An Optimal Approach For Knowledge Protection In Structured Frequent Patterns

Cynthia Selvi P *pselvi1501@gmail.com*
Associate Professor, Dept. of Computer Science
Kunthavai Naacchiyar Govt. Arts College for Women(Autonomous), Thanjavur 613007
Affiliated to Bharathidasan University, Tiruchirapalli, TamilNadu, India.

Mohamed Shanavas A.R *vas0699@yahoo.co.in*
Associate Professor, Dept. of Computer Science,
Jamal Mohamed College, Tiruchirapalli 620 020
Affiliated to Bharathidasan University, Tiruchirapalli, TamilNadu, India.

Abstract

Data mining is valuable technology to facilitate the extraction of useful patterns and trends from large volume of data. When these patterns are to be shared in a collaborative environment, they must be protectively shared among the parties concerned in order to preserve the confidentiality of the sensitive data. Sharing of information may be in the form of datasets or in any of the structured patterns like trees, graphs, lattices, etc., This paper propose a sanitization algorithm for protecting sensitive data in a structured frequent pattern(tree).

Keywords: Rank Function, Restricted Node, Sanitization, Structured Pattern, Victim States.

1. INTRODUCTION

Data mining is an emerging technology to provide various means for identifying the interesting and important knowledge from large data collections. When this knowledge is to be shared among various parties in decision making activities, the sensitive data is to be preserved by the parties concerned. In particular, when multiple companies want to share the customer's buying behavior in a collaborative business environment that promote business, the sensitive information of the individual company should be protected against sharing. The information to be shared may be in the form of datasets, frequent itemsets, structured patterns or subsequences. Here structured pattern refers to substructures like graphs, trees or lattices that contain frequent itemsets[1]. Various approaches have been proposed so far to address this problem of preserving sensitive patterns. This paper propose an algorithm that is aimed to sanitize sensitive information in a frequent pattern tree(because trees exhibits the relationships among the itemsets more clearly) which leaves no trace for the counterpart or an adversary to extract the hidden information back, by blocking all possible inferences.

In this article, section-II briefs the literature review; section-III states the definitions needed for the sanitization approach and algorithm presented in this article and section-IV gives the proposed algorithm. In section-V illustration with sample graphs are given and in section-VI the performance metrics are discussed with sample results.

2. LITERATURE REVIEW

Due to wide applicability of the field data mining and in particular for the task of association rules, the focus has been more specific for the problem of protection of sensitive knowledge against inference and it has been addressed by various researchers[2-12]. This task is referred to as sanitization in [2] which blocks inference of sensitive rules that facilitate collaborators to mine independently their own data and then sharing some of the resulting patterns. The above work

concentrate on hiding frequent itemsets in databases based on the support and/or confidence framework. In many situations, it would be more comfortable to share the information in the form of structured patterns like graphs, trees, lattices, etc instead of sharing the entire databases.

In structured patterns when a particular sensitive pattern is to be removed, its supersets and subsets should also be removed in order to block the forward and backward references. The work presented in [7], proposes an algorithm(DSA) that sanitize sensitive information in Graphs and have compared the efficiency with that of Naïve approach. Naïve blocks only the forward inference attack; but DSA blocks both forward and backward inference attacks. But in DSA, the subsets of the sensitive pattern are chosen at random. In this situation, possibility is there for the removal of more number of patterns; because, when a subset pattern is removed, the other patterns associated with it would also be removed failing which would leave forward trace for the counterpart to infer the details of the hidden pattern. Hence, this random removal would reduce the data utility of the source dataset. To overcome this problem, the work proposed in this article presents an algorithm(RSS) for sanitizing sensitive information in structured pattern tree that use a rank function for reducing the computational complexity and legitimate information loss. In comparison with DSA, this algorithm completely blocks the forward and backward inference attacks by removing the sensitive information and its associated information in an optimized way by means of the rank function.

3. BASIC DEFINITIONS

Tree: A Tree is a finite set of one or more nodes such that there is a specially designated node called the root and the remaining nodes are partitioned into $n \geq 0$ disjoint sets $T_1, \ldots T_n$, where each of these nodes is a tree. The sets $T_1,..T_n$ are called the subtrees of the root[13].

