Ant Colony System Sanitization Approach to Hiding Sensitive Itemsets

Jimmy Ming-Tai Wu[1], Justin Zhan[1] and Jerry Chun-Wei Lin[2*], *Member, IEEE*
[1]Department of Computer Science, University of Nevada, Las Vegas, USA
[2]School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China

In recent years, privacy-preserving data mining (PPDM) has received a lot of attention in the field of data mining research. While some sensitive information in databases cannot be revealed, PPDM can discover additional important knowledge and still hide critical information. There are different ways to approach this exhibited in previous research, which applied addition and deletion operations to adjust an original database in order to hide sensitive information. However, it is an NP-hard problem to find an appropriate set of transactions/itemsets for hiding sensitive information. In the past, evolutionary algorithms (EAs) were developed to hide sensitive itemsets by building an appropriate database. Genetic-based algorithms (GAs) and a particle swarm optimization (PSO)-based algorithm, proposed in previous works, not only hide sensitive itemsets but minimize the side effects of sanitization processes. In this paper, an ant colony system (ACS)-based algorithm called ACS2DT is proposed to decrease side effects and enhance the performance of the sanitization process. Each ant in the population will build a tour for each iteration, and each tour indicates the deleted transactions in the original database. The proposed algorithm introduces a useful heuristic function to conduct each ant to select a suitable edge (transaction) for the current situation and also designs several termination conditions to stop the sanitization processes. The proposed heuristic function applies the pre-large concept to monitor side effects and calculates the degree of hiding information to adjust the selecting policy for deleted transactions. The experimental results show that the proposed ACS2DT algorithm performs better than the Greedy algorithm and other two evolutionary algorithms in terms of runtime, fail to be hidden (F-T-H), not to be hidden (N-T-H), not to be generated (N-T-G) and database similarity (DS) on both real-world and synthetic datasets.

***Index Terms*—privacy-preserving data mining, evolutionary algorithm, sensitive itemsets, ant colony system.**

I. INTRODUCTION

Knowledge Discovery in Database (KDD) is a technique to discover useful information and interesting relationships in an undigested database. It has become more and more important and causes a variety of issues due to the rapid growth of information technology and e-commerce applications. Several techniques in KDD are commonly used to find obscure but useful relationships among purchased products for market basket analysis. For instance, KDD techniques can be generally classified as association rules [5], [10], [21], sequential patterns [6], [37], [46], [18], clusters [27], and classifications [41]. Association rule mining is a principle KDD that will provide valuable information and knowledge in customer transactions.

In recent years, a huge amount of data has been stored in tremendous databases, which contain private or secure information such as credit card numbers, personal identification numbers, telephone numbers and other confidential data, which are easily revealed by previous data mining techniques. Therefore, developing novel techniques from traditional approaches to hide critical information is an important topic. Moreover, business collaborators (which were also potential competitors) could obtain sensitive knowledge by using data mining techniques to analyze shared databases. A current collaborator might become a competitor and cause immeasurable damage by utilizing sensitive knowledge to make more profitable business decisions, and thus decrease the business performance of the data provider as a result of increased competition.

Privacy-preserving data mining (PPDM) is a data mining technique for hiding the private and critical information in a dataset. PPDM was first introduced by Agrawal and Srikant [4]. Lindell and Pinkas applied decision tree learning with the ID3 algorithm to the issue of PPDM [36]. Clifton et al. then presented a toolkit to address various problems in PPDM [11]. Pandya et al. proposed a multiplicative perturbation algorithm to achieve a better balance between privacy and utility of data [40]. Dwork and Nissim generalized previous research by considering both multi-attribute databases and vertically partitioned databases, and proposed several algorithms for handling published noisy statistics [16]. Several approaches were also proposed to hide sensitive frequent itemsets of sensitive association rules by custom sanitization procedures [17], [24], [31], [47].

