A Novel Algorithm for Completely Hiding Sensitive Association Rules

Chih-Chia Weng, Shan-Tai Chen, Hung-Che Lo
Dept. of Computer Science, Chung Cheng Institute of Technology, National Defense University, Taiwan, R.O.C.
{g971204, stchen, c2981223}@ndu.edu.tw

Abstract

With rapid advance of the network and data mining techniques, the protection of the confidentiality of sensitive information in a database becomes a critical issue when releasing data to outside parties. Association analysis is a powerful and popular tool for discovering relationships hidden in large data sets. The relationships can be represented in a form of frequent itemsets or association rules. One rule is categorized as sensitive if its disclosure risk is above some given threshold. Privacy-preserving data mining is an important issue which can be applied to various domains, such as Web commerce, crime reconnoitering, health care, and customer's consumption analysis.

The main approach to hide sensitive association rules is to reduce the support or the confidence of the rules. This is done by modifying transactions or items in the database. However, the modifications will generate side effects, i.e., nonsensitive rule falsely hidden (i.e., lost rules) and spurious rules falsely generated (i.e., new rules). There is a trade-off between sensitive rules hidden and side effects generated.

In this study, we propose an efficient algorithm, FHSAR, for fast hiding sensitive association rules (SAR). The algorithm can completely hide any given SAR by scanning database only once, which significantly reduces the execution time. Experimental results show that FHSAR outperforms previous works in terms of execution time required and side effects generated in most cases.

Key Words: association rules, privacy preserving data mining, sensitive association rules, side effects.

1. Introduction

The data mining technologies have been an important technology for discovering previously unknown and potentially useful information from large data sets or databases. They can be applied to various domains, such as Web commerce, crime reconnoitering, health care, and customer's consumption analysis. However, the technologies can be threats to data privacy. Association rule analysis is a powerful and popular tool for discovering relationships hidden in large data sets. Some private information could be easily discovered by this kind of tools. Therefore, the protection of the confidentiality of sensitive information in a database becomes a critical issue to be resolved.

The problem for finding an optimal sanitization to a database against association rule analysis has been proven to be NP-Hard [1]. The research can be divided into hiding sensitive rules [2-5] and sensitive items [6-8]. Vassilios S. Verykios et al. [2] conducted a thorough investigation and presented five algorithms for hiding sensitive association rules. They concluded that among the proposed algorithms there is not a best solution for all the metric, including 1) the execution time required and 2) the side effects produced by the proposed algorithms. Later on, mush research has been done and focused on some issues. Shyue-Liang Wang [6] proposed algorithms to hide sensitive items instead of hiding sensitive association rules. The algorithm needs less number of database scans but the side effects generated are also high. Ali Amiri [7] presented heuristic algorithms to hide sensitive items, while maximizing data utility at the expense of computational efficiency. Finally, Yi-Hung Wu et al. [3] proposed a heuristic method that could hide sensitive association rules with limited side effects. However, it also spent a lot of time on comparing and checking if the sensitive rules are hidden and if side effects are produced. Besides, it fails to hide some sensitive rules in some cases.

In this paper, we propose a novel algorithm, FHSAR, for hiding sensitive association rules (SAR). The algorithm can completely hide any given SAR by scanning database only once. Experimental results

978-0-7695-3382-7/08 $25.00 © 2008 IEEE
DOI 10.1109/ISDA.2008.180

show that FHSAR outperforms previous works in terms of execution time and side effects generated.

This paper is organized as follows: Section 2 introduces the problem and notations. In Section 3, we describe the main concept of the proposed algorithm and give an illustrative example. Section 4 presents the experimental results, which compare the performance of the proposed algorithm to that of previous works. Section 5 is the conclusion.

2. Problem Formulation and Notations

In Table 1, we summarize the notations used hereafter in this paper. The support of itemset S can be computed by the following equation:

$$support(S) = ||S|| / |D|, \quad (1)$$

where ||S|| denotes the number of transactions in the database that contains the itemset S, and |D| denotes the number of the transactions in the database D. We call S as a *frequent itemset* if support(S) ≥ *min_support*, a given threshold. A transaction t_i *supports* S, if $S \subseteq t_i$.

