Computing and Informatics, Vol. 42, 2023, 280–310, doi: 10.31577/cai_2023_2_280

NOVEL APPROACH TO HIDE SENSITIVE ASSOCIATION RULES BY INTRODUCING TRANSACTION AFFINITY

Kshitij Pathak

Department of Computer Science and Engineering
Government Polytechnic College
Mandsaur, MP, India
e-mail: er.k.pathak@gmail.com

Sanjay Silakari

Department of Computer Science and Engineering
University Institute of Technology
RGPV Bhopal, MP, India
e-mail: ssilakari@yahoo.com

Narendra S. Chaudhari

Department of Computer Science and Engineering
Indian Institute of Technology
Indore, MP, India
e-mail: nsc0183@yahoo.com

Abstract. In this paper, a novel approach has been proposed for hiding sensitive association rules based on the affinity between the frequent items of the transaction. The affinity between the items is defined as Jaccard similarity. This work proposes five algorithms to ensure the minimum side-effects resulting after applying sanitization algorithms to hide sensitive knowledge. Transaction affinity has been introduced which is calculated by adding the affinity of frequent items present in the transaction with the victim-item (item to be modified). Transactions are selected either by increasing or decreasing value of affinity for data distortion to hide

association rules. The first two algorithms, MaxaffinityDSR and MinaffinityDSR, hide the sensitive information by selecting the victim item as the right-hand side of the sensitive association rule. The next two algorithms, MaxaffinityDSL and MinaffinityDSL, select the victim item from the left-hand side of the rule whereas the Hybrid approach picks the victim item from either the left-hand side or right-hand side. The performance of proposed algorithms has been evaluated by comparison with state-of-art methods (Algo 1.a and Algo 1.b), MinFIA, MaxFIA and Naive algorithms. The experiments were performed using the dataset generated from IBM synthetic data generator, and implementation has been performed in R language.

Keywords: Association rule hiding, transaction affinity, affinity, AffinityDSR, AffinityDSL

Mathematics Subject Classification 2010: 68T05, 68T30

1 INTRODUCTION

1.1 Terminology and Preliminaries

1. Let $I = \{I_1, I_2, I_3, I_4, \ldots, I_n\}$, a set containing a finite set of literal, known as items and cardinality of the set represented by $|I|$ is n.
2. Any subset $I_s \subseteq I$ is known as an itemset over I. An itemset of size n, represented as n-itemset. Example: Let $I = \{a, b, c, d, e\}$ then a, b, c, d and e are known as items or 1-itemset. ab, bc and cd are examples of 2-itemset. abc, bcd and cde are examples of 3-itemset and so on.
3. The support of itemset I is equal to the number of transaction having itemset I in database D. It is represented by Support(I) or Support(I, D).
4. Database D consisting of m transactions is represented by $D = \{T_1, T_2, T_3, T_4, \ldots, T_m\}$ and $|D|$ is equal to the number of transactions present in the dataset.
5. A transaction set T_s over I is pair $T_s = (\text{tid}, I_s)$ where I_s is the itemset and tid is a unique identifier.
6. $T[m, n]$ is used to identify n^{th} item of m^{th} transaction in the database.
7. The support of the association rule $A \to B$ is calculated by

$$\text{Support}(A \to B) = \text{Support}(A, B) \div |D|. \tag{1}$$

8. The Confidence of the association rule $A \to B$ is calculated by

$$\text{Confidence}(A \to B) = \text{Support}(A, B) \div \text{Support}(A). \tag{2}$$

9. An itemset is said to be frequent if its support is greater than user defined mfreq, minimum frequency threshold and itemsets having frequency lower than mfreq are treated as non-frequent items.

10. Relation between mfreq and user defined threshold MST is MST (%) = mfreq $\div$ $|D| * 100$.

11. An association rule $A \rightarrow B$ is considered to be significant if support and confidence of the rule is greater than or equal to user defined support and Confidence i.e. MST (Minimum Support Threshold) and MCT (Minimum Confidence Threshold) i.e. Support$(A \rightarrow B) \geq$ MST and Confidence$(A \rightarrow B) \geq$ MCT.

12. User marked some association rules to be hided from the set of generated association rules, known as *sensitive* and remaining rules are known as *non-sensitive association rules*.

13. A subset of transactions that can be modified/updated to hide sensitive association rule is known as *candidate transactions or sensitive transactions* whereas modified candidate transactions are known as *victim transactions*.

14. *Victim item* is a term used to represent the itemset being modified by the association rule hiding method.

1.2 Motivating Example

Let us suppose that the officials of BigMDSBazaar company purchase socks from XYZ socks company. XYZ socks company official gives an offer to BigMDSBazaar official that they are ready to reduce the price of their socks if BigMDSBazaar would share their customer data with XYZ socks company. BigMDSBazaar thought that it is a good deal and customer's data can be shared, so, both the parties have accepted the deal. XYZ socks company now started mining the customer's data they receive from BigMDSBazaar. They identified that, in 70 % cases, whoever purchases A type of shoes also going to purchase XYZ socks. After learning this knowledge from the datasets, XYZ socks company made a public offer that the customer purchasing the socks from their company will be given a 20 % cashback on the purchase of A type of shoes. After this particular advertisement, many customers who are purchasing A type of shoes earlier from BigMDSBazaar have moved to XYZ socks company. In this particular scenario, we can see that BigMDSBazaar is losing its customers. At the same time, XYZ socks company increases the price of socks purchased by BigMDSBazaar in the next deal since the sales of socks have decreased at the stores of BigMDSBazaar. BigMDSBazaar is harming its own business by sharing its customer's data. Finally, we can conclude that BigMDSBazaar lost business to the XYZ socks company as a result of exchanging data without sanitizing it. Several other motivating examples have been discussed in [1, 2, 3, 4].

1.3 Association Rule Hiding

Association rules are defined as statements of the form $\{X_1, X_2, \ldots, X_n\} \rightarrow Y$ which means that Y may be present in the transaction if X_1, X_2, …, X_n are all in the transaction. Notice that the use of may imply that the rule is only probable, not identical. Note also, that there can be a set of items, not just a single item. The probability of finding Y in a transaction with all X_1, X_2, …, X_n is called confidence. The threshold (percentage) that a rule holds in all transactions is called support. The level of confidence that a rule must exceed is called interestingness.

In this research work, we focus on knowledge hiding in the database, i.e., association rule hiding. The association rule hiding problem is an extension of the very well-known database inference control problem that has been applied to multilevel and statistical databases. In the database inference control problem, the primary objective is to hide sensitive information that can be inferred from non-sensitive data, whereas in association rule hiding it is inferred that it is not only the data but the hidden information is also a threat to privacy. So, in association rule hiding, before sharing the database various data mining techniques have been applied to identify the sensitive knowledge and then sanitization algorithms are applied to modify the dataset before releasing it to hide the sensitive information/knowledge earlier present in the dataset. The main challenges that are associated with association rule hiding are what strategy should be adopted in modifying the transactions of the database so that sensitive association rule gets hidden whereas non-sensitive association rule can still be mind as earlier as possible. Another critical factor that has to be taken into consideration is how much we are modifying the dataset, i.e., there must be a proper balance between privacy and utility. Association rule hiding or hiding large itemsets approaches can be divided into five major classes.

They are heuristic, border-based, exact, data reconstruction based and cryptographic approaches. The first class, i.e., *heuristic approaches* algorithms, [4, 2, 5, 6, 7, 8] are efficient, fast, and selectively sanitize the candidate transactions and victim items from original datasets to hide the sensitive information. Heuristic approaches are a dominant area of importance and research interest because of their efficiency and scalability. However, such algorithms suffer from various side-effects because the decisions taken are local decisions, which do not necessarily provide the global best solution.

The second class, *border-based approaches*, hide sensitive association rules by modifying the borders in the lattice of frequent and non-frequent itemsets by selecting itemsets of the lattice that control the borderline position which separates frequent and nonfrequent itemsets. Border-based algorithms hide sensitive association rules by tracking the non-sensitive frequent itemsets borders, and greedily data modifications are applied such that there is a minimum effect on the quality of the border to accommodate the hiding of the rules. The work related to the border-based approach is proposed in Main [9], BBA [9], Max-Min [10], positive and negative border-based algorithms [11], and the AARHIL algorithm [12].

The third class, *exact approaches*, treats the hiding problem as a constraint satisfaction problem solved by integer programming. The constraint satisfaction problem is an optimization problem that enables the algorithm to identify optimal solutions by minimum distorting the database and introducing no new side effects. On the contrary, the time complexity of exact approaches is higher when compared to heuristic approaches because of the time taken by the integer programming solver to solve the optimization problem [13].

The fourth class, *data reconstruction-based approaches* [14, 15, 16, 17], perform the hiding process in three phases. In the first phase, association rules are mined, sensitive rules are selected by the data owner and itemset lattice is built. In the second phase, knowledge sanitization is applied on itemset lattice and the database is reconstructed from the modified lattice in the third phase.

The fifth class, *cryptographic approaches*, is used in multi-party computation where data is dispersed in several locations [18]. The owner may want to share their data but does not want the confidential information to be disclosed. The cryptographic approach can be divided into two categories: vertical partitioned distributed data and horizontal partitioned distributed data. Various approaches in this class are discussed in [19, 20, 21, 22, 23, 24].