When sensitive information overlaps with important but non-sensitive information, minimizing side effects for producing an appropriate set of transactions/itemsets is a harsh challenge for traditional PPDM algorithms. Some important information might be hidden or modified, or some inauthentic knowledge might be produced, due to hiding and securing sensitive information at the same time. It is thus difficult to find the optimal solution in a short time by a traditional algorithm in PPDM. An evolutionary algorithm is a suitable and efficient approach for searching near optimal solutions to NP-hard problems. The evolutionary algorithms such as Genetic Algorithm (GA), Particle Swarm Optimization (PSO) or Ant Colony Algorithm (ACO) can be used to obtain beneficial solutions quickly. Thus, for the issue of PPDM,

Manuscript received October 12, 2016; revised April 03, 2017. Corresponding author: Jerry Chun-Wei Lin (email: jerrylin@ieee.org, website: http://ikelab.net).

it is more reasonable to design or apply the evolutionary algorithms to obtain the global optimal solution than to utilize the traditional methods.

Genetic algorithm (GA) [19], [28] is a famous evolutionary algorithm. It simulates the evolutionary process of creatures in a natural environment. A population approaches a rather promising individual by several evolutionary operators, including a crossover operator, mutation operator, and selection policy. It has been frequently applied to several discrete and continuous problems which might include nonlinear objectives or constant functions without gradient information.

Particle swarm optimization (PSO), invented in 1995, is another effective evolutionary algorithm [29]. It is inspired by birds flocking to deduce rich find sources. Like GA, it is also designed to solve optimization problems. Each particle in PSO represents a feasible solution which would be evaluated by a predefined fitness function. A particle would change location by its speed and also update its speed by its acceleration in each iteration. The acceleration is calculated by two factors, *pbest* and *gbest*. The personal best (*pbest*) records the best location which the defined particle arrives at before and the global best (*gbest*) records the best location which the whole population finds from then till now.

Evolutionary algorithms were proposed to find a good enough solution in a limited time for PPDM. In the past, Lin et al. proposed a GA-based algorithm [32], [33] and a PSO-based algorithm [34] to hide sensitive itemsets using a designed sanitization procedure. In these approaches, selecting a set of transactions for deletion is accomplished using a GA framework and a PSO framework. It has been shown that the evolutionary algorithms can provide better solutions to PPDM problems with fewer side effects compared to traditional Greedy algorithms. Previous works using ant-based algorithms hide sensitive information in datasets. Narmadha proposed an ACO approach to modify the transactions in a database in order to hide private data [38]. Selvan and Veni proposed another ant-based algorithm to scan the items in a transaction one by one to try to reduce the side effects during the hiding process [42]. Ansari et al. applied fuzzy concept into an ACO method to propose a data modification technique in PPDM [8]. In this paper, an ACS-based ACS2DT algorithm is proposed to enhance the performance and further decrease the side effects of previous algorithms. The proposed method does not modify transactions in a database but deletes some critical transactions to hide sensitive information. The key contributions of the designed algorithm are listed below.

1) In the past, few evolutionary algorithms have been proposed to sanitize databases for hiding sensitive information. The previous approaches utilized the GA framework and PSO framework. This is the first article to address the problem of hiding sensitive itemsets by deleting some transactions in a database using an ACS-based algorithm.
2) Due to the characteristics of ACS, ACS2DT would select the transactions for deletion one-by-one. It always refers to previous selections and monitors the current situation to determine the better choice with a well-defined heuristic function. The heuristic function utilizes both a pre-large concept and fitness function to reduce the side effects for a designed sanitization procedure.
3) According to the specific selection process for ACS, the proposed algorithm could record the best solution during the selection process. The results of the proposed method reduce the number of unnecessary deleted transactions, and the similarity of the projected dataset to the original dataset is better than previous approaches.

The rest of this paper is organized as follows. Related work is briefly reviewed in Section II. Preliminaries and problem definition are outlined in Section III. The proposed ACS2DT sanitization algorithm is presented in Section IV. Experimental results are reported in Section VI. Finally, conclusions are given in Section VII.

II. Related Work

This section reviews related work about GA, PSO, ACS and PPDM.