An *association rule* is an implication of the form X→Y, where X⊂I, Y⊂I and X∩Y= Ø. A rule X→Y is *strong* if

1) support(X→Y) ≥ *min_support* and

2) confidence(X→Y) ≥ *min_confidence*,

where *min_support* and *min_confidence* are two given minimum thresholds, and the support(X→Y) and confidence(X→Y) can be computed by the following equations:

$$support(X \rightarrow Y) = ||X \cup Y|| / |D|; \quad (2)$$

$$confidence(X \rightarrow Y) = ||X \cup Y|| / |X|. \quad (3)$$

Example 1. An example database is shown in Table 2. There are nine items, |I|=9, and five transactions, |D|=5, in the database. Table 3 shows the frequent itemsets generated from Table 2 for *min_support* = 60%. For the example S = {1,4,7}, since $S \subseteq t_1$, $S \subseteq t_2$ and $S \subseteq t_3$, we obtain ||S||=3. Therefore, support(1,4,7) = ||S|| / |D| = 60%. Table 4 shows the association rules generated from Table 2 for *min_support* = 60% and *min_confidence* = 75%. For the example rule 1,4→7, since ||{1,4}|| = 3 and ||{1,4,7}||=3, with the equations (2) and (3), we can get support(1,4→7) = 60% and confidence(1,4→7) = 100%. ■

Our study goal is to completely hide all SAR while minimizing the side effects generated from the database modification. Figure 1 shows the relationships among the sets, U, U', and SAR. That is, the goal is to gurantee U'∩SAR = Ø and to minizize the two sets, U–SAR–U' and U'–U.

Table 1. Notations and Definitions

Notation	Definition
I	$I = \{i_1, i_2, ..., i_m\}$ a set of items in a transaction database
D	the original database $D = \{t_1, t_2, \ldots, t_n\}$, where every transaction t_i is a subset of I, i.e., $t_i \subseteq I$.
D'	the released database which is transformed from D
U	the sets of association rules generated from D
U'	the sets of association rules generated from D'
$\|\cdot\|$	the number of elements in a set $\cdot$
SAR	the set of sensitive association rules to be hidden, $SAR=\{SAR_1, SAR_2, \ldots, SAR_m\}$
SAR'	the set of sensitive association rules has been hidden
$L(\cdot)$	an itemset on the left hand side of a rule
$R(\cdot)$	an itemset on the right hand side of a rule
$\|\| \cdot \|\|$	the support count of an itemset, i.e., the number of transactions in the database that contain the itemset
PWT	a table for storing ID and weight, w_i, for each transaction in an order decreasing by w_i
$t_i.k$	an item in transaction t_i

Table 2. Database D

ID	transaction
1	1,2,4,5,7
2	1,4,5,7
3	1,4,6,7,8
4	1,2,5,9
5	6,7,8

Table 3. Frequent itemsets generated from Table 2, *min_support* = 60%

Itemset	Support
1	80%
4	60%
5	60%
7	80%
1,4	60%
1,5	60%
1,7	60%
1,4,7	60%
4,7	60%

Table 4. Association rules generated from Table 2, *min_support*=60% and *min_confidence*=75%

rules	support	confidence	rules	support	confidence
1→ 4	60%	75%	7→ 4	60%	75%
4→ 1	60%	100%	1,4→ 7	60%	100%
1→ 5	60%	75%	1,7→ 4	60%	100%
5→ 1	60%	100%	4,7→ 1	60%	100%
1→ 7	60%	75%	1→ 4,7	60%	75%
7→ 1	60%	100%	4→ 1,7	60%	100%
4→ 7	60%	100%	7→ 1,4	60%	75%

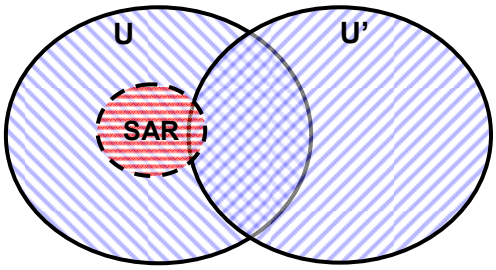

Figure 1. The relationships among the sets, U, U', and SAR

3. The Proposed Algorithms

We now demonstrate the algorithm, FHSAR. Given D, SAR, *min_support*, and *min_confidence*, The goal of FHSAR is to generate a database to be released, D', in which the sensitive association rules are completely hidden and the side effects generated are minimized. The sketch of the FHSAR algorithm is shown in Figure 2, which can be depicted as the following stages.

