algorithm the number of rules decreases more rapidly with respect to different minimum confidence threshold as compared to the modified algorithm (Figure 3). Further, the execution time and their difference remain nearly constant for both the apriori and the modified algorithm for different minimum confidence threshold values (Figure 4).

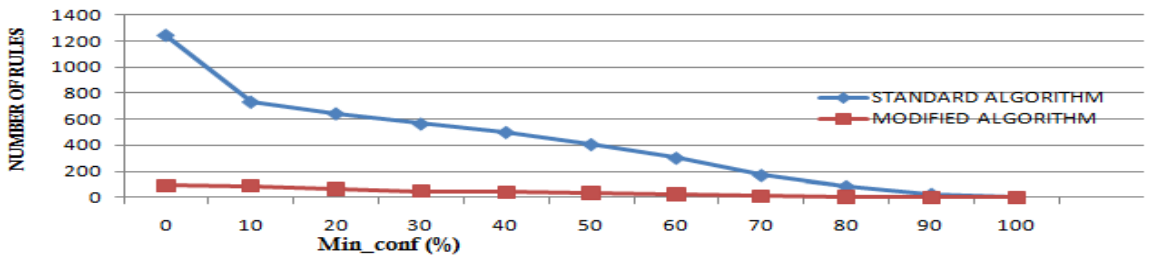


**Figure3:** Comparison of the association rules discovered with the apriori algorithm and the modified algorithm by varying the minimum confidence threshold at constant minimum support threshold.

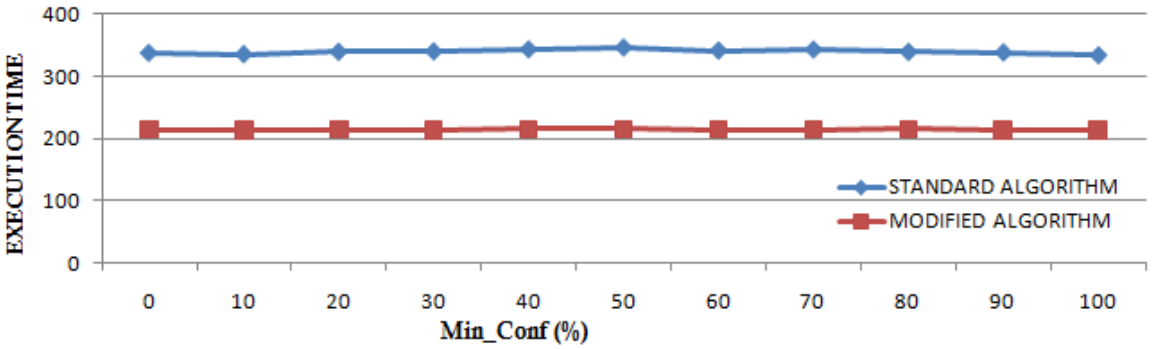


**Figure 4:** Comparison of the Execution time required for the apriori algorithm and the modified algorithm by varying Minimum Confidence threshold at constant minimum support threshold and fixed averageIID.

3. The effect of varying averageIID with the Modified Algorithm at constant minimum support threshold (1%) and minimum confidence (20%) for a dataset of size 100kB is shown below (Table 3).

**Table 3:** Effect of varying averageIID with the modified algorithm at constant thresholds for minimum support (1%) and minimum confidence (20%) for a dataset of size 100kB.

| Average IID | Support Count at minimum support (1%) | No. of Large Itemsets (LI) Discovered | No. of Closely Spaced Large Itemsets (CSLI) Discovered | No. of Rules | Execution Time (Sec) |
|---|---|---|---|---|---|
| 0 | 18 | 440 | 0 | 0 | 218.759 |
| 1 | 18 | 440 | 1 | 0 | 243.323 |
| 2 | 18 | 440 | 2 | 0 | 216.169 |
| 3 | 18 | 440 | 4 | 2 | 217.916 |
| 4 | 18 | 440 | 5 | 4 | 217.988 |
| 5 | 18 | 440 | 7 | 6 | 214.529 |
| 10 | 18 | 440 | 12 | 16 | 215.545 |
| 15 | 18 | 440 | 19 | 28 | 245.576 |
| 20 | 18 | 440 | 26 | 44 | 231.099 |
| 25 | 18 | 440 | 35 | 64 | 214.61 |
| 30 | 18 | 440 | 48 | 73 | 222.581 |
| 35 | 18 | 440 | 62 | 84 | 237.243 |
| 40 | 18 | 440 | 83 | 99 | 238.197 |
| 45 | 18 | 440 | 104 | 132 | 229.305 |
| 50 | 18 | 440 | 125 | 166 | 215.95 |
| 55 | 18 | 440 | 154 | 210 | 215.015 |
| 60 | 18 | 440 | 185 | 269 | 217.059 |
| 65 | 18 | 440 | 222 | 331 | 218.808 |
| 70 | 18 | 440 | 257 | 404 | 217.652 |
| 75 | 18 | 440 | 285 | 431 | 237.276 |
| 80 | 18 | 440 | 320 | 462 | 213.762 |
| 85 | 18 | 440 | 356 | 517 | 217.157 |