2 AFFINITY-BASED APPROACH

The affinity between the two items i and j is defined by Aggarwal et al. [25] as

$$A(i,j) = \text{support}(i,j)/(\text{support}(i) + \text{support}(j) - \text{support}(i,j)), \tag{3}$$

where support(.) is the count of the presence of an item in the dataset. This means that affinity is defined as the Jaccard similarity between items.

We propose five sanitization algorithms based on the affinity between the frequent itemsets by introducing the concept of *transaction affinity*. Transaction affinity is calculated by adding the affinity of frequent items present in the transaction with the victim-item (Victim-item is the item selected for modification in the transaction.). Transactions having a high value of transaction affinity signify that the set of items present in the transaction has high similarity with the victim-item. We have conducted experiments to analyze the proposed methods of picking the transaction for modification based on the similarity between victim-item and frequent items present in the transaction.

The first two algorithms, MAXAffinityDSR and MINAffinityDSR, hide the sensitive association rule by selecting transactions on the basis of the similarity of the victim-item (Right-hand side of sensitive association rule) with frequent items present in the transaction. This method hides the rule by decreasing the support of the right-hand side of the rule (victim-item) thereby reducing the confidence or support of the sensitive association rule below a minimum threshold. The third and fourth approach, MAXAffinityDSL and MINAffinityDSL, consider the victim-item as the left-hand side of the rule whereas the fifth one, the hybrid approach, selects

the victim-item by using HybridCode function present in Algorithm 3. Initially, the candidate transaction picked is the one that contains all the items of sensitive association rules, i.e., if $B \rightarrow A$ is to be hidden, then candidate transactions are the ones having BA as the subset of items present in the transaction. In earlier approaches, then candidate transactions are arranged in ascending or descending order either on the basis of length of the transaction as done in [4] or they completely remove the frequent itemset from all transactions which results in increasing the number of side-effects like the border-based approach. The proposed approach uses the concept of transaction affinity to select transactions to be sanitized.

2.1 Hiding Strategy

A sensitive association rule $A \rightarrow B$ can be hidden by following two hiding strategies as per the taxonomy of association rule hiding algorithms:

1. *Support-based or reducing support below user-specified threshold:* We know that support of rule $A \rightarrow B$ is defined as the count of transactions having both A and B divided by a total number of transactions. Support can be reduced by removing either the left-hand side (A) or right-hand side (B) from the transactions in which both A and B are present.
2. *Confidence-based or reducing confidence below user-specified confidence:* Confidence of the rule is defined as support of rule items divided by the support of left-hand side of the rule. To hide the rule by confidence, any item from the right-hand side of rule is to be removed from candidate transaction (Candidate transaction, in this case, are the ones which fully supports the rule) or support of left-hand side of rule is increased by adding items in candidate transaction (Candidate transactions, in this case, are the ones which do not support left-hand and right-hand side of rule).

Example: Consider a sample database D shown in Table 1.

Transaction_Id	Items
1	ABC
2	AB
3	A
4	C

Table 1. Database D

Support Based: In the database, support of rule $A \rightarrow B$ is $2 \div 4$ or 50 % as both A and B are present in two transactions (T_1 and T_2) out of 4 transactions. Let the user-specified support threshold is 30 %. Here, the candidate transactions are T_1 and T_2, as they fully support the rules (i.e., all rule items are present in T_1 and T_2). If we remove any item from the rule from either T_1 or T_2, then the support of the rule will be $1 \div 4$ which is equal to 25 % which makes it falls below a user-specified threshold, and eventually, rules get hided.

Confidence-Based: In the database, the confidence of rule $A \rightarrow B$ is $2 \div 3$ or 66.67 % as both A and B are present in two transactions (T_1 and T_2) out of 4 transactions and A is present in 3 transactions. Let the user-specified confidence threshold is 55 %.

To hide the rule by deleting a subset of the right-hand side of the rule, we need to make 1 modification to hide the rule. Support of AB is reduced to 1 and confidence will be reduced to $1 \div 3$ or 33.33 %.

To hide the rule by increasing support of the left-hand side, we need to make 1 modification to hide the rule. Candidate transaction is 4 since transaction 4 does not support rule items. Item A will be added to transaction 4 which increases support of A to 4. Confidence will be reduced to $2 \div 4$ or 50 % which makes it falls below the user-specified confidence threshold and eventually rule gets hided.

2.2 Preliminary Definitions

Definition 1. The Transaction Id's in support of sensitive rule S_r and will be the candidate for MAXAffinityDSR and MINAffinityDSR approach is defined by:

$$T_{id}(S_r) = T_{id}(L_{S_r}) \cap T_{id}(R_{S_r}), \tag{4}$$

where L_{S_r} and R_{S_r} are the left-hand and right-hand side of the sensitive association rule.

Definition 2. The affinity of each transaction, i.e., transaction affinity in MAXAffinityDSR and MINAffinityDSR algorithm is calculated by

$$\text{affinity}_{sum}(T_{id}) = \sum_{\substack{for\ all\ 1-itemset\ m\ in\ T_{id} \\ and \\ m \subset F \\ and \\ m\ !=\ L_{S_r} \\ and \\ for\ all\ r\ in\ R_{S_r}}} \text{affinity}(m, r), \tag{5}$$

where F is set of frequent itemsets, T_{id} is a transaction, L_{S_r} and R_{S_r} are left-hand and right-hand side of association rule. It is calculated by summation of the affinity between every 1-itemset from the right-hand side of a sensitive association rule, and every frequent 1-itemset present in the transaction ignoring itemsets belongs to the left-hand side of the transaction. Consider a sample database TD shown in Table 2.

Consider a transaction T_1 with items $UWXYZ$, sensitive association rule $U \rightarrow X$, frequent items = $\{U, W, X, Z\}$. Then, $m = \{W, Z\}$, $r = \{X\}$ and transaction affinity for T_1 is affinity(X, W) + affinity(X, Z).

Transaction_Id	Items
T1	UWXYZ
T2	UWXZ
T3	UWX
T4	WXZ

Table 2. Database *TD*

Definition 3. All the candidate transactions identified for approaches are ordered in decreasing order in MAXaffinityDSR and MAXAffinityDSL on the basis of the affinity value evaluated for transactions ($\text{affinity}_{sum}(T_{id})$).

$$\text{SORT}\left(T_{id}, \text{affinity}_{sum}\left(T_{id}\right), \text{decreasing} = \text{TRUE}\right). \tag{6}$$

Definition 4. All the candidate transactions identified for approaches are ordered in increasing order in MINaffinityDSR and MINAffinityDSL on the basis of $\text{affinity}_{sum}(T_{id})$ as described in Equation (5).

$$\text{SORT}\left(T_{id}, \text{affinity}_{sum}\left(T_{id}\right), \text{decreasing} = \text{FALSE}\right). \tag{7}$$

Definition 5. The candidate transaction for AffinityDSL is defined by:

$$T_{id}\left(S_r\right) = T_{id}\left(L_{S_r}\right) \cap T_{id}\left(R_{S_r}\right), \tag{8}$$

where L_{S_r} and R_{S_r} are the left-hand and right-hand side of association rule(S_r).

Definition 6. The transaction affinity in MAXAffinityDSL and MINAffinityDSL is calculated by

$$\text{affinity}_{sum}\left(T_{id}\right) = \sum_{\substack{for\ all\ 1-itemset\ m\ in\ T_{id} \\ and \\ m \subset F \\ and \\ m\ !=\ R_{S_r} \\ and \\ for\ all\ l\ in\ L_{S_r}}} \text{affinity}(m, l), \tag{9}$$

where F is set of frequent itemsets, T_{id} is a transaction, L_{S_r} and R_{S_r} are left-hand and right-hand side of association rule. It is calculated by summation of the affinity between every 1-itemset from the left-hand side of a sensitive association rule, and every frequent 1-itemset present in the transaction ignoring itemsets belongs to the right-hand side of the transaction.

2.3 Algorithm Parameters

In association rule hiding, the decision has to be taken in the following aspects which significantly affect the performance of the algorithm:

1. *Victim-item:* In the proposed work, first, two algorithms, MaxAffinityDSR and MinAffinityDSR, pick the victim item as a subset of the right-hand side of the rule whereas the next two algorithms, MaxAffinityDSL and MinAffinityDSL, pick victim item as a subset of the left-hand side of the rule. AffinityHybrid algorithm picks the victim item at runtime based on a minimum number of modifications required to hide the rule. All five sanitization algorithms remove the item from the database.
2. *Picking the candidate transactions:* In the proposed algorithms, candidate transactions are the ones that fully support the rule.
3. *Selecting transactions for modification from candidate transactions:* The primary decision of the association rule hiding algorithm falls in this step. All the candidate transactions are not modified. A subset of candidate transactions is altered until the rule's support or confidence is below the minimum support threshold and minimum confidence threshold, respectively. In proposed approaches, an affinity between frequent items is calculated and stored in a two-dimensional array in which dimensions are items. Then transaction affinity for each transaction is calculated as per the approach. Now candidate transactions are sorted by transaction affinity. The transaction affinity value is stored in a one-dimensional array.