In stage 1, FHSAR scans database once while collects information about the correlation between each transaction and sensitive rules. As shown in Figure 3, the correlation between a transaction, e.g., t_1, and the SAR can be represented by a graph G. Each node is for an item i_k in t_1, which can be represented as $\langle R_k, |R_k| \rangle$, where $R_k = \{ j \mid SAR_j \subseteq t_i, i_k \in SAR_j \}$. The w_i is a prior weight of a transaction t_i, which provides a heuristic for estimating side effects and can be computed by the formula.

$$w_i = MIC_i / 2^{(|t_i| - 1)}, \quad (4)$$

where $MIC_i = \max(|R_k|)$.

The weight associated with each edge (u, v) denotes the number of itemsets in SAR that contain the both items, i.e., i_u and i_v, connected by the edge. Example 2 illustrates the way to calculate the prior weight for each transaction.

```
   FHSAR( );
   Input: D, SAR, min_support, min_confidence;
   Output: D', where all SAR will be hidden.
   Stage 1
01 For each transaction t_i in the database D Do
02 { For each sensitive rule SAR_j ∈ SAR Do
03   { If SAR_j supported by t_i Then
04     { ||SAR_j|| = ||SAR_j|| + 1;
05       ||L(SAR_j)|| = ||L(SAR_j)|| + 1;
06     }
07     Else IF L(SAR_j) suppoted by t_i Then
08       ||L(SAR_j)|| = ||L(SAR_j)|| + 1;
09   }
10   If exist any SAR_j supported by t_i
11   { MIC_i = Item_Selection ( );
12     w_i = MIC_i / 2^(|t_i| - 1);
13     Store the t_i's ID and the w_i in PWT;
14   }
15 }

     Stage 2
16 While SAR is not empty (≠∅) do
17 { Select t_ID from PWT with maximal weight w;
18   t_ID.k = Item_Selection ( );
19   IF Checking_and_Removing Item( ) == True Then
20   { Modify w_ID of the t_ID and insert the ID into the PWT
         in the maintained order;
21     For each SAR_j that t_ID.k∈SAR_j Do
22     { IF SAR_j⊆t_ID Then
23         ||SAR_j|| = ||SAR_j|| – 1;
24       IF (L(SAR_j)⊆t_ID) and ((t_ID.k)∈L(SAR_j) ) Then
25         ||L(SAR_j)|| = ||L(SAR_j)|| – 1;
26       IF (support(SAR_j) < min_support) or
            (confidence(SAR_j) < min_confidence) Then
27         Remove SAR_j from SAR;
28     }
29   }
30 }
```

Figure 2. The pseudo code of the FHSAR algorithm

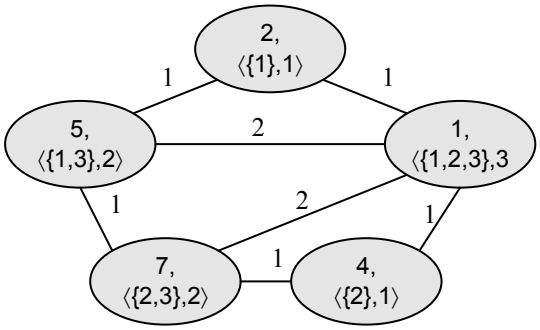

Figure 3. The correlation between t_1 and SAR

Example 2. Table 5 shows an example of sensitive association rules to be hidden. Let t_1 = {1, 2, 4, 5, 7},

which supports SAR_1, SAR_2 and SAR_3. The correlation between t_1 and SAR is shown in Figure 3. The node ⟨ {1, 3}, 2 ⟩ for item '5' indicates that t_1 supports two sensitive rules, SAR_1 and SAR_3, that contain the item '5', i.e., R_k = {1,3} and $|R_k|$=2. As shown in Figure 3, max($|R_k|$) = max(3,1,1,2,2) = 3. Hence, we obtain MIC_1 = 3 and w_1 = 3/16. ■

Stage 2 repeats to modify transactions one-by-one until all rules in SAR have been hidden. The order of the modifications is according to the prior weight associated with each transaction. The following tasks are repeated until SAR is empty.