Set-Enumeration (SE)-tree: It is a tool for representing and/or enumerating sets in a best-first fashion. The *complete* SE-tree systematically enumerates elements of a power-set using a pre-imposed order on the underlying set of elements.

Structured Pattern Tree: A structured pattern tree denoted by T=(N, L) consists of nonempty set of frequent itemsets N, a set of links L that are ordered pairs of the elements of N such that for all nodes $a, b \in N$ there is a link from a to b if $a \cap b = a$ and $|b| - |a| = 1$, where $|x|$ is the size of the itemset x.

Level: Let T=(N, L) be a structured pattern(frequent itemset) tree. The level k of an itemset x such that $x \in N$, is the length of the path connecting an 1-itemset (usually al level-0) to x.

Height: Let T=(N, L) be a structured pattern tree. The height, h of T is the length of the maximum path connecting an 1-itemset a with any other itemset b, such that $a,b \in N$ and $a \subset b$.

Delete: The *deletion* of a node x from T_i, is denoted as *Del(x)*. The resulting T_i' is the same as T_i without the node x. In particular, if $p_1,..,p_m, x, s_1, ..., s_n$ is the sibling sequence in a level of T_i, then $p_1, ...,p_m, s_1, ..., s_n$ is the sibling sequence in T_i'.

Negative Border Nodes: Negative border nodes possess the property of having all its members (proper subsets) are frequent.

Problem state: Each node in the tree is a problem state.

R-nodes: Nodes that are sensitive and to be restricted before allowing the structured pattern to be shared.

P-nodes: Predecessor node(subsets) of R-nodes which are to be identified (in order to block forward inference) before selecting the particular nodes that are to be deleted.

S-nodes: Successor nodes(supersets) of R-nodes which are to be identified (in order to block backward inference) before selecting the particular nodes that are to be deleted.

Victim states: Problem states for which the path from P-node(s) at level-1(containing 2-itemsets) to *R-node* and/or to S-node(s) are to be searched to select nodes for deletion.

V-node(s): Node(s) to be deleted selectively (based on rank function) among the victim states.

Rank function - r(.): Choose *P-node* (of *R-node*) which leads to only one *S-node* (with single Primary Link); choose one at random when tie occurs. The search for victim nodes can often be speeded up by using the ranking function *r(.)* for all *P-nodes.* The ideal way to assign ranks would be on the basis of minimum additional computational cost needed when this *P-node* is to be removed.

4. ALGORITHM

Rank-based Structured-pattern Sanitization(RSS):

Input: Frequent Pattern Tree(T), Set of Restricted Nodes(R-nodes)

Output: Sanitized Tree(T')

```
Begin
Obtain height h of the input tree;
identify r_i Є R ( R-nodes to be restricted);
//Select victim nodes//
for each r_i Є R
{
  find level k;
  V_nodes[ ] ← r_i ;
  if k > 0
  {
    do while(k<=h)
    {
      obtain S-nodes of r_i (ie supersets);
      V_nodes[ ] ← V_nodes[ ]+S-nodes(r_i);
    }
    do while(k>=1)
    {
      obtain P-nodes of r_i (ie subsets );
      V_nodes[ ] ← V_nodes[ ]+P-node that satisfy r(.);
    }
  }
  delete V_nodes[ ];
}
T'←T;
End
```

5. ILLUSTRATION

A sample frequent pattern SE-tree is given in Fig.1. Let the node to be protected(R-node) is the one with itemset *abc(dark-filled in Fig.2)*; When this is marked for deletion(V-nodes), conventionaly it becomes infrequent. As per antimonotone property of frequent itemsets, *if a set cannot pass a test, all of its supersets will fail the same test as well*. Hence all of its supersets(S-nodes) are to be identified and deleted until the level equals the height of the tree. In this example, node *abcd (shaded)* is the superset of *abc* and so it is marked for deletion(V-nodes). Deletion of R-node and its supersets may completely hide the details of the sensitive data(Restricted nodes) and this ensures the blocking of backward inference of R-node.