2.4 Procedure

The procedure for MaxAffinityDSR, MinAffinityDSR, MaxAffinityDSL and MinAffinityDSL is as follows:

1. Generate association rule from the database using the Apriori algorithm.
2. The owner analyzes association rules generated and identifies sensitive association rules.
3. Repeat steps 4 to 10 for every sensitive association rule.
4. Calculate affinity between the items using Equation (3).
5. Pick candidate transactions like the ones which support all the items present in the sensitive association rule defined in Equations (4) and (8).
6. Calculate transaction affinity for all candidate transactions identified in step 5. The transaction affinity is calculated by adding the affinity of the victim item with every frequent 1-itemset present in the transaction. It is defined in Equations (5) and (9).
7. Sort candidate transactions by transaction affinity as defined in Equations (6) and (7).
8. Calculate the minimum number of modifications required to hide the rule.
9. Select the victim item as the right-hand side of the rule for AffinityDSR and the left-hand side of the rule for AffinityDSL respectively, which is going to be deleted from candidate transactions.

10. Pick one by one transaction from the sorted candidate transaction list prepared in step 5 and remove the victim item, till the number of modifications required is achieved.

2.5 MAXAffinityDSR and MINAffinityDSR Algorithm

```
input  : dataset, minsupp, minconf, sensitive association rules
output: Modified database to hide association rules
1  rulesdataset = apriori (dataset, parameter = list(supp = minsupp, conf = minconf,
     minlen = 2));
   // extract rules from dataset with user defined support and confidence threshold
2  inspect (rulesdataset);
   // user decides which rules are sensitive, needs to be hided
3  ruletohide = subset (rulesdataset);
   // selects sensitive rule in ruletohide
4  ruletohidec = |ruletohide |;
5  for o ← 1 to ruletohidec do
6      aff ← affinity(dataset);
       // calculate affinity among the frequent itemset
7      lhs ← selectLhs(ruletohide [o]);
8      rhs ← selectRhs(ruletohide [o]);
9      lhslist = {t | tεtransactions and t fully support LHS of rule};
10     rhslist = {t | tεtransactions and t fully support RHS of rule};
11     transactionDSR ← intersect(lhslist, rhslist);
       // prepare candidate transaction list in transactionDSR by intersecting
          lhslist and rhslist
12     tosorttransactionDSR ← NULL;
13     for i Input: transactionDSR
14      do
15         k ← o;
16         for j Input: items
17          do
18             if j ≠ lhs and j ≠ rhs and j is frequent and i contains j then
19                k = k + aff[rhs, j];
20             end
21             affinity(i) = k;
22             tosorttransactionDSR = concate(tosorttransactionDSR, concate(i, affinity(i)));
               // tosorttrnasactionDSR contain candidate transaction with their
                  respective affinity sum
23         end
24     end
25     sort(tosorttransactionDSR, sortby(affinity(i),i) Decreasing = FALSE);
       // Sorted Candidate Transactions
26     NoOfModificationinDSR
         ← minimum(ceiling((length(transactionDSR) − (TotalNumberOfTransaction ∗ MST
       /100))), ceiling((length(transactionDSR) − (length(lhslist) ∗ MCT /100))));
27     NoofModificationsDoneinDB ← 0;
28     for i ← 1 to NoOfModificationinDSR do
29         NoofModificationsDoneinDB ← NoofModificationsDoneinDB + 1;
30         pick transaction i from tosorttransactionDSR;
31         Dataset [i, rhs] ← 0;
32     end
33 end
```

Algorithm 1. MINAffinityDSR algorithm

The basis of the first two approaches, MAXAffinityDSR and MINAffinityDSR, select those transactions as victim transactions from the candidate transactions having a maximum and minimum value of transaction affinity respectively. In MAXAffinityDSR and MINAffinityDSR, the victim item is on the right-hand side of the sensitive association rule. Suppose there is a transaction t having items A, B, C, D, E. Suppose A is to be removed from transactions assuming it is present on the right-hand side of the sensitive rule $B \rightarrow A$. Let D and E be frequent itemsets and C is non-frequent itemsets. The affinity value of the transaction is identified as the sum of the affinity of A with D and E. Affinity of B is not considered while calculating the affinity of the transaction since the item belongs to the sensitive association rule which needs to be hidden. Affinity calculation with item C is not considered in the affinity value of the transaction since there does not exist any association rule having C on either side of the rule, as it is a non-frequent itemset. In this way all the transactions now have their affinity value then transactions are sorted by their affinity value in decreasing order in MAXAffinityDSR algorithm and in increasing order in case of the MINAffinityDSR algorithm. Then, one by one transactions are picked, and item present on the right-hand side of the rule is deleted from transaction till the confidence or support of the rule falls below the minimum threshold. The motivation for doing that is to reduce the side effects associated with the distorted database. This approach, MINAffinityDSR, is presented in Algorithm 1. In the MAXAffinityDSR algorithm, the only change is a selection of victim transactions from candidate transaction set; here candidate transactions are sorted in decreasing order of their affinity value.

Consider an example shown in Table 3 with user-defined support threshold of 55 % and user-defined confidence threshold is 80 %, following 14 association rules gets generated as shown in Table 4.

Transaction_Id	**Items**
1	ABCD
2	ABCD
3	ABC
4	ABCD
5	C
6	B
7	ABDEF

Table 3. Database D1

The affinity values of the items are calculated by Equation (3) as shown in Table 5.

Let $B \rightarrow A$ be the sensitive association rule that needs to be hided, then candidate transactions to be sanitized are identified as 1, 2, 3, 4, 7 since B and A both are present in this transactions. The affinity of these five transactions is calculated by adding the affinity of frequent items present in the transaction with the items of the right-hand side of a rule by Equation (5). So, the calculation of affinity of

S_No	LHS → RHS	Support	Confidence	Lift
1	$\{C\} \to \{A\}$	0.5714286	0.8	1.12
2	$\{A\} \to \{C\}$	0.5714286	0.8	1.12
3	$\{C\} \to \{B\}$	0.5714286	0.8	0.9333333
4	$\{D\} \to \{A\}$	0.5714286	1	1.4
5	$\{A\} \to \{D\}$	0.5714286	0.8	1.4
6	$\{D\} \to \{B\}$	0.5714286	1	1.1666667
7	$\{A\} \to \{B\}$	0.7142857	1	1.1666667
8	$\{B\} \to \{A\}$	0.7142857	0.8333333	1.1666667
9	$\{A, C\} \to \{B\}$	0.5714286	1	1.1666667
10	$\{B, C\} \to \{A\}$	0.5714286	1	1.4
11	$\{A, B\} \to \{C\}$	0.5714286	0.8	1.12
12	$\{A, D\} \to \{B\}$	0.5714286	1	1.1666667
13	$\{B, D\} \to \{A\}$	0.5714286	1	1.4
14	$\{A, B\} \to \{D\}$	0.5714286	0.8	1.4

Table 4. Association rules for sample databases D1

	A	B	C	D	E	F
A	0	0.8333333	0.6666667	0.8	0.2	0.2
B	0.8333333	0	0.5714286	0.6666667	0.1666667	0.1666667
C	0.6666667	0.5714286	0	0.5	0	0
D	0.8	0.6666667	0.5	0	0.25	0.25
E	0.2	0.1666667	0	0.25	0	1
F	0.2	0.1666667	0	0.25	1	0

Table 5. Affinity matrix

transactions is as follows:

- $\text{Affinity}(1) = \text{affinity}(A, C) + \text{affinity}(A, D) = 0.6666667 + 0.80 = 1.4666667$,
- $\text{Affinity}(2) = \text{affinity}(A, C) + \text{affinity}(A, D) = 0.6666667 + 0.80 = 1.4666667$,
- $\text{Affinity}(3) = \text{affinity}(A, C) = 0.6666667$,
- $\text{Affinity}(4) = \text{affinity}(A, C) + \text{affinity}(A, D) = 0.6666667 + 0.80 = 1.4666667$,
- $\text{Affinity}(7) = \text{affinity}(A, D) = 0.80$.

Note. E and F are not included in finding affinity of the transaction as they are non-frequent items and B is not considered since it belongs to sensitive association rule in the process of being hided.

For hiding the rule $B \to A$, item A, right-hand side of the rule, must be removed from transaction to reduce the support or confidence below the threshold. For reducing support below the threshold, number of modification required is 2, and for reducing confidence below the threshold, number of modification required is 1. Picking the minimum among these two is a sufficient and necessary number of modification required to successfully hide sensitive association rule.

Transaction_Id	Items
1	ABCD
2	ABCD
3	BC
4	ABCD
5	C
6	B
7	ABDEF

Table 6. Released Database D1

Since transaction affinity of Transaction 3 is minimum for MINAffinityDSR, so it is picked as the first transaction to be sanitized, and A is removed. Released database is shown in Table 6.

2.6 MAXAffinityDSL and MINAffinityDSL Algorithm

In this approach, sensitive association rules are hided by reducing the support of left-hand side thereby reducing the support of sensitive association rule. Same process as applied for hiding right-hand side of the rule in MAXAffintyDSR can be implemented with LHS in MAXAffinityDSL and MINAffinityDSL, i.e., the affinity of all the transaction is calculated by summing the affinity of the LHS with every frequent 1-itemset present in the transaction but ignoring the itemsets of the right-hand side of the rule. The insight for selecting the transaction based on affinity sum lies in the fact that more petite will be a side-effect of the modification if the similarity between the victim item and other frequent items is considered while selecting transactions. This approach is presented in Algorithm 2.