- Select a transaction t_k from PWT such that w_k is maximal.
- Select an item to be deleted, according to the heuristic shown in Figure 4.
- The function, checking-and-removing Item, shown in Figure 5 is for avoiding hidden failures, That is, given any set of SAR, FHSAR is able to completely hide all sensitive association rules in SAR; i.e., U'∩SAR = Ø.
- Compute w_k again after each item modified, and then insert t_k into the PWT in the maintained order.
- Modify $||SAR_j||$ and $||L(SAR_j)||$.
- Remove SAR_j from SAR, if SAR_j is not still a strong rule.

Now, we use the following example for illustrating the proposed algorithm FHSAR.

Example 3. Given D (in Tables 2), SAR (in Tables 5), *min_support* = 40% and *min_confidence* = 60%, FHSAR performs the following tasks. First, scan the D and collect information. As shown in Table 6, the $||SAR_j||$ and $||L(SAR_j)||$ for each SAR_j can be obtained from D and SAR. For example, SAR_3, 1,5→7, is supported by t_1 and t_2, so $||SAR_3||$ = 2, and $L(SAR_3)$, {1,5}, is supported by t_1, t_2, and t_4, so $||L(SAR_3)||$ = 3. Table 7 lists the length, MIC, and the prior weight for each transaction in the database. The PWT, as shown in Table 8, can be obtained by sorting Table 7 in the decreasing order by *w*. Then, the first transaction, i.e., t_2, in PWT is chosen to be modified. According the heuristic shown in Figure 4, the item '1' or '7' (selected randomly) in t_2 is removed. If item '1' is removed, then $||SAR_2||$, $||SAR_3||$, $||L(SAR_2)||$, and $||L(SAR_3)||$ will be decreased by 1. SAR_3 will be removed from SAR because the ($||SAR_3||$ / |D|) < *min_support*. The process is repeated until the SAR is empty. Finally, the FHSAR algorithm removes the item '1' in t_2, the item '6' or '8' (selected randomly) in t_5, and the item '1' in t_1. Now all sensitive association rules in SAR have been hidden. ■

Table 5. An example of sensitive association rules

j	SAR (X→Y)	X∪Y	X
1	1,2→5	1,2,5	1,2
2	1,4→7	1,4,7	1,4
3	1,5→7	1,5,7	1,5
4	6→8	6,8	6

Table 6. The support and confidence for each rule in SAR

j	SAR_j	$\|\|SAR_j\|\|$	$\|\|L(SAR_j)\|\|$	support	confidence
1	1,2→5	2	2	40%	100%
2	1,4→7	3	3	60%	100%
3	1,5→7	2	3	60%	66.67%
4	6→8	2	2	40%	100%

Table 7. The MIC and prior weight for each transaction in D

ID	Transaction	$\|t_i\|$	MIC	w_i
1	1,2,4,5,7	5	3	3/16
2	1,4,5,7	4	2	2/8
3	1,4,6,7,8	5	1	1/16
4	1,2,5,9	4	1	1/8
5	6,7,8	3	1	1/4

Table 8. The example PWT

order	ID	w_i
1	2	2/8
2	5	1/4
3	1	3/16
4	4	1/8
5	3	1/16

```
Item_Selection ( );
Input: transaction t_i, SAR;
Output: the item t_i.k to be deleted, MIC_i;
01 {
02   For each SAR_j in SAR do
03   { IF SAR_j ⊆ t_i then
04     { For each item k in SAR_j Do
05         |R_k| = |R_k| + 1;
06     }
07   }
08   MIC_i = max(|R_k|);
09   t_i.k = the item with maximum |R_k|;
10   Return (t_i.k, MIC_i);
11 }
```

Figure 4. The pseudo code of the heuristic for item selection.

```
Checking_and_Removing Item( );
Input: t_TID, t_TID.k, SAR_j', min_support;
// Return True if an item has been deleted and no hidden
// failure occurred, else return False.
01 {
02    IF (support(SAR_j') ≥ min_support) and ( t_TID.k ∈
      L(SAR_j') and SAR_j' ⊄ t_TID) Then
03    { IF (any other item t_TID.p ∉ L(SAR_j') and |R_p≠0|)
        Then
04      { Delete the item t_TID.p;
05        Return True;
06      }
07      ELSE
08      { Skip the t_TID;
09        Return False;
10      }
11    }
12    ELSE
13    { Delete the item t_TID.k;
14      Return True;
15    }
```

Figure 5. The Hidden_Failure_Avoided_Procedure pseudo code

4. Performance Evaluation

We have performed two series of experiments on a PC with Pentium III 450 MHz CPU and 256 MB memory, under the Windows 98 operating system. The first series of experiments estimates performance of the FHSAR according to three criteria: CPU time requirements, the side effects generated, and the number of transactions modified. The second series of experiments compares our results with previous works [2].