Morever, the negative border nodes are also to be deleted to completely block the future inference of the sensitive data. This can be achieved by identifying the Predecessor nodes(P-node) of R-node and suitably removing them by means of *rank function-r(.)* defined earlier. In this example, *abc* has two P-nodes, *ab* and *ac* which are having primarly links and of them as *ac(shaded)* has only one primary link, it is marked for deletion(V-nodes) with all its successors(in this case, *acd*). Refer Fig.2.

Finally delete all victim nodes and thus the sanitized frequent pattern tree to be shared is resulted (the one given Fig.3) and hence forward inference is also blocked.

On the contrary, if *ab* would have been chosen as victim node, then three more nodes would have been additionally removed which would result in more information loss and utility loss. Hence the rank function used in this approach sanitizes the structured frequent pattern tree with reduced information loss and utility loss.

However, the nodes at level-0 (1-itemsets) are not deleted in any way and this preserves the distinct items in the given structured frequent pattern tree.

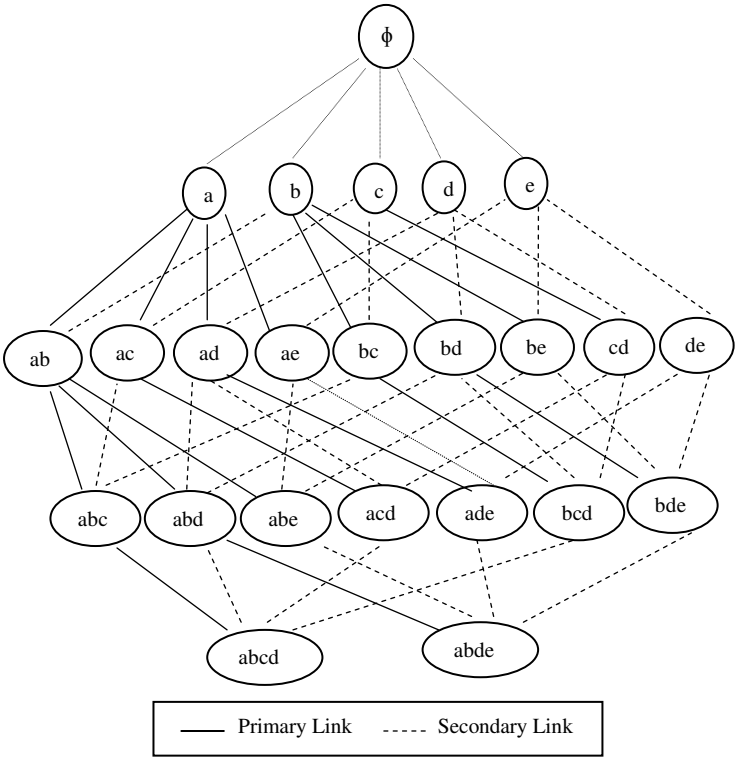

FIGURE 1: Frequent Pattern Tree before Sanitization.

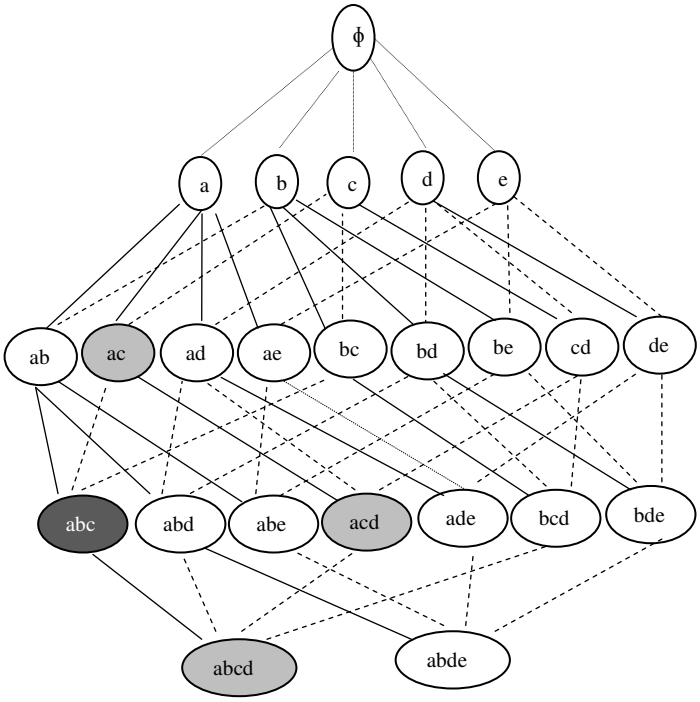

FIGURE 2: Frequent Pattern Tree with Victim Nodes.