A. Genetic Algorithm

Genetic Algorithm (GA) was developed in the early 1970s by John Holland [28], and is the most popular evolutionary algorithm. It is widely used to find near-optimal solutions to optimization problems. It has been confirmed in previous works that GA can obtain good quality near-optimal solutions to several different kinds of NP-hard problems. In GA, a feasible solution would be encoded on a chromosome which is an individual in the population. Each chromosome would be estimated by a predefined fitness function. Three evolutionary operators, a crossover operator, a mutation operator and a selection policy are performed to generate the next population in order to find better solutions in each iteration.

1) **Crossover:** It is an interaction between two selected individuals in a population. In general, this operator would generate two or more offspring for selected individuals. Offspring always consist of some features or characteristics from their parents. For instance, if a chromosome is encoded as a binary string, the mutation operator could be defined to swap some bits between two chromosomes (individuals) to generate offspring.
2) **Mutation:** It is an operator which performs adjustment to generate a similar unit from a certain chromosome. This operation has the ability to let a population escape some local optimal solutions. For instance, mutation can randomly flip several bits of a certain chromosome with a low probability if the chromosome is a binary string.
3) **Selection:** This operator simulates natural selection of the fittest to choose some individuals from the current population to move on to the next iteration. It ensures that the characteristics or the features of the better solutions would be retained in the population in the future iterations.

The brief procedure of GA is given below. In general, a predefined maximum generation G would be set as a termination condition. When the number of generations reaches G, the GA process would be interrupted and would output the current

global optimal solution *S*. Each individual (chromosome) is evaluated by the predefined fitness function which is designed by applying the problem to access goodness solutions. If a new solution is better than the current global optimal solution, it would be set as the new global optimal solution. Although GAs could be designed easily and generally perform well in different kinds of applications, two important drawbacks of GAs are that the convergence speed is slower than other evolutionary computations and setting crossover and mutation operators is a non-trivial task.

B. Particle Swarm Optimization

Particle swarm optimization (PSO) is another famous evolutionary method, proposed in 1995 by Kenny and Eberhar [29]. PSO is inspired by the behavior of birds flocking to find better food sources. Each particle is used to indicate a solution in PSO. The brief procedure of PSO is proposed in Algorithm 2. In the beginning, PSO would generate several particles in the solution space and set the velocity for each particle. Each particle would update its position by its velocity in each iteration and the velocity for each particle would also be updated by the global best *gbest* and it's personal best *pbest* in each iteration. The updating formula of velocity for each particle is given in Eq. (1).

$$\begin{aligned} v_i(t+1) = w \times v_i(t) + c_1 \times r_1 \times (pbest_i - x_i(t)) \\ + c_2 \times r_2 \times (gbest - x_i(t)) \end{aligned} \tag{1}$$

In the above equations, *w* is used to balance the influences of global search and local search. The v_i denotes the velocity of the *i*-th particle in the population and *t* is used to indicate the number of the current iteration. c_1 and c_2 are constants, respectively called the individual weight and social weight, and are both set to 2 usually. Both r_1 and r_2 are random numbers generated by a uniform distribution in the range of [0, 1]. Like GA, a predefined maximum generation *G* would be set as a termination condition.

PSO was always applied to find solutions from a continuous solution space. Kennedy and Eberhart proposed discrete PSO to find near optimal solutions to discrete optimization problems [30]. Then Lin et al. proposed a sanitization approach to hiding sensitive itemsets using this discrete PSO [34]. The experimental results showed that the PSO-based algorithm is much faster and generates smaller N-T-H (not to be hidden) side-effects compared to a GA-based algorithm.

C. Ant Colony System

Ant colony optimization (ACO) has been successfully applied to solving several optimization problems in the past. Specifically, they are effective in finding nearly-optimal solutions for a complex problem. There are several different kinds of ACOs proposed in the previous works. This section reviews work related to the original ant system (AS) and ant colony system (ACS), which is one of its extended versions.