| | | | | | |
|---|---|---|---|---|---|
| 90 | 18 | 440 | 390 | 560 | 211.646 |
| 95 | 18 | 440 | 408 | 593 | 212.972 |
| 100 | 18 | 440 | 434 | 637 | 233.853 |
| 104.81 | 18 | 440 | 440 | 646 | 213.705 |

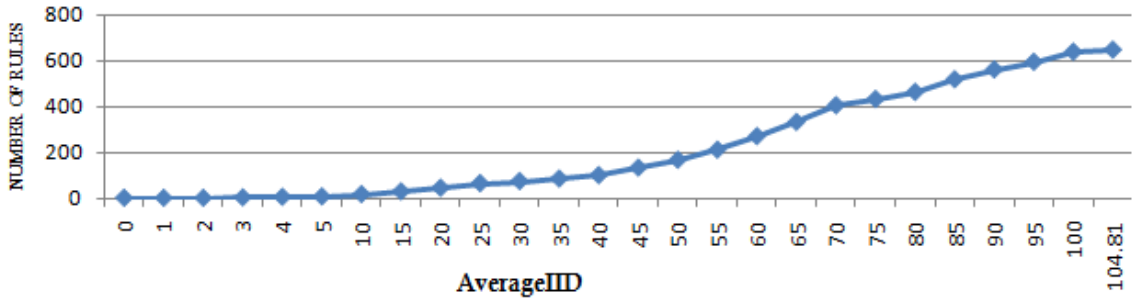


**Figure 5:** Association Rules discovered with the modified algorithm by varying the AverageIID for for a dataset of size 100kB at constant minimum support (1%) and minimum confidence (20%).

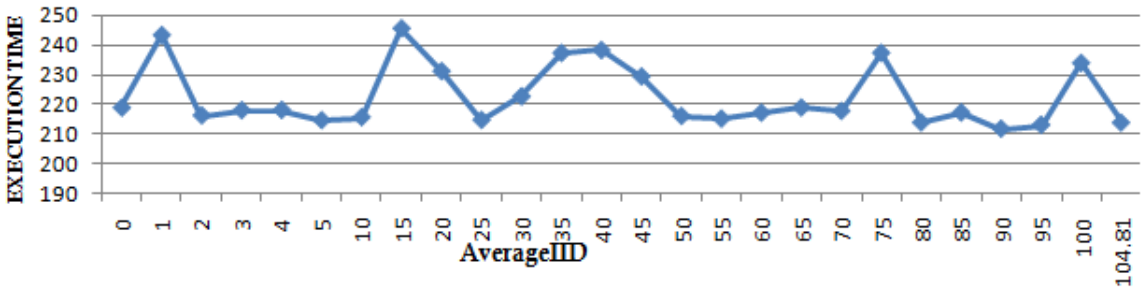


**Figure6:** Execution time Required for the modified algorithm versus AverageIID for a dataset of size 100kB at constant minimum support (1%) and minimum confidence (20%).

**Comparative Performance:** The comparison of the conventional and the modified apriori algorithms shows that the number of discovered large itemsets and the association rules in the modified version of the algorithm is reduced with the introduction of the average inter itemset distance as a new measure of interestingness. We are calling such rules as the closely spaced association rules as these are discovered from the closely spaced large itemsets.