The IBM data generator [9] is used to synthesize the databases and the Apriori package [10] to generate rules for the experiments. The parameter settings are as follows:

- Databases sizes: 10K, 20K, 30K, 40K, 50K, and 100K.
- The average length of transactions: 5
- | I | = 50, *min_support* = 2%, *min_confidence* = 15%

Tables 9 and 10 present the experimental results of FHSAR for |SAR|=5 and |SAR|=10, respectively. The CPU time requirements, new rules generated, lost rules produced, and the number of entries modified for varied |D| and |SAR| are shown in Figures 6, 7, 8, and 9, respectively.

Table 9. Experiment results of FHSAR for |SAR|=5

|D|	CPU time(ms)	|U|	|U'|	#new rules	#lost rules	#modified entries
10K	160	502	491	2	8	154
20K	330	494	484	0	5	300
30K	550	496	486	2	8	449
40K	820	485	475	0	5	640
50K	1040	488	475	0	8	725
100K	3900	483	474	0	4	1414

Table 10. Experiment results of FHSAR for |SAR|=10

|D|	CPU time(ms)	|U|	|U'|	#new rules	#lost rules	#modified entries
10K	390	502	473	0	19	589
20K	880	494	465	1	20	843
30K	1420	496	467	1	20	1718
40K	2420	485	455	2	22	2389
50K	3180	488	458	1	21	2889
100K	10420	483	452	0	21	5801

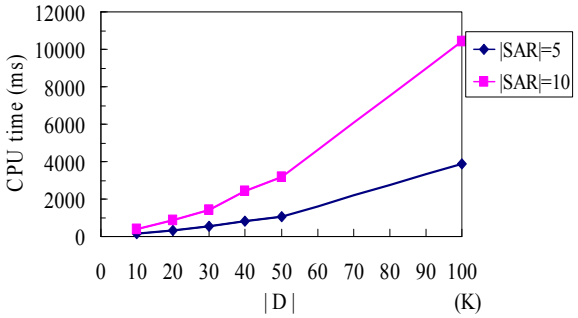

Figure 6. The CPU time requirements.

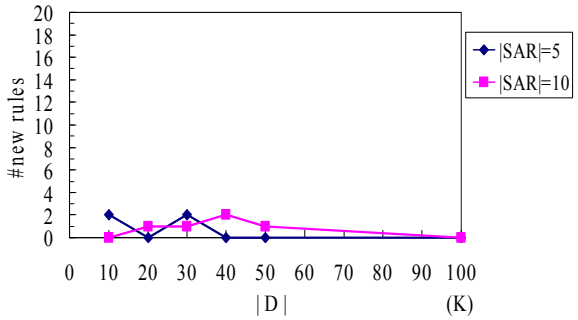

Figure 7. The new rules generated for hiding 5 and 10 rules

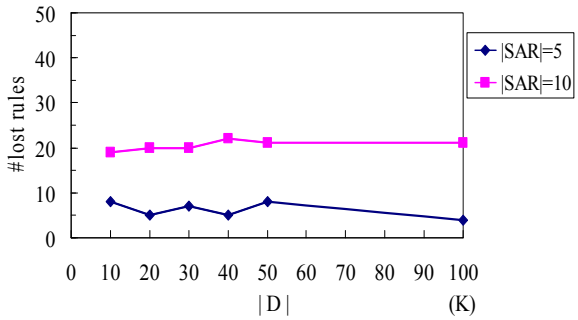

Figure 8. The lost rules produced for hiding 5 and 10 rules

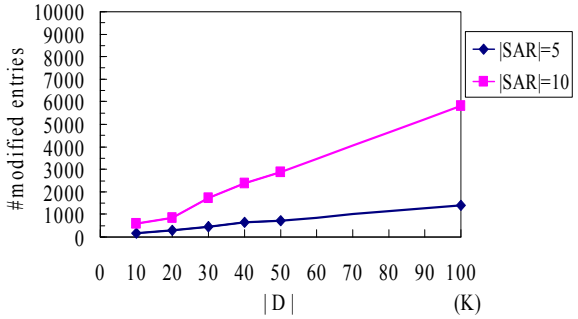

Figure 9. The number of entries modified for hiding 5 and 10 rules

The experimental results of FHSAR can be summarized as follows:

- As shown in Figure 6, the CPU time required for all varied |D| and |SAR| is within 11 seconds. The efficiency is due to only one database scan required in FHSAR.
- The number of new rules is minimized and independent of the size of database, which can be discovered in Figure 7.
- The number of lost rules is independent of the size of database, which can be discovered in Figure 8.
- The number of the modified entries depends on the size of the database and the size of SAR. However, since heuristic procedures are used to determine the order of modifications, we can observe in Figure 9 that only a small part of transactions in the database are modified. For the example of |D|=50000, only 3000 transactions are modified for completely hiding the 10 rules in SAR.

The second series of experiments presents the comparison results to the previous works [2], including CPU time, new rules and lost rules. We show the comparison results in Figures 10 to 15 respectively.

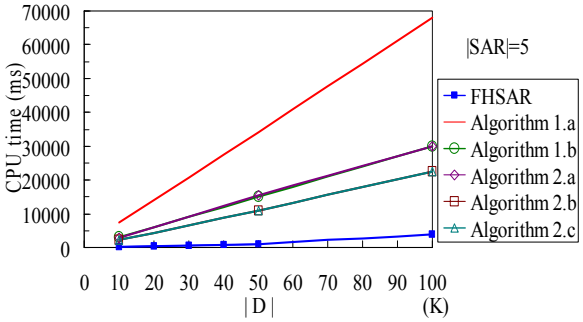

Figure 10. The CPU time for hiding 5 rules

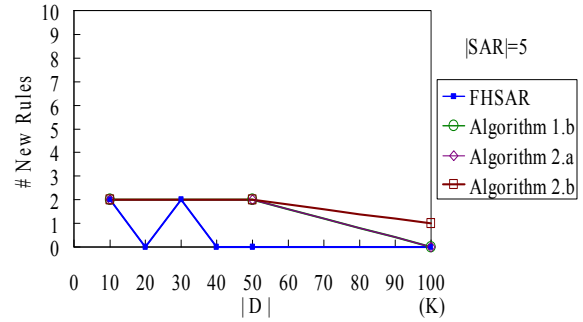

Figure 11. The new rules generated for hiding 5 rules

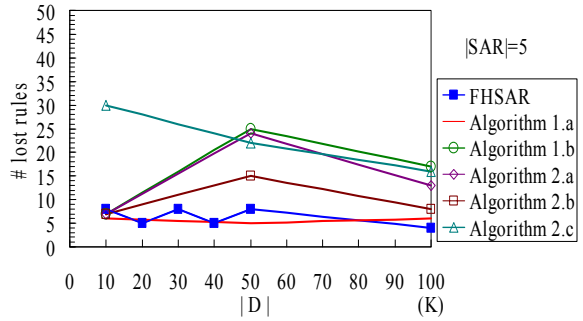

Figure 12. Comparison results of lost rules when |SAR|=5

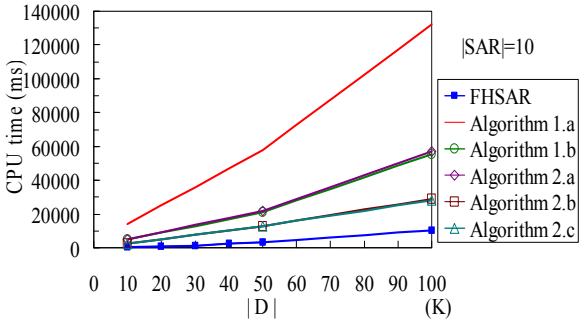

Figure 13. Comparison results of CPU time when |SAR|=10

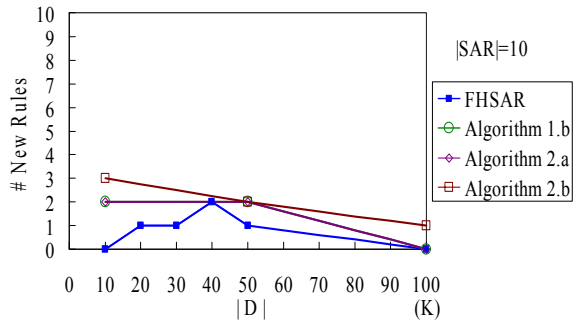

Figure 14. Comparison results of new rules generated when |SAR|=10

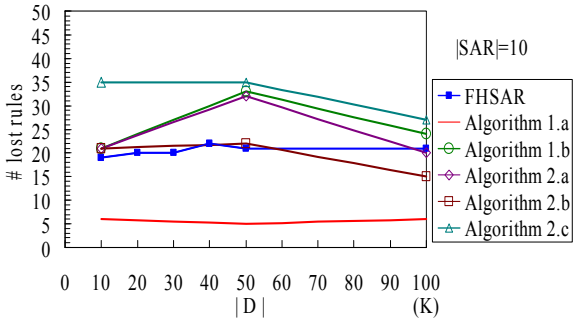

Figure 15. Comparison results of lost rules produced when |SAR|=10

The experimental results can be summarized as follows:

- The results of time requirements are shown in Figures 10 and 13. We can see that the FHSAR is more efficient than others.
- As shown in Figures 11 and 14, the FHSAR generated less new rules than previous works.
- In Figures 12 and 15, Although Algorithm 1.a consistently outperforms others, it will generate lots of new rules. The number of new rules generated by Algorithm 1.a is more than 2000, which is outside the range of Figure 14. So, we omit it there.

5. Conclusions

In this paper, we have presented the FHSAR algorithm for completely hiding sensitive association rules with limited side effects. In FHSAR, a strategy is developed for avoiding hidden failures. In addition, we propose two heuristic approaches for improving the performance for solving the problem. First, a heuristic function is used to obtain a prior weight for each transaction, by which the order of transactions modified can be efficiently decided. Second, the correlations between the sensitive association rules