1) Ant System

Ant system (AS) simulates the behavior of real ant colonies searching for food and was first introduced by Colorni et al. [12], [15]. By depositing pheromones on the path, a real ant population has the ability to find the shortest path between its nest and destinations with high probability. Each ant selects the next direction on the route according to pheromone density. AS observes the behavior of real ant populations, encodes the solution space in applied problems to a searching graph, and enhances the building solution process to improve the searching performance. Besides using the information of pheromones, AS would also design a heuristic function to influence ants to select a direction. The amount of pheromone on each tour, which are built in this iteration will have been modified once all the ants have finished their tours. The brief algorithm is shown in Algorithm 3. Details of the state transaction rule and the pheromone-updating process are described in section II-C2.

2) Ant Colony System

Ant Colony System (ACS), which was proposed by Dorigo et al. [14], is an extended algorithm from original ant system (AS). It uses the different state transaction rule and the different pheromone-updating rules with an AS approach in order to increase performance. The pseudo code is shown in Algorithm 4.

Details of the algorithm are described below.

1) State transaction rule
The state transaction rule is a process to determine an ant's next node (state) probabilistically. The traveling salesman problem (TSP) is taken as an example. Assume the *k*-th ant is currently in the city *j* (node). The next city *s* (node) for the *k*-th ant to visit is:

$$s = \begin{cases} \arg n \in \vec{R}_k(j) \max \left\{ [\tau(j,n)]^\alpha \times [\eta(j,n)]^\beta \right\} & \text{, if } q \le q_0 \\ i \text{ with a probability } P_k(j,i) & \text{, if } q > q_0 \end{cases} \tag{2}$$

where $R_k(j)$ is the set of cities that can be visited by the *k*-th ant; $\tau(j,n)$ is the accumulated pheromone on the edge between city *j* and city *n*; and α, β are two parameters that indicate the relative influence of pheromone versus the distance between two cities in this problem. In addition, the parameter *q* is a random number uniformly distributed between 0 and 1, q_0 is a parameter ($0 \le q_0 \le 1$), usually set as a number close to 1, and $P_k(j,i)$ is the probability from city *j* (node) to city *i* (node) if $q > q_0$. $P_k(j,i)$ is calculated as follows:

$$P_k(j,i) = \begin{cases} \frac{[\tau(j,i)]^\alpha \times [\eta(j,i)]^\beta}{\sum_{n \in R_k(j)} [\tau(j,n)]^\alpha \times [\eta(j,n)]^\beta} & \text{, if } i \in R_k(j) \\ 0 & \text{, otherwise} \end{cases} \tag{3}$$

The state transaction rule selects the node with the highest $[\tau(j,i)]^\alpha \times [\eta(j,i)]^\beta$ to follow, if the value of the random variable *q* is less than the pre-defined parameter q_0. Otherwise, it probabilistically selects a node according to Eq. (3). This combination selection method is called

Algorithm 1 Genetic Algorithm

Initialize
generate initial populations;
evaluate each individual in the population by the fitness function;
set the best individual as the global optimal solution *S*;
while does not satisfy the termination condition **do**
    generate the new individuals by the crossover/mutation operators;
    insert the new individuals into the population;
    evaluate each individual in the population by the fitness function;
    update the global optimal solution *S*;
    select some individuals as the new population;
end while
output the global optimal solution *S*;

Algorithm 2 Particle Swarm Optimization

Initialize
generate initial populations (position and velocity);
set the personal best *pbest* for each particle as current location;
evaluate each particle in population by the fitness function;
set the best particle as the global best *gbest*;
while does not satisfy the termination condition **do**
    update velocity for each particle;
    update position for each particle;
    evaluate each particle in the population by the fitness function;
    update the personal best *pbest* for each particle;
    update the global best *gbest*;
end while
output the global optimal solution *S*;

Algorithm 3 Ant System

Initialize
put each ant on its starting node;
while the ants have not built their tour **do**
    choose an ant which has not finished its tour;
    build a solution of an ant incrementally by the state transaction rule;
end while
update the pheromone information by each tour in this iteration;
output the best solution;

Algorithm 4 Ant Colony System

Initialize
put each ant on its starting node;
while the ants have not built their tour **do**
    choose an ant which has not finished its tour;
    while the ant has not finished its tour **do**
        select an edge by the pseudo random proportional rule;
        update the pheromone on selected edge using the local updating rule;
    end while
end while
update the pheromone using the global updating rule;
output the best solution;closest

the pseudo random proportional rule. In AS, just Eq. (3) is applied in its selection method, which is called random proportional rule. Pseudo-random proportional rule can increase the convergence speed effectively resulting in obtaining better solutions in early iterations.