2.7 AffinityHybrid Algorithm

The third approach (Algorithm 3 AffinityHybrid) shown in Figure 1 combines the above two approaches to have a mixture of reducing the support of a subset of the left-hand side and right-hand side of the sensitive association rule.

In this method first, it is identified that for hiding sensitive association rule how many modifications are required to hide the rule on lowering the confidence below minimum confidence threshold and how many modifications are necessary to cover up the rule by reducing the support below the minimum support threshold. If the number of modification required is less for reducing confidence below MCT rather than reducing support below MST then victim item belongs to the right-hand side of the rule and removed from the transaction, we call it a hybrid(0) otherwise, items belongs to either left-hand side or right-hand side are removed as per HybridCode function. We call it a hybrid(1).

For hybrid(0), the approach is applied in the same way defined for the AffinityDSR approach.

```
input : Database, minsupp, minconf, sensitive association rules
output: Modified database to hide association rules
rulesdataset = apriori (dataset, parameter = list(supp = minsupp, conf = minconf,
  minlen = 2));
// extract rules from dataset with user defined support and confidence threshold
inspect (rulesdataset);
// user decides which rules are sensitive, needs to be hided
ruletohide = subset (rulesdataset);
// selects sensitive rule in ruletohide
ruletohidec = |ruletohide|;
for o ← 1 to ruletohidec do
    aff ← affinity(dataset);
    // calculate affinity among the frequent itemset
    lhs ← selectLhs(ruletohide [o]);
    rhs ← selectRhs(ruletohide [o]);
    lhslist = {t | tε transactions and t fully support LHS of rule};
    rhslist = {t | tε transactions and t fully support RHS of rule};
    transactionDSL ← intersect(lhslist, rhslist);
    // prepare candidate transaction list in TransactionDSL by intersecting
       lhslist and rhslist
    tosorttransactionDSL ← NULL;
    for i Input: transactionDSL
     do
        k ← o;
        for j Input: items
         do
            if j ≠ lhs and j ≠ rhs and j is frequent and i contains j then
              k = k + aff[lhs, j]
            end
            affinity(i) = k;
            tosorttransactionDSL = concate(tosorttransactionDSL, concate(i, affinity(i)))
               // TosorttrnasactionDSL contain candidate transaction with their
               respective affinity sum
        end
    end
    sort(tosorttransactionDSL, sortby((affinity(i), i)));
    // MINAffinityDSL and MAXAffinityDSL sort transactions in increasing and
       decreasing order respectively.
    NoOfModificationinDSL
      ← ceiling((length(transactionDSL) − (TotalNumberOfTransaction ∗ MST /100)));
    NoofModificationsDoneinDB ← 0;
    for i ← 1 to NoOfModificationinDSL do
        NoofModificationsDoneinDB ← NoofModificationsDoneinDB + 1;
        pick transaction i from tosorttransactionDSL;
        Dataset [i, lhs] ← 0;
    end
end
RulesNew = apriori (dataset, parameter = list(supp = minsupp, conf = minconf, minlen = 2));
inspect (RulesNew);
```

Algorithm 2. MAXAffinityDSL and MINAffinityDSL algorithm

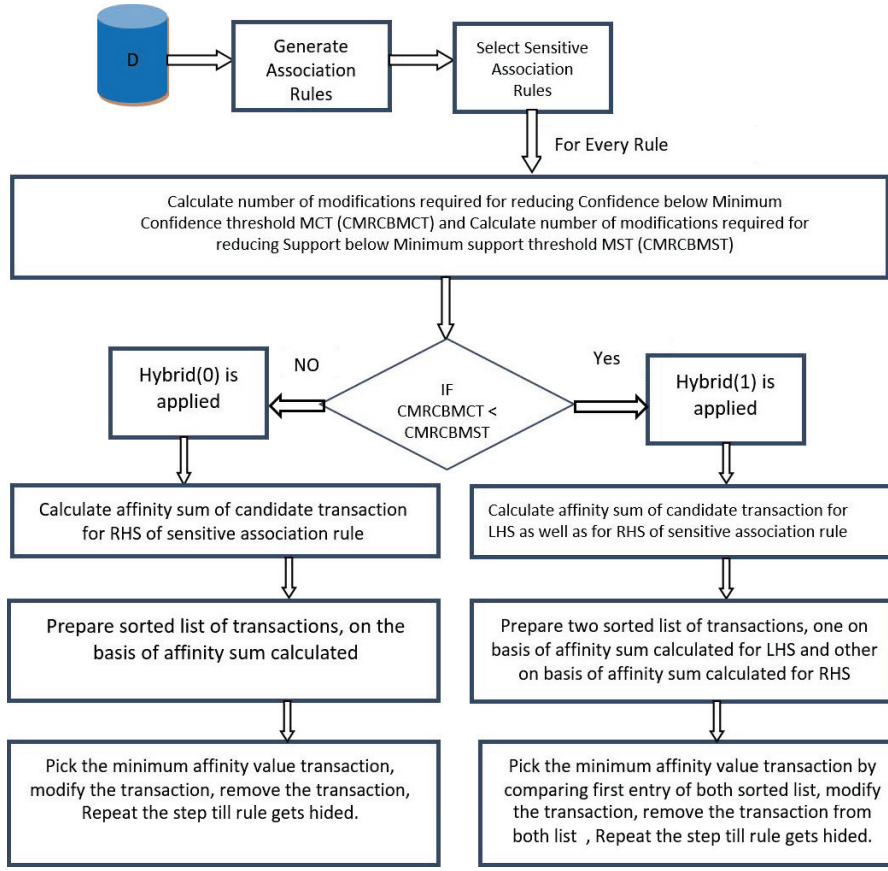

Figure 1. Hybrid Approach

For hybrid(1), the affinity of the transaction is divided into two parts:

1. Affinity of the transaction for left-hand side.
2. Affinity of the transaction for right-hand side.

The affinity of the transaction for the left-hand side is calculated by summing the affinity of the left-hand side with frequent items presenting in the transaction ignoring items belongs to the right-hand side of the transaction. The affinity of transactions for the right-hand side is calculated by summing the affinity of the right-hand side with frequent items present in the transaction ignoring the items belongs to the left-hand side. Then affinity of the transaction for the left-hand side is sorted on the basis of affinity calculated in increasing order. Similarly, affinity for the right-hand side is sorted on the basis of affinity calculated in the increasing order.

```
input  : Database, minsupp, minconf, sensitive association rules
output: Modified database to hide association rules
rulesdataset = apriori (Dataset, parameter = list(supp = minsupp, conf = minconf,
  minlen = 2));
inspect (rulesdataset);
ruletohide = subset (rulesdataset);
ruletohidec = |ruletohide|;
for o ← 1 to ruletohidec do
    aff ← affinity(dataset);
    lhs ← selectLhs(ruletohide [o]);
    rhs ← selectRhs(ruletohide [o]);
    lhslist = {t | tε transactions and t fully support LHS of rule};
    rhslist = {t | tε transactions and t fully support RHS of rule};
    TransactionHybrid ← intersect(lhslist, rhslist);
    // prepare candidate transaction list in TransactionHybrid by intersecting
       lhslist and rhslist
    TosorttransactionHybrid ← NULL;
    for i Input: TransactionHybrid
     do
        kl ← o;
        kr ← o;
        for all frequent items j Input: items
         do
            if j ≠ lhs and j ≠ rhs and i contains j then
                kl = kl + aff[lhs, j];
                kr = kr + aff[rhs, j];
            end
            TosorttransactionHybrid = concate(TosorttransactionHybrid, concate(i, kl, kr))
               // TosorttransactionHybrid contain candidate transaction with their
               respective affinity sum with lhs and affinity sum with rhs
        end
    end
    SortedLHSHybrid ← sort(TosorttransactionHybrid over (i, kl));
    NewMatrixOrderedLHS ← as(SortedLHSHybrid, "Matrix");
    // Sorted candidate transaction by affinity sum with LHS
    SortedRHSHybrid ← sort(TosorttransactionHybrid over (i, kr));
    NewMatrixOrderedRHS ← as(SortedRHSHybrid, "Matrix");
    // Sorted candidate transaction by affinity sum with RHS
    CMRCBMST ← ⌈(|TransactionHybrid| − (|Dataset| ∗ MST/100))⌉;
    CMRCBMCT ← ⌈(|TransactionHybrid| − (|lhslist| ∗ MCT/100))⌉;
    NoOfModificationinHybrid ← minimum(CMRCBMST, CMRCBMCT);
    if CMRCBMST ≤ CMRCBMCT then
        NoHybrid(NoOfModificationinHybrid, Dataset, SortedRHSHybrid, rhs)
    end
    else
        Call HybridCode(NoOfModificationinHybrid, Dataset, NewMatrixOrderedLHS,
          NewMatrixOrderedRHS, lhs, rhs)
    end
end
call CheckPerformance(Dataset, rulesdataset);
```