and each transaction in the original database are analyzed, which can effectively select the proper item to modify. By using these strategies, FHSAR is able to scan database only once and efficiently to deal with the NP-hard problem. The proposed algorithm not only limits the side effects generated, but also guarantees to completely hide all given sensitive rules. Experimental results show that FHSAR outperforms previous works in terms of execution time required and side effects generated in most cases. How to efficiently sanitize sensitive information when the database is updated could be studied in the future. We hope the proposed algorithm could promote the motivation for sharing data among a group of organizations, such as the ISAC[11] project.

Acknowledgements

This research was supported in part by the National Science Council of ROC under grants NSC 96-3114-P-606-002-Y and NSC 96-3114-P- 606-001-Y.

References

[1] M. Atallah, E. Bertino, A. Elmagarmid, M. Ibrahim, and V. Verykios, "Disclosure limitation of sensitive rules" *Knowledge and Data Engineering Exchange*, pp. 45-52, 1999.

[2] Vassilios S. Verykios, A.K. Elmagarmid, E. Bertino, Y. Saygin, and E. Dasseni, "Association Rule Hiding," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, no. 4, pp. 434-447, 2004.

[3] Yi-Hung Wu, Chia-Ming Chiang, and Arbee L.P. Chen, "Hiding Sensitive Association Rules with Limited Side Effects", *IEEE Transactions on Knowledge and Data Engineering*, vol. 19, issue 1, pp. 29 - 42, 2007.

[4] Jiuyong Li, Hong Shen, Rodney Topor, "Mining the Smallest Association Rule Set for Predictions", *Proceedings of the 2001 IEEE International Conference on Data Mining*, pp.361-368, 2001.

[5] Shyue-Liang Wang, Kuan-Wei Huang, Tien-Chin Wang, and Tzung-Pei Hong, "Maintenance of discovered informative rule sets: incremental deletion", *International Conference on Systems, Man and Cybernetics*, pp.170-175, 2005.

[6] Shyue-Liang Wang, "Hiding sensitive predictive association rules", *Systems, Man and Cybernetics, 2005 IEEE International Conference on Information Reuse and Integration,* vol. 1, pp. 164-169, 2005.

[7] Ali Amiri, "Dare to share: Protecting sensitive knowledge with data sanitization", *Decision Support Systems archive* vol. 43, issue 1, pp. 181-191, 2007.

[8] Shyue-Liang Wang, Bhavesh Parikh, and Ayat Jafari, "Hiding informative association rule sets", *Expert Systems with Applications*, pp.316-323, 2007.

[9] http://www.almaden.ibm.com/cs/projects/iis/hdb/Projects/data_mining/mining.shtml, May, 2007.

[10] http://www.borgelt.net/apriori.html, May, 2007

[11] http://www.msisac.org/