In general, the heuristic function $\eta(j,i)$ is a specific formula related to the optimization. It is an effective design to lead ants to find better solutions in ant-based algorithms. An appropriate heuristic function is set according to the domain knowledge of the given application.

2) Global updating rule

The global updating rule will be applied to adjust the pheromone density on the best tour when all of the ants have already finished their tours. It is different from the pheromone-updating process in AS. AS increases the pheromones on the paths in all of the tours passed by the ants in each iteration. But ACS only increases the pheromones on the best tour. Therefore, the best solution would be distinguished between other feasible solutions by the global updating rule, thus this process increases the convergence speed sharply for ACS.

There are two kinds of global updating rules which can be selected. The first updating rule takes the best tour (the global best solution) among the ones in all the executed iterations and the second rule takes the best tour (the iterative best solution) in the current iteration. Take the traveling salesman problem as an example. Formally, the global updating rule is stated as follows:

$$\tau_{t+1}(j,s) = (1-\alpha) \times \tau_t(j,s) + \alpha \times \Delta\tau(j,s) \quad (4)$$

$$\Delta\tau(j,s) = \begin{cases} \frac{1}{L_{best}} & \text{, if } (j,s) \in \text{ best tour} \\ 0 & \text{, otherwise} \end{cases} \quad (5)$$

Where t is the current iteration, α $(0 < \alpha < 1)$ is the evaporation parameter of pheromone, and L_{best} is the length of the global best solution or the iterative best solution passed by the ants. Thus, the global updating rule will provide a greater increase of pheromone to a shorter tour in the traveling salesman problem.

3) Local updating rule

Due to the influences of the pseudo random proportional rule and the global updating rule, ACS always guides an ant population to pass the same edges. The local updating rule is designed to adjust the density of pheromones and to always avoid selecting similar tours in one iteration. When an ant chooses an edge between nodes j and s, it immediately updates the pheromone density of the edge to avoid local optimum as follows:

$$\tau_{after}(j,s) = (1-\rho) \times \tau_{before}(j,s) + \rho \times \tau_0 \quad (6)$$

where $\tau_{before}(j,s)$ is the pheromone amount on the edge from city j to s. When an ant passes through the edge, the pheromone amount will be updated to $\tau_{after}(j,s)$. And ρ $(0 < \rho < 1)$ is a parameter for adjusting the pheromone density. For this formula, if the pheromone density $\tau_{before}(j,s)$ on the constructed tour is higher than the initial pheromone τ_0, the local updating rule decreases the pheromone density to reduce repeated visiting probability from the edge. Otherwise, it increases the pheromone density.

The local updating rule gives a reasonable probability to any possible solution. An edge, which is the local optima for an ant but not global optima for all the ants, will decrease its pheromone output, causing the ant to explore the other edges with higher probabilities and thus avoiding local optima. According to the local updating process above, an ant usually decreases the pheromone density of the edge it chooses. This prevents the ant population from selecting similar paths and makes sure that other possible solutions have a good chance to be selected.