Algorithm 3. AffinityHybrid algorithm

```
1  kl ← 1;
   // Index for sorted transaction list having transaction ID and
      Transaction affinity with victim-item as LHS
2  kr ← 1;
   // Index for sorted transaction list having transaction ID and
      Transaction affinity with victim-item as RHS
3  NoofModificationsDoneinDB ← 0;
4  NoofModificationsDoneinLHS ← 0;
5  NoofModificationsDoneinRHS ← 0;
6  while NoofModificationsDoneinDB < NoOfModificationinHybrid do
       // Transaction is selected by comparing the two sorted list
          prepared
7      if NewMatrixOrderedLHS [kl, 2] < NewMatrixOrderedRHS [kr, 2] then
8          Dataset [(NewMatrixOrderedLHS [kl, 1]),Lhs ] ← 0;
           // transaction picked from sorted list prepared by having
              victim-item as LHS
9          x ← which(NewMatrixOrderedRHS [, 1] == NewMatrixOrderedLHS
            [kl, 1]);
10         NewMatrixOrderedRHS ← NewMatrixOrderedRHS [−x, ];
           // Transaction removed from sorted list prepared by
              having victim-item as RHS, as it is already sanitized
11         NoofModificationsDoneinLHS ← NoofModificationsDoneinLHS + 1;
12         NoofModificationsDoneinDB ← NoofModificationsDoneinDB + 1;
13         kl ← kl + 1;
14     end
15     else
16         Dataset [(NewMatrixOrderedRHS [kr, 1]), Rhs ] ← 0;
           // transaction picked from sorted list prepared by having
              victim-item as RHS
17         x ← which(NewMatrixOrderedLHS [, 1] == NewMatrixOrderedRHS
            [kr, 1]);
18         NewMatrixOrderedLHS ← NewMatrixOrderedLHS [−x, ];
           // Transaction removed from sorted list prepared by
              having victim-item as LHS, as it is already sanitized
19         NoofModificationsDoneinRHS ← NoofModificationsDoneinRHS + 1;
20         NoofModificationsDoneinDB ← NoofModificationsDoneinDB + 1;
21         kr ← kr + 1;
22     end
23 end
```

Algorithm 4. HybridCode algorithm

```
for i ← 1 to NoOfModificationinHybrid do
    NoofModificationsDoneinDB ← NoofModificationsDoneinDB + 1;
    Dataset [m in SortedRHSHybrid, Rhs ] ← 0;
end
Return(Dataset);
```

Algorithm 5. NoHybrid algorithm

```
RulesNew = Apriori (dataset, parameter = list(supp = minsupp,
  conf = minconf, minlen = 2));
Inspect (RulesNew);
GhostRules ← SetDiff(RulesNew, rulesdataset);
LostRules ← SetDiff(rulesdataset, RulesNew);
GhostRulesCount ← Length(SetDiff(RulesNew, rulesdataset));
LostRulesCount ← Length(SetDiff(rulesdataset,
  RulesNew)) − ruletohidec;
```

Algorithm 6. Performance After Hybrid algorithm

Let x be the number of modification required for hybrid(1). Then one by one transaction is selected by comparing the two sorted lists prepared, and transaction having the least affinity is modified with the condition that if transaction has been picked from sorted affinity list of the transaction for the left-hand side then the left-hand side of the rule gets removed from transaction otherwise if the transaction is picked from sorted affinity of the transaction for the right-hand side, then the right-hand side of the rule gets removed. Once the transaction picked from the one list, it cannot be picked again from the second list since if the same transaction is used for removing the left-hand side and the right-hand side – this does not add any benefit to the approach.

Let X and Y be present on the left-hand side and the right-hand side of the rule. For reducing the support below MST either X or Y is sufficient to remove the transaction to reduce the support of the overall rule. The rationale for a hybrid approach is that changing only one side of the rule would result in a large reduction in its support, which will have more unintended consequences that can be managed by taking into account both the left-hand and right-hand sides of the rule. Additionally, the side-effect is diminished because the choice of transaction is fully chosen based on how it would affect another frequently used itemset.

3 COMPUTATIONAL EXPERIMENTS AND RESULTS

Approaches present in the literature for hiding the rule fall into two broad categories viz

1. hiding a large itemset,

2. hiding an association rule directly.

It is more complicated to hide rules in comparison to hiding itemsets. This paper presents five approaches that hide sensitive association rule by directly working on hiding rules as it gives the database owner more control. Latest work that has been done on association rule hiding hides large itemsets to preserve the privacy in the database like the border based approach. Greedy approaches [26] (2013) hides a sensitive association rule by increasing the number of transactions that will greatly affect the database size as well as a lot of computation needed to add the items to the added transaction. So, we evaluate our work against algorithms that fall under the category of heuristic algorithms. The algorithms used for comparison are Algo 1.a [4], Algo 1.b [4], MinFIA [2], MaxFIA [2] and Naive [2]. To evaluate performance, we ran two experiments. In the first setup, experiments were performed with dataset generated by IBM Synthetic data generator where database size is ranging from 10 K to 100 K. In the second setup, experiments were performed with real-world datasets downloaded from UCI repository and fimi.

3.1 Experiment Setup 1

We have performed experiments on a computer with a core-i7 processor, 8 GB RAM running on Windows 10 Operating System. The datasets used in the assessment trials are generated using IBM synthetic data generator [27]. The database size employed in the dataset range from 10 K to 100 K with the average transaction length, $|\text{ATL}| = 5$, and a total number of items is 50. The minimum support threshold picked is 4 % and the minimum confidence threshold picked is 20 %. In the series of experiments, database size ranging from 10 K to 100 K is generated ten times and arbitrary five rules are selected for hiding. All the graphs were plotted to represent the average of 10 iterations of experiments. The language used for implementation is *R* [28]. To evaluate the performance of the algorithms following side-effects are considered:

1. Rule Hiding Failure (RHF);
2. Rule Falsely Generated or Ghost Rules (GR) (Also known as Artifactual Patterns (AF));
3. Rules Falsely Hidden or Lost Rules (LR);
4. Dissimilarity measure.

The rule hiding failure side-effect counts the number of sensitive association rules; the algorithm fails to hide. Rule falsely generated (Ghost rules) side-effect counts the number of rules that were not available with the original dataset, but after the modifications performed by the algorithm, the rule appears. The rules falsely hidden (Lost rules) side-effect counts the number of nonsensitive rules hided because of the data distortion process. Comparisons were made with Algo1.a, Algo1.b, MinFIA, MaxFIA and naive algorithm against various database sizes ranging from 10 K to 100 K.

Hiding failure is 0 in all algorithms except Algo 1.a. So Algo 1.a is not considered in analyzing other side-effects.

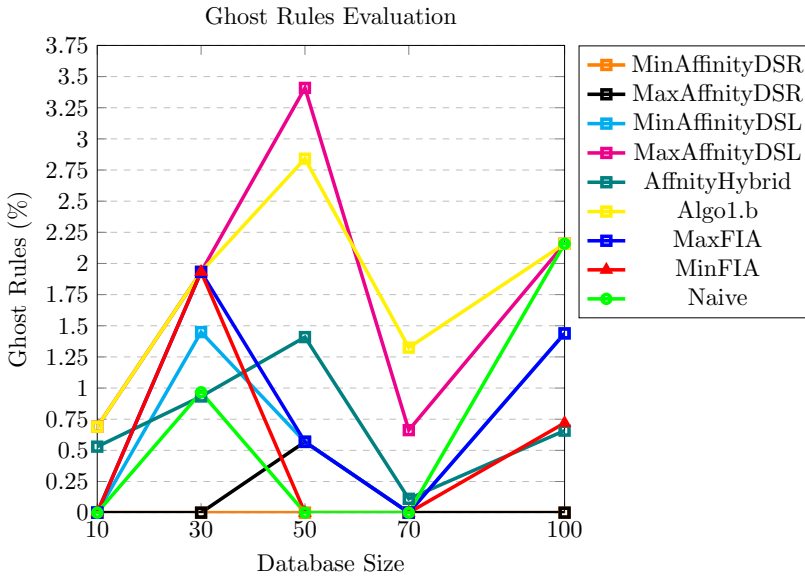

Figure 2. Performance of Algorithms with respect to Ghost Rules

Figures 2 and 3 account for the ghost rule and lost rule side-effect evaluation of algorithms, respectively. Table 7 represents data for the ghost rule and lost rule side-effect in a tabular form, where each average value is accompanied by a value of the standard error.

It is depicted in Figure 2 that MinAffinityDSR algorithm is free from the ghost rule side-effect. It never generates ghost rules in all our experiment trials. MaxAffnityDSR also performs well in the ghost rule side-effect. It suffered from this side-effect only once.

It is clear from Figure 3 that MinAffinityDSR, MaxAffinityDSL and AffinityHybrid algorithm performs best with lost rules side-effects. Less rules lost means more is the utility of the database.

Dissimilarity measure is based upon the count of the frequency of the items before the sanitization algorithm and after the sanitization algorithm, i.e., to measure the frequencies of the items in the original database and the released database. Dissimilarity measure evaluation is shown in Figure 4. Table 8 represents data for Dissimilarity measure evaluation in a tabular form, where each average value is accompanied by a value of the standard error. It is clear from the graph that AffinityHybrid outperforms all algorithms used in experiments for comparison. MinAffinityDSR, MaxAffinityDSR and MaxAffinityDSL also has good results and Naive algorithm performance was the weakest with respect to dissimilarity measure.