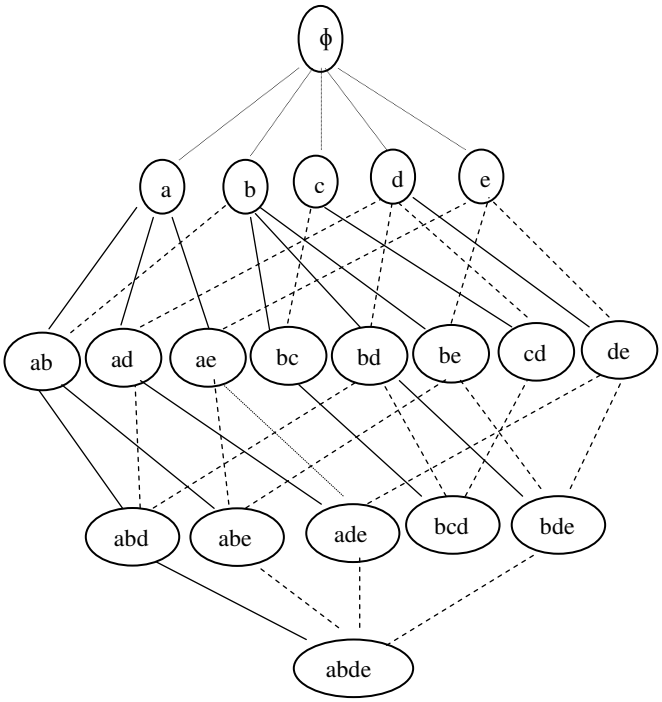

FIGURE 3: Frequent Pattern Tree after Sanitization.

6. EXPERIMENTAL ANALYSIS

The algorithm was tested for real dataset T10I4D100K[14] with number of transactions ranging from 1K to 10K and number of restricted nodes from 1 to 5. The test run was made on Intel core i5 processor with 2.3 GHz speed and 4GB RAM operating on 32 bit OS; The implementation of the proposed algorithm was done with windows 7 - Netbeans 6.9.1 - SQL 2005. The frequent patterns were obtained using Matrix Apriori[15], which requires only two scans of original database and uses simpler data structures.

The efficiency of this approach is studied based on the measures given below and it has been compared (Figures 4 to 7) with the previously proposed algorithms IMA, PMA, TMA[9-12] which sanitizes the sensitive patterns(itemsets) in the source datasets.

Dissimilarity(dif) : The dissimilarity between the original(D) and sanitized(D') databases is measured in terms of their contents which can be measured by the formula,

$$dif(D, D') = \frac{1}{\sum_{i=1}^{n} f_D(i)} \times \sum_{i=1}^{n} [f_D(i) - f_{D'}(i)]$$

where fx(i) represents the i^{th} item in the dataset X. This approach has very low percentage of dissimilarity and this shows that information loss is very low and so the utility is well preserved.

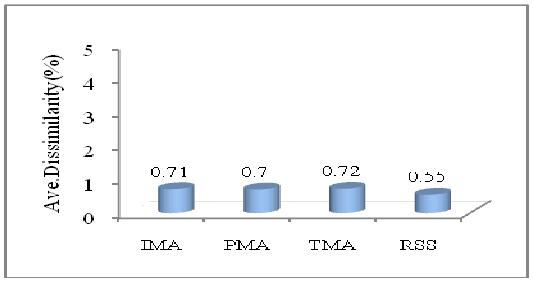

FIGURE 4: Dissimilarity (varying no.of rules).

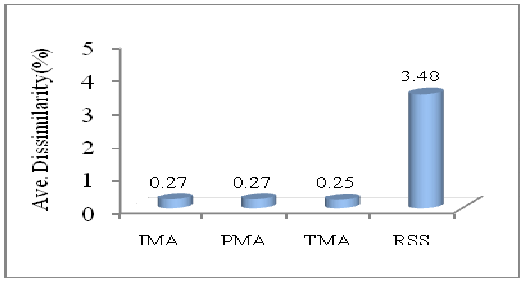

FIGURE 5: Dissimilarity (varying no.of transactions).