## 6. CONCLUSION

In this paper, a detail algorithm based on apriori algorithm is designed to discover frequent itemsets and association rules with Average Inter Itemset Distance along with support and confidence. A theoretical formulation is provided for Inter Itemset Distance and a range for values of Average Inter Itemset Distance for an itemset is worked out. Then both the algorithms are implemented and their results are compared while mining closely spaced frequent itemsets and the corresponding closely spaced association rules with average inter itemset distance along with the conventional measures of support and confidence. The results show that the number of generated rules is reduced in comparison to the conventional apriori algorithm. As future scope of work, the modified approach shall be extended to mine association rules integrated in database environment by using inter itemset distance.

## REFERENCES


[1] R. Agrawal, T. Imielinsky and A. Swami. Mining Association Rules between Sets of Items in Large Database, Proceedings of ACM SIGMOD Conference on Management of Data, Washington DC, pp. 207 – 216, 1993.

[2] R. Agrawal and R. Srikant. Fast Algorithms for Mining Association Rules in Large Databases, Proceedings of 20th International Conference on Very Large Data Bases (VLDB), Santiago, Chile, pp. 487 – 499, 1994.

[3] R. Agrawal and J. Shafer. Parallel Mining of Association Rules, IEEE Transactions on Knowledge and Data Engineering, 8(6), pp. 962 –969, 1996.

[4] R. Agrawal, H. Mannila, R. Srikant H. Toivonen, and A. I. Verkamo. Fast Discovery of Association Rules, Advances in Knowledge Discovery and Data Mining, U. Fayyad and et. el. (Ed.) pp. 307 – 328, Menlo Park, California: AAAI Press, 1996.

[5] S. Brin, R. Motwani and C. Silverstein. Beyond Market Baskets: Generalizing Association Rules to Correlations, Proceedings of the ACM International Conference on Management of Data, pp. 265 – 276, 1997.

[6] P. K. D. Sarma, and A. K. Mahanta. Reduction of Number of Association Rules with Inter Itemset Distance in Transaction Databases. International Journal of Database Management Systems (IJDMS), 4(5), pp. 61 – 82, 2012.

[7] A. Savasere, E. Omiecinsky, and S. Navathe. An Efficient Algorithm for Mining Association Rules in Large Databases, Proceedings of 21st International Conference on Very Large Databases. Zurich, Switzerland, pp.432-444, 1995.

[8] J. Han, J. Pei and Y. Yin. Mining Frequent Patterns without Candidate Generation. Proceedings of 2000 ACM-SIGMOD International Conference on Management of Data, Dallas, USA, pp. 1–12, 2000.

[9] P. N. Tan, V. Kumar and J. Srivastava. Selecting the Right Interestingness Measure for Association Patterns, Proceedings of ACM SIGKDD International Conference on Knowledge Discovery in Databases, Edmonton, Canada, pp. 32–41, 2002.

[10] R. J. Hilderman, and H. J. Hamilton. Knowledge Discovery and Measures of Interest. Kluwer Academic Publishers, 2001.

[11] E. Omiecinski. Alternative Interest Measures for Mining Associations, IEEE Trans. Knowledge and Data Engineering, 15, 57–69, 2003.

[12] F. Guillet and H. Hamilton. Quality Measures in Data Mining. Springer, 2007.

[13] I.N.M. Shaharanee, F. Hadzic and T.S. Dillon. Interestingness measures for association rules based on statistical validity. Knowledge-Based Systems, 24(3), pp. 386–392, 2011.

[14] Y. Xu, Y. Li, and G. Shaw. Reliable representations for association rules. Data & Knowledge Engineering, 70(6), pp. 555–575, 2011.

[15] H. Liu, L. Liu and H. Zhang. A fast pruning redundant rule method using Galois connection. Applied Soft Computing, 11(1), pp.130 – 137, 2011.

[16] J. Hills, A. Bagnall, B. Iglesia and G. Richards. Brute Suppression: a size reduction method for Apriori rule Sets, J Intell Inf Syst, Springer Science+Business Media, New York, 2013.

[17] B. Vo and B. Le. A Frequent Closed Itemset Lattice based Approach for Mining Minimal Non- Redundant Association Rules. International Journal of Database Theory and Applications, 4(2), pp. 23 – 34, 2011.

[18] J. Cheng, Y. Ke, and W. Ng. Effective elimination of redundant association rules. Data Mining Knowledge Discovery. 16, pp. 221 – 249, 2008.

[19] Y. Xu, Y. Li and G. Shaw. A Reliable Basis for Approximate Assiciation Rules. IEEE Information Bulletin, 9(1), pp. 25 – 31, 2008.