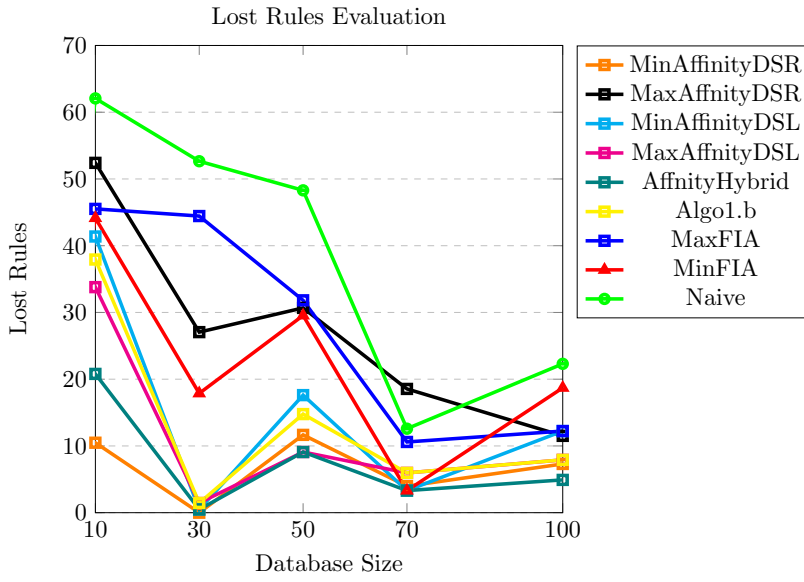

Figure 3. Performance of Algorithms with respect to Lost Rules

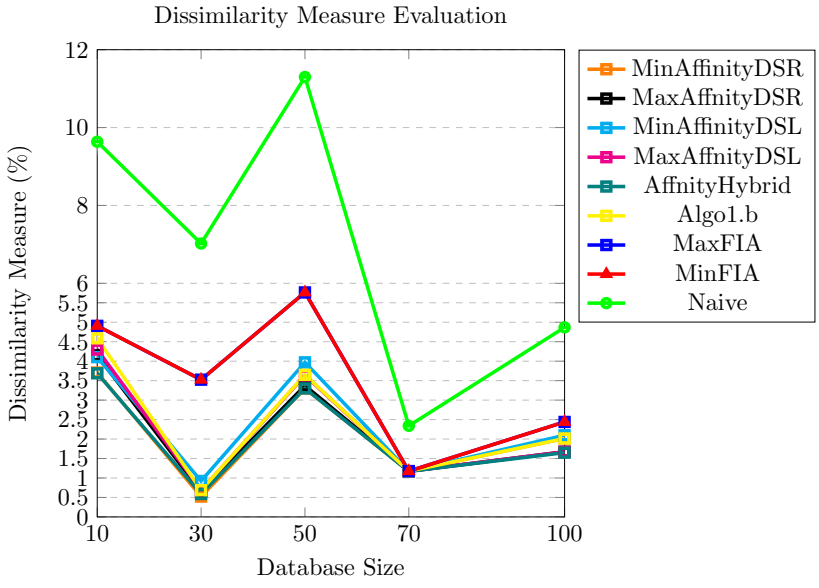

Figure 4. Performance of Algorithms with Dissimilarity Measure

Algorithm	Database Size	Ghost Rules (GR)	Standard Error (GR)	Lost Rules (LR)	Standard Error (LR)
MinAffinityDSR	10	0	0	10.48	2.82
	30	0	0	0	0.2
	50	0	0	11.66	3.2
	70	0	0	3.97	0.67
	100	0	0	7.3	1.56
MaxAffinityDSR	10	0	0	52.41	4.81
	30	0	0	27.05	3.22
	50	0.57	0.01	30.68	1.56
	70	0	0	18.54	3.39
	100	0	0	11.51	1.22
MinAffinityDSL	10	0	0	41.38	4.29
	30	1.45	0.11	0.48	0.55
	50	0.57	0.12	17.61	4.99
	70	0	0	3.31	2.23
	100	1.44	0.49	12.23	4.81
MaxAffinityDSL	10	0.69	0.66	33.79	1.64
	30	1.93	1.21	1.45	2.59
	50	3.41	0.23	9.09	3.58
	70	0.66	0.47	5.96	2.47
	100	2.16	1.09	7.91	3.29
Affinityhybrid	10	0.53	0.04	20.79	1.01
	30	0.93	0.37	0.48	1.2
	50	1.41	0.01	9.09	0.25
	70	0.11	0.01	3.31	1.23
	100	0.66	0.78	4.91	0.45
Algo 1.b	10	0.69	0.16	37.93	0.68
	30	1.93	0.19	1.45	1.26
	50	2.84	1.01	14.77	2.32
	70	1.32	1.07	5.96	1.67
	100	2.16	1.22	7.91	0.44
MaxFIA	10	0	0	45.52	4.25
	30	1.93	1.19	44.44	2.25
	50	0.57	0.09	31.82	3.29
	70	0	0	10.6	4.17
	100	1.44	0	12.23	2.38
MinFIA	10	0	0	44.14	1.82
	30	1.93	0.08	17.87	0.17
	50	0	0	29.55	3.68
	70	0	0	3.31	4.25
	100	0.72	0.49	18.71	1.19
Naive	10	0	0	62.07	2.28
	30	0.97	0.11	52.66	4.28
	50	0	0	48.3	4.66
	70	0	0	12.58	2.38
	100	2.16	0.19	22.3	2.87

Table 7. Performance of Algorithms – Ghost Rules and Lost Rules (With Standard Error)

3.2 Experiment Setup 2

We have performed performance evaluation experiments on a PC with a core-i7 processor, 8 GB RAM running on Windows 10 Operating System and the language used for implementation is R Language.

We tested the proposed algorithm on three real representative databases. One is a mushroom [29] (descriptions of hypothetical samples corresponding to 23 species

Algorithm	Database Size	Dissimilarity	Standard Error
MinAffinityDSR	10	3.70819848975189	1.52
	30	0.52254906665387	2.31
	50	3.30037822918644	0.65
	70	1.17524852996062	0.58
	100	2.01695689527802	0.29
MaxAffinityDSR	10	4.15110779188449	2.28
	30	0.64602428139546	3.47
	50	3.37266527447172	1.69
	70	1.17055066774891	2.59
	100	1.67285940210462	3.57
MinAffinityDSL	10	4.10339390921915	1.22
	30	0.92254906665387	0.52
	50	3.96994611329351	0.47
	70	1.17524852996062	1.66
	100	2.09695689527802	2.59
MaxAffinityDSL	10	4.29839847315575	3.69
	30	0.68561351312443	2.48
	50	3.63516217123998	3.58
	70	1.17524852996062	2.46
	100	1.67285940210462	3.46
Affinityhybrid	10	3.68819848975189	2.59
	30	0.5882	3.57
	50	3.30037822918644	2.58
	70	1.17055066774891	2.69
	100	1.65	4.24
Algo 1.b	10	4.58675628578541	0.66
	30	0.687413023657565	2.25
	50	3.64830527038276	3.28
	70	1.17524852996062	2.69
	100	2.01414387940674	2.47
MaxFIA	10	4.90726910629823	2.56
	30	3.52464129756706	3.58
	50	5.76507440454459	3.98
	70	1.17524852996062	2.58
	100	2.43897100267592	3.69
MinFIA	10	4.90726910629823	3.57
	30	3.52464129756706	2.49
	50	5.76507440454459	3.74
	70	1.17524852996062	4.25
	100	2.43897100267592	1.12
Naive	10	9.63612978176085	2.28
	30	7.02169010029272	3.48
	50	11.2986842297414	1.69
	70	2.33953538142726	2.49
	100	4.86772718401089	1.66

Table 8. Performance of Algorithms – Dissimilarity (With Standard Error)