From the fig.4 &5, it is observed that the proposed algorithm, RSS has very low dissimilarity in comparison with previous algorithms. However, when the no. of transactions are increased, the dissimilarity gets increased; this is due to the removal of subsets(with its associated nodes) of the sensitive nodes for blocking backward inference attack and it is observed to be less than 5%.

CPU Time: The execution time is tested for the proposed algorithm by varying the number of nodes to be restricted. Fig.6 & 7 shows that the execution time required for RSS algorithm is low in comparison with the other algorithms. It is also observed that execution time is minimum, when the no. of transactions in the source dataset is more. However, time is not a significant criteria as the sanitization is done offline.

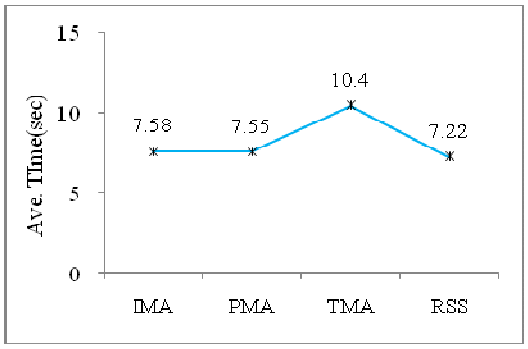

FIGURE 6: Execution Time (varying no.of rules).

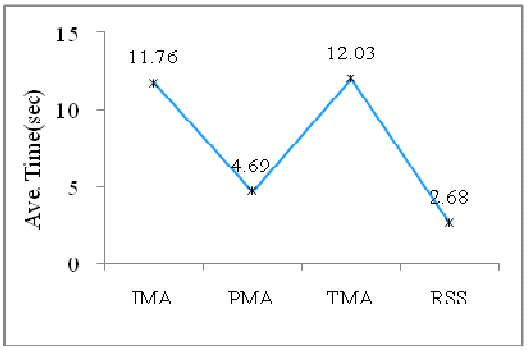

FIGURE 7: Execution Time (varying no.of transactions).

Scalability: In order to effectively hide sensitive knowledge in patterns, the sanitization algorithms must be efficient and scalable which means the running time of any sanitizing algorithm must be predictable and acceptable. The efficiency and scalability of the proposed approach is proved below:

Theorem: The running time of the Rank-based Structured-pattern Sanitization(RSS) approach is at least O[r(l+s)]; where r is the number of restrictive nodes(R-nodes), l is the number of preceding levels that have subsets of R-nodes and s is the number of S-nodes(supersets) of R-nodes.

Proof : Let *T* be a given Structured frequent pattern tree with *N* being the total number of nodes in *T; r be* the number of sensitive nodes(*R-nodes*) to be restricted among *N*; *l* be the number of preceding levels of *R-nodes* and *s* be the number of *S-nodes(supersets)* of *R-nodes* in *T*.

The proposed approach finds the height of the given tree. For every given *R-node*, find the victim states which are the collection of its S-nodes(supersets) and P-nodes(subsets) that lead with only one primary link for their own successors. As this approach satisfies anti-monotone property, all *s* S-nodes are victim nodes and to be deleted to block the backward inferences. However among the P-nodes(subsets), at each preceding level (other than level-0) the node(subset) which forms as a single primary link for its successors is to be obtained and deleted (with all its successors) in order to block all forward inferences; this selection process is quiet straightforward and it gets repeated for all R-nodes.

This algorithm makes use of both depth-wise and breadth-wise search which requires atleast *O(l+s)* computational complexity for every R-node.