[20] M. Hashler, and S. Cellubonia. Visualizing Association Rules: Introduction to R – extension Package arulsViz, pp. 1 – 27, 2011.

[21] W. Davis, P. Schwarz and E. Terzi. Finding Representative Association Rules from Large Rule Collection, SIAM, pp. 521 -532, 2008.

[22] S. Prakash and R. M. S. Parvathi. An Enhanced Scaling Apriori for Association Rule Mining Efficiency, European Journal of Scientific Research, 39(2), pp. 257 – 264, 2010.

[23] G. Liu, H. Zhang, and L. Wong. Controlling False Positives in Association Rule Mining, Proceedings of VLDB Endowment, 5(2), pp. 145 – 156, 2011.

[24] N. Radhika, and K. Vidya. Association Rule Mining based on Ontological Relational Weights, International Journal of Scientific and Research Publication, 2(1), pp. 1 – 5, 2012.

[25] C. Marinica, and F. Guillet. Knowledge Based Interactive Postmining of Association Rules using Ontologies. IEEE transactions on Knowledge and Data Engineering, 22(8), pp. 784 – 797, 2010.

[26] V. Bogorny, B. Kuijpers and L. O. Alvares. Reducing Uninteresting Spatial Association Rules in Geographic databases using Background Knowledge: A summary of Results. International Journal of Geographic Information Science, Taylor & Francis, 22(4 – 5), pp. 361 – 386, 2008.

[27] J. Wang, J. Han, Y. Lu, and P. Tzvetkov. TFP: An Efficient Algorithm for Mining Top-k Frequent Closed Itemsets, IEEE Transactions on Knowledge and Data Engineering, 17:652–664, 2005.

[28] F. N. Afrati A. Gionis and H. Mannila. Approximating a Collection of Frequent Sets, Proceedings of 2004 ACM SIGKDD International Conference on Knowledge Discovery in Databases, Seattle, USA, pp. 2–19, 2004
[29] X. Yan, H. Cheng, D. Xin & J. Han. Summarizing Itemset Patterns: A Profile-based Approach, Proceedings of ACM SIGKDD International Conference on Knowledge Discovery in Databases, Chicago, USA pp.314–323, 2005.
[30] D. Xin, J. Han, X. Yan, and H. Cheng. Mining Compressed Frequent-Pattern Sets, Proceedings of International Conference on Very Large Data Bases (VLDB), Trondheim, Norway, pp. 709–720, 2005.
[31] S. Jaroszewics and D. Simovici. Interestingness of Frequent Itemsets Using Bayesian Networks and Background Knowledge, Proceedings of 10th International Conference on Knowledge Discovery and Data Mining, Seattle, USA, pp. 178 -186, 2004.
[32] M.C. Tseng, and W.Y. Lin. Efficient Mining of Generalized Association Rules With Non-uniform Minimum Support, Data & Knowledge Engineering, Science Direct, 62, pp. 41–64, 2007.
[33] M. Hashler, C. Buchta and K. Hornik. Selective Association Rule Generation, Computational Statistic, Kluwer Academic Publishers, 23(2), pp. 303-315, 2008.
[34] F. Bonchi and C. Lucchese. Pushing tougher constraints in frequent pattern mining, 2005.

## AUTHORS

**Pankaj Kumar Deva Sarma**
He received the B.Sc (Honours) and M. Sc. Degrees in Physics from the University of Delhi, Delhi, India before receiving the Post Graduate Diploma in Computer Application and the M. Tech degree in Computer Science from New Delhi, India. He is currently an associate professor of Computer Science in the University Department of Computer Science at the Assam University, Silchar, India. His primary research interest is in algorithms, data base systems, data mining and knowledge discovery, parallel and distributed computing, artificial intelligence and neural network. He was the former head of the department of Computer Science of Assam University and was the organizing vice president of the national conference on current trends in computer science organized at the Assam University in the year 2010.

**Anjana Kakati Mahanta**
She received the Bachelors and Masters Degrees in Mathematics from the Gauhati University, Guwahati, India before receiving the Ph. D. in Computer Science from the same university. She is currently a professor of Computer Science in the University Department of Computer Science at the Gauhati University, Guwahati, India. Her primary research interest is in algorithms, data base systems, data mining and knowledge discovery. She is presently the head of the department of Computer Science of Gauhati University.