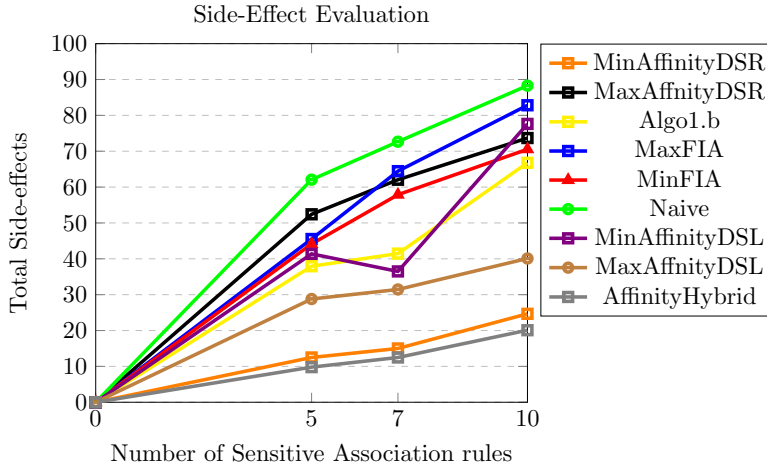

Figure 5. Performance on real datasets – Mushroom, BMS-WebView-1, BMS-WebView-2

of gilled mushrooms in the Agaricus and Lepiota Family (pp. 500–525). Each species is identified as definitely edible, definitely poisonous, or of unknown edibility and not recommended), which was prepared by Roberto Bayardo and is publicly available through the FIMI repository website located at `http://fimi.ua.ac.be/data/` (Frequent Itemset Mining Dataset Repository). The other datasets were BMS-WebView-1 (downloaded from [30]) from Blue Martini Software Inc. that were used for the KDD Cup of 2000. This dataset contains 59,601 sequences of clickstream data from an e-commerce. It contains 497 distinct items. The average length of sequences is 2.42 items with a standard deviation of 3.22. In this dataset, there are some long sequences. For example, 318 sequences contain more than 20 items. Another dataset used is BMS-WebView-2 (downloaded from [30]) which is a second dataset used in the KDD-CUP 2000 competition. It contains 77 512 sequences of click-stream data. It contains 3340 distinct items. The average length of sequences is 4.62 items with a standard deviation of 6.07 items. The thresholds of minimum support were appropriately set to ensure an adequate amount of frequent itemsets. Comparisons were made with state-of-art approaches (Algo 1.a and Algo 1.b), MinFIA, MaxFIA and Naive, and the results are summarized in Figure 5. The graph represents the average of total side-effects generated on all three datasets when number of sensitive association rule is varied from 0 to 10. The result obtained is similar to the case when experiments are performed with datasets generated from IBM Synthetic data generator. AffinityHybrid algorithm gives the best solution. It can also be concluded that while hiding the rule by reducing the support of right-hand side of rule, transactions must be selected in increasing value of affinity, i.e., MinAffinityDSR is preferred. If the rule is to be hided by decreasing the support of the left-hand side of the rule, transactions must be selected in decreasing value of affinity, i.e., MaxAffinityDSL is pre-

ferred.

4 ANALYSIS AND DISCUSSION

4.1 Accuracy

In association rule hiding techniques, the accuracy of the heuristic algorithms depends on the victim item as well as the victim transaction. In [4] approaches selects the victim transaction by count of itemsets present in the transaction. Picking the transaction in this fashion can significantly increase side-effects. Let there are four transactions all having precisely four itemsets, then as per [4], all four will be considered equally as the candidate transaction to be modified.

Proposed strategy while selecting a potential transaction, takes into account the effects of modifying it. This is done by the concept of affinity of transaction introduced in the paper. Also in [4], approaches either select victim item from the left-hand side of the rule or right-hand side of the rule.

In proposed hybrid approach, victim item, as well as victim transaction both are selected on the basis of transaction affinity calculated exclusively for the left-hand side as well as the right-hand side of the sensitive association rule.

4.2 Effect of MST and MCT

For hiding sensitive association rule, either support of the rule should be below the minimum support threshold (MST), or confidence should be below minimum confidence threshold (MCT). The number of modification increases as MST and MCT selected by the data owner decreases.

It is identified by experimental results that proposed approaches performed better not only with the higher value of MST and MCT but also with low range values. Although the side-effect may get an increase, as the too low value of MST and MCT is considered by data owner, while the optimal sanitization in association rule hiding belongs to the class of NP-Complete problems.

4.3 Number of Modifications

1. The number of modifications to be done in MAXAffinityDSR and MINAffinityDSR can be identified by:

$$NM_{DSR} = \left\lceil MIN(|\mathbb{T}_{id}(S_r)| - \frac{|\mathbb{T}_{id}| * MST}{100}, |\mathbb{T}_{id}(S_r)| - \frac{|\mathbb{T}_{id}(L_{S_r})| * MCT}{100}) \right\rceil . \tag{10}$$

   $\mathbb{T}_{id}(S_r)$ or $(\sum X \cup Y)$ and $\mathbb{T}_{id}(L_{S_r})$ contains the set of all transactions containing X ∪ Y and X respectively. $|\mathbb{T}_{id}|$ is the total number of transactions. To hide X → Y, removing items in Y from the transactions will decrease $\text{support}_{X \cup Y}$ i.e., it

will reduce the number of transactions supporting the rule by deleting elements from transactions present on the right-hand side of the rule. Let NM_{DSR} (θ), an integer, be number of modified transactions when the rule $X \rightarrow Y$ is hidden. This make either support is reduced below MST as defined in Equation 11 or confidence reduced below MCT as defined in Equation 12 which is sufficient to hide the sensitive association rule.

$$\frac{|\mathfrak{T}_{id}(S_r)| - \theta}{|\mathfrak{T}_{id}|} < \frac{MST}{100}, \tag{11}$$

$$\frac{|\mathfrak{T}_{id}(S_r)| - \theta}{|\mathfrak{T}_{id}(L_{S_r})|} < \frac{MCT}{100}. \tag{12}$$

Hence, the number of modifications required is minimum value of θ to satisfy either Equations (11) or (12).

2. The number of modifications in MAXAffinityDSL and MINAffinityDSL is identified by:

$$NM_{DSL} = \left\lceil |\mathfrak{T}_{id}(S_r)| - \frac{|\mathfrak{T}_{id}| * MST}{100} \right\rceil. \tag{13}$$

$\mathfrak{T}_{id}(S_r)$ or $(\sum X \cup Y)$ contains the set of all transactions containing $X \cup Y$. To hide $X \rightarrow Y$, removing items in X from the transactions will decrease the $\text{support}_{X \cup Y}$ i.e., it will reduce the number of transactions supporting the rule by deleting elements from transactions present on left-hand side of rule. Let NM_{DSL} (θ), an integer, be number of modified transactions when rule $X \rightarrow Y$ is hidden. This makes support to reduce below MST as defined in Equation (14) which eventually hide the sensitive association rule.

$$\frac{|\mathfrak{T}_{id}(S_r)| - \theta}{|\mathfrak{T}_{id}|} < \frac{MST}{100}. \tag{14}$$

3. The third approach AffinityHybrid combines the above two approaches to have a mixture of reducing the support of a subset of the left-hand side and right-hand side of the sensitive association rule. Therefore, number of modification can be identified by:

$$NM_H = \left\lceil MIN\left(|\mathfrak{T}_{id}(S_r)| - \frac{|\mathfrak{T}_{id}| * MST}{100}, |\mathfrak{T}_{id}(S_r)| - \frac{|\mathfrak{T}_{id}(L_{S_r})| * MCT}{100} \right) \right\rceil. \tag{15}$$

4.4 Results Summary

- MinAffinityDSR algorithm never generates ghost rules.

- Algo 1.b perform worst in case of ghost rules side-effects.
- MaxAffinityDSR and MinAffinityDSR generate good results with dissimilarity measure.
- Ghost rule percentage is low in comparison to lost rule percentage in case of all algorithms as the maximum percentage of ghost rule side effect is under 3.5. This shows that lost rule plays a primary concern in evaluating the performance of the algorithm.
- MaxAffinityDSL and MinAffinityDSL performance were better in comparison to Algo 1.a, Algo 1.b, MinFIA, MaxFIA and naive algorithm regarding lost rule side-effect. Also, both perform better with dissimilarity measure.
- AffinityHybrid algorithm outperforms all the algorithms used for comparison with the lost rule, ghost rule and dissimilarity measure.

5 CONCLUSION

This work proposes five new algorithms based on modifying transactions by considering the side effect, by calculating affinity sum of victim items with other frequent items present in the transaction. The experimental result clearly shows the fruitfulness of the approach. Experiments suggest that proposed approach not only outperforms Algo 1.a, Algo 1.b, MINFia, MaxFIA and NAive algorithm regarding dissimilarity measure but at the same time, side-effects have been reduced. A drawback of the proposed algorithm is its running time. Algorithm performs a bit slower in comparison to other algorithms when experiments performed with large databases. The reason for a slow speed is that the affinity calculation time increases with database size. Among the five algorithms presented in the paper, AffinityHybrid algorithm gives the best solution. It can also be concluded that while hiding the rule by reducing the support of the right-hand side of the rule, transactions must be selected in the increasing value of affinity, i.e., MinAffinityDSR is preferred. If the rule is to be hided by the decreasing the support of the left-hand side of the rule, transactions must be selected in decreasing value of affinity, i.e., MaxAffinityDSL is preferred. If there is no restriction on selecting the victim item from the rule, the AffinityHybrid algorithm is preferred as it is the top performer among the set of algorithms discussed in the paper.

REFERENCES

[1] Clifton, C.—Marks, D.: Security and Privacy Implications of Data Mining. ACM SIGMOD Workshop on Research Issues on Data Mining and Knowledge Discovery, Citeseer, 1996, pp. 15–19.

[2] Oliveira, S. R. M.—Zaiane, O. R.: Privacy Preserving Frequent Itemset Mining. In: Clifton, C., Estivill-Castro, V. (Eds.): IEEE ICDM Workshop on Privacy, Security and Data Mining. ACS, Maebashi City, Japan, CRPIT, Vol. 14, 2002, pp. 43–54.

[3] Sun, X.—Yu, P.: A Border-Based Approach for Hiding Sensitive Frequent Itemsets. Fifth IEEE International Conference on Data Mining (ICDM'05), 2005, 8 pp., doi: 10.1109/ICDM.2005.2.