Hence, the running time of proposed algorithm for k R-nodes is atleast *O[r(l+s)],* which is *linear* and better than $O(n^2)$, $O(n^3)$, $O(2^n)$, *O(n log N).*

7. CONCLUSION

The proposed algorithm in this work sanitizes the structured frequent pattern tree in an optimal way, by using a rank function that reduces the computational complexity as well as the information loss and utility loss. Moreover, this approach blocks all the inference channels of the restrictive patterns in both forward and backward directions leaving no trace of the nodes that are restricted(removed) before sharing. This simulation process facilitates the task of sanitizing the structured pattern with different set of restricted information when it is to be shared between different set of collaborators. However, when the database is large, it is sometimes unrealistic to construct a main-memory based pattern tree. The proposed algorithm sanitizes patterns in static dataset and also the sanitization is done offline due to the offline decision analysis of the restricted rules. But further effort is being taken to apply optimized heuristic approach to sanitize continuous and dynamic dataset.

8. REFERENCES

[1] J.Han, M.Kamber, *Data Mining Concepts and Techniches, Oxford University Press*, 2009.

[2] M.Atallah, E.Bertino, A.Elmagarmid, M.Ibrahim and V.Verykios “Disclosure Limitation of Sensitive Rules”, Proc. of IEEE Knowledge and Data Engineering Workshop, pages 45–52, Chicago, Illinois, Nov 1999.

[3] E.Dasseni, V.S.Verykios, A.K.Elmagarmid & E.Bertino, “Hiding Association Rules by Using Confidence and Support”, Proc. of the 4th Information Hiding Workshop, pages 369– 383, Pittsburg, PA, Apr 2001.

[4] Y.Saygin, V.S.Verykios, and C.Clifton, “Using Unknowns to Prevent Discovery of Association Rules”, SIGMOD Record, 30(4):45–54, Dec 2001.

[5] S.R.M.Oliveira, and O.R.Zaiane, "Privacy preserving Frequent Itemset Mining", Proc. of the IEEE ICDM Workshop on Privacy, Security, and Data Mining, Pages 43-54, Maebashi City, Japan, Dec 2002.

[6] S.R.M.Oliveira, and O.R.Zaiane, "An Efficient One-Scan Sanitization for Improving the Balance between Privacy and Knowledge Discovery", Technical Report TR 03-15, Jun 2003.

[7] S.R.M.Oliveira, and O.R.Zaiane, "Secure Association Rule Mining", Proc. of the 8th Pacific-Asia Conference on Knowledge Discovery and Data Mining(PAKDD'04), Pages 74-85, Sydney, Australia, May 2004..

[8] B.Yildz, and B.Ergenc, "Hiding Sensitive Predictive Frequent Itemsets", Proc. of the International MultiConference of Engineers and Computer Scientists 2011, Vol-I.

[9] P.Cynthia Selvi, A.R.Mohamed Shanavas, "An effective Heuristic Approach for Hiding Sensitive Patterns in Databases", International Organization of Scientific Research-Journal of Computer Engineering(IOSRJCE) Vol. 5, Issue 1(Sep-Oct 2012), PP 06-11.

[10] P.Cynthia Selvi, A.R.Mohamed Shanavas, "An Improved Item-based Maxcover Algorithm to protect Sensitive Patterns in Large Databases", International Organization of Scientific Research-Journal of Computer Engineering(IOSRJCE) Vol.14, Issue 4, Oct 2013, Pages 1-5.

[11] P.Cynthia Selvi, A.R.Mohamed Shanavas, "Output Privacy Protection With Pattern-Based Heuristic Algorithm", International Journal of Computer Science & Information Technology(IJCSIT) Vol 6, No 2, Apr 2014, Pages 141 – 152.

[12] P.Cynthia Selvi, A.R.Mohamed Shanavas, "Towards Information Privacy Using Transaction-Based Maxcover Algorithm", World Applied Sciences Journal 29 (Data Mining and Soft Computing Techniques): 06-11, 2014.

[13] Ellis Horowitz, Sartaj Sahni, Sanguthevar Rajasekaran, *Fundamentals of Computer Algorithms,* Galgotia Pub. Pvt. Ltd, Delhi, 1999.

[14] The Dataset used in this work for experimental analysis was generated using the generator from IBM Almaden Quest research group and is publicly available from http://fimi.ua.ac.be/data/.

[15] J.Pavon, S.Viana, S.Gomez, "Matrix Apriori: speeding up the search for frequent patterns", Proc. 24th IASTED International Conference on Databases and Applications 2006, pp. 75-82.