D. Privacy-Preserving Data Mining

Data mining has become a critical technology for business in recent years. It can discover valuable knowledge and useful relationships from transaction or customer databases. A manager or retailer can implement efficient and useful business strategies using this information. Notwithstanding the fact that, people can obtain huge benefits from these implicit relationships and knowledge, they might also get some private, confidential, or sensitive information from the original database. Therefore, before publishing or sharing a database containing critical data, confidential or secure information needs to be carefully hidden. For this reason, privacy-preserving data mining (PPDM) has recently emerged as a key research area [4], [7], [13], [17], [36], [44]. Agrawal and Srikant proposed a reconstruction process that accurately evaluates the distribution of original data structures, and defined several classifiers to estimate the accuracy of a sanitized database respective to its corresponding original database [7]. Verykios et al. proposed a hierarchical classification and an overview of PPDM approaches [44]. A generalization-based approach proposed by Hajian et al. preserves privacy and prevents discrimination [20]. This method has good scalability by extending to several privacy models. A novel approach proposed by Lindell et al. was used to hide privileged information from decision tree learning using the well-known ID3 classification method [36]. Evfimievski proposed some approaches to randomize categorical numerical data for PPDM [17]. Clifton et al. developed a toolkit and provided several techniques for some specific PPDM applications [11]. Islam and Brankovic constructed a method to protect numerical and categorical data which belonged privately to individuals [26]. A heuristic algorithm for data perturbation, which was proposed by Atallah et al. , adjusted an original dataset by decreasing the support of sensitive rules below a given user-specified threshold [9]. Oliveira and Zaiane proposed several sanitization approaches by a specific heuristic framework to hide frequent itemsets. These approaches rely on an item-restriction algorithm in order to avoid noise addition and limit the deletion of real data [39]. Sweeney provided an

algorithm dependent on a generalized *k*-anonymity to protect and suppress sensitive attributes [43].

Some sanitization algorithms, which are different from traditional hiding algorithms developed for PPDM, focus on minimizing the side effects of sanitization. Minimizing side effects of sanitization can be considered an NP-hard optimization problem [4], [44]. There are few evolutionary computation algorithms proposed to resolve the problem of hiding sensitive information in PPDM. Han and Ng proposed a genetic algorithm (GA) to define a secure protocol, which can discover a set of rules and ensure the non-disclosure of private data [22]. Lin et al. also developed several GAs, sGA2DT, pGA2DT [32] and cpGA2DT [33], for hiding sensitive itemsets by pruning transactions. If a chromosome indicated a subset of transactions in the database, then that chromosome would be deleted. Three side effects of sanitization were considered to design the fitness function in these three GAs. The fitness function combined the three factors of side effects with different weight as a unique function. sGA2DT process was designed as a simple GA with this specifically designed fitness to find an appropriate set of transactions to be deleted to hide sensitive itemsets. pGA2DT process is an extended version of sGA2DT, which adopted the pre-large concept [25] to speed up the evolution process. pGA2DT maintained a buffer of pre-large itemsets during the evolution process to avoid performing multiple database scans for each iteration. In general, a predefined size of a population needs to be set before performing GAs. cpGA2DT applies a compact GA [23] algorithm, which only generates two individuals in the population for each iteration. Thus, it can reduce memory usage during the evolution process effectively. Lin et al. proposed another evolutionary computation algorithm PSO2DT based on the PSO approach [35]. In this method, each particle represents a set of transactions to be deleted. Substantial experiments showed PSO2DT performs better than the Greedy algorithm and GA-based algorithm in terms of runtime, fail to be hidden (F-T-H), not to be hidden (N-T-H), and database similarity (DS). PSO2DT can even get a obtain performance than previous approaches. In this paper, we also applied another evolutionary computation algorithm, ACS approach, to further improve the performance of PPDM.

III. Preliminaries and Problem Statement

Preliminaries and the definition of hiding sensitive frequent itemsets while minimizing side effects will be introduced in this section.

A. Preliminaries

Let $I = \{i_1, i_2, \ldots, i_r\}$ be a finite set of *r* distinct items. A database $D = \{T_1, T_2, \ldots, T_n\}$ is a set of transactions. For each transaction $T_q \in D$, T_q is a subset of *I*, and *q* is a unique identifier for T_q called Transaction IDentifier (TID) for a specific transaction. A pre-defined minimum support threshold $\delta\,(0\% \sim 100\%)$ is assumed to be manually set by users or experts. Table I will be used as an example for the following description. It contains 10 transactions and each item is