[4] Verykios, V.—Elmagarmid, A.—Bertino, E.—Saygin, Y.—Dasseni, E.: Association Rule Hiding. IEEE Transactions on Knowledge and Data Engineering, Vol. 16, 2004, No. 4, pp. 434–447, doi: 10.1109/TKDE.2004.1269668.

[5] Oliveira, S.—Zaiane, O.: Protecting Sensitive Knowledge by Data Sanitization. Third IEEE International Conference on Data Mining, 2003, pp. 613–616, doi: 10.1109/ICDM.2003.1250990.

[6] Pontikakis, E.—Tsitsonis, A.—Verykios, V.: An Experimental Study of Distortion-Based Techniques for Association Rule Hiding. Research Directions in Data and Applications Security XVIII, 2004, pp. 325–339, doi: 10.1007/1-4020-8128-6_22.

[7] Jain, D.—Khatri, P.—Soni, R.—Chaurasia, B. K.: Hiding Sensitive Association Rules Without Altering the Support of Sensitive Item(s). Advances in Computer Science and Information Technology. Networks and Communications, 2012, pp. 500–509, doi: 10.1007/978-3-642-27299-8_52.

[8] Lin, Y.—Wang, E.—Lee, G.: A Novel Method for Protecting Sensitive Knowledge in Association Rules Mining. Proceedings of the 29th Annual International Computer Software and Applications Conference (COMPSAC 2005), IEEE Computer Society, Los Alamitos, CA, USA, Vol. 1, 2005, pp. 511–516, doi: 10.1109/COMPSAC.2005.27.

[9] Sun, X.—Yu, P. S.: Hiding Sensitive Frequent Itemsets by a Border-Based Approach. Journal of Computing Science and Engineering, Vol. 1, 2007, No. 1, pp. 74–94, doi: 10.5626/jcse.2007.1.1.074.

[10] Moustakides, G. V.—Verykios, V. S.: A Maxmin Approach for Hiding Frequent Itemsets. Data and Knowledge Engineering, Vol. 65, 2008, No. 1, pp. 75–89, doi: 10.1016/j.datak.2007.06.012 (Including Special Section: Privacy Aspects of Data Mining Workshop (2006) – Five Invited and Extended Papers).

[11] Gkoulalas-Divanis, A.—Verykios, V.: A Hybrid Approach to Frequent Itemset Hiding. 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007), Vol. 1, 2007, pp. 297–304, doi: 10.1109/ICTAI.2007.68.

[12] Quoc Le, H.—Arch-Int, S.—Arch-Int, N.: Association Rule Hiding Based on Intersection Lattice. Mathematical Problems in Engineering, Vol. 2013, 2013, doi: 10.1155/2013/210405.

[13] Gkoulalas-Divanis, A.—Verykios, V. S.: An Integer Programming Approach for Frequent Itemset Hiding. Proceedings of the 15th ACM International Conference on Information and Knowledge Management – CIKM'06, ACM Press, 2006, doi: 10.1145/1183614.1183721.

[14] Guo, Y.: Reconstruction-Based Association Rule Hiding. Proceedings of SIGMOD 2007 Ph.D. Workshop on Innovative Database Research, Vol. 2007, 2007, pp. 51–56.

[15] Chen, X.—Orlowska, M.—Li, X.: A New Framework of Privacy Preserving Data

Sharing. Proceedings of the 4th IEEE ICDM Workshop: Privacy and Security Aspects of Data Mining. IEEE Computer Society, Citeseer, 2004, pp. 47–56.

[16] Wang, Y.—Wu, X.: Approximate Inverse Frequent Itemset Mining: Privacy, Complexity, and Approximation. Fifth IEEE International Conference on Data Mining (ICDM '05), 2005, 8 pp., doi: 10.1109/ICDM.2005.27.

[17] Guo, Y.—Tong, Y.—Tang, S.—Yang, D.: A FP-Tree-Based Method for Inverse Frequent Set Mining. In: Bell, D. A., Hong, J. (Eds.): Flexible and Efficient Information Handling. Springer Berlin Heidelberg, Berlin, Heidelberg, 2006, pp. 152–163, doi: 10.1007/11788911_13.

[18] Ahmed, G.—Abd_Ellatif, L.—Sharaf, A.: Association Rules Hiding for Privacy Preserving Data Mining: A Survey. International Journal of Computer Applications, Vol. 150, 2016, No. 12, pp. 34–43, doi: 10.5120/ijca2016911664.

[19] Vaidya, J.—Clifton, C.: Privacy Preserving Association Rule Mining in Vertically Partitioned Data. Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Association for Computing Machinery, New York, NY, USA, KDD '02, 2002, pp. 639–644, doi: 10.1145/775047.775142.

[20] Vaidya, J.—Clifton, C.: Secure Set Intersection Cardinality with Application to Association Rule Mining. J. Comput. Secur., Vol. 13, 2005, No. 4, pp. 593–622, doi: https://dl.acm.org/doi/10.5555/1239367.1239368.

[21] Zhong, S.: Privacy-Preserving Algorithms for Distributed Mining of Frequent Itemsets. Information Sciences, Vol. 177, 2007, No. 2, pp. 490–503, doi: 10.1016/j.ins.2006.08.010.

[22] El-Sisi, A.: Fast Cryptographic Privacy Preserving Association Rules Mining on Distributed Homogenous Database. Int. Arab J. Inf. Technol., Vol. 7, 2010, No. 2, pp. 152–160.

[23] Kaosar, M. G.—Paulet, R.—Yi, X.: Fully Homomorphic Encryption Based Two-Party Association Rule Mining. Data and Knowledge Engineering, Vol. 76–78, 2012, pp. 1–15, doi: 10.1016/j.datak.2012.03.003.

[24] Alborzi, S.—Raji, F.—Saraee, M.: Privacy Preserving Mining of Association Rules on Horizontally Distributed Databases. International Conference on Software and Computer Applications ICSCA 2012, IACSIT Press, Singapore, 2012, pp. 158–164, http://usir.salford.ac.uk/id/eprint/42930/.

[25] Aggarwal, C.—Procopiuc, C.—Yu, P.: Finding Localized Associations in Market Basket Data. IEEE Transactions on Knowledge and Data Engineering, Vol. 14, 2002, No. 1, pp. 51–62, doi: 10.1109/69.979972.

[26] Lin, C. W.—Hong, T. P.—Chang, C. C.—Wang, S. L.: A Greedy-Based Approach for Hiding Sensitive Itemsets by Transaction Insertion. J. Inf. Hiding Multim. Signal Process., Vol. 4, 2013, No. 4, pp. 201–214.

[27] Agrawal, R.—Srikant, R.: Fast Algorithms for Mining Association Rules in Large Databases. In: Bocca, J. B., Jarke, M., Zaniolo, C. (Eds.): VLDB '94, Proceedings of 20th International Conference on Very Large Data Bases, September 12–15, 1994, Santiago De Chile, Chile. Morgan Kaufmann, 1994, pp. 487–499, http://www.vldb.org/conf/1994/P487.PDF.

[28] R Core Team: R: A Language and Environment for Statistical Computing. R Foun-

dation for Statistical Computing, Vienna, Austria, 2017, https://www.R-project.org/.

[29] Mushroom. 1987, doi: 10.24432/C5959T.

[30] Fournier-Viger, P.—Lin, J. C. W.—Gomariz, A.—Gueniche, T.—Soltani, A.—Deng, Z.—Lam, H. T.: The SPMF Open-Source Data Mining Library Version 2. In: Berendt, B., Bringmann, B., Fromont, É., Garriga, G., Miettinen, P., Tatti, N., Tresp, V. (Eds.): Machine Learning and Knowledge Discovery in Databases. Springer International Publishing, Cham, 2016, pp. 36–40, doi: 10.1007/978-3-319-46131-1_8.

Kshitij Pathak is a Lecturer in the Department of Computer Science and Engineering at Government Polytechnic College Mandsaur, MP, India. He has been awarded a Ph.D. degree in computer science and engineering. He has published 42 papers in International and National journals and conferences. His research interest includes data mining, information security and artificial intelligence. He is a life member of the Computer Society of India and the Institution of Engineers India.

Sanjay Silakari is Professor in the Department of Computer Science and Engineering at the University Institute of Technology, RGPV Bhopal, MP, India. He has more than two decades of teaching and administrative experience and has guided several students in their doctoral and master's studies. He has several research publications to his credit in different reputed national and international conferences and journals. His areas of interest include network security, web engineering, web personalization and search engines, operating systems, computer networks and e-commerce. He is a life member of ISTE, CSI, and IAENG and a member of IEEE and ACM. He is the author of the book Basic Computer Engineering.

Narendra S. Chaudhari is Professor (HAG) in the Computer Science and Engineering Department at Indian Institute of Technology (IIT), Indore (M.P.), India. He has done significant research work on game AI, novel neural network models and security of the wireless mobile communication. He has been referee and reviewer for a number of premier conferences and Journals including IEEE Transaction, Neurocomputing, etc. Also, he is fellow and recipient of Eminent Engineer Award (Computer Engineering) of the Institution of Engineers, India (IE-India), as well as fellow of the Institution of Electronics and Telecommunication Engineers (IETE) (India), senior member of Computer Society of India, senior member of IEEE (USA), member of Indian Mathematical Society (IMS), Cryptology Research Society of India (CRSI) and many other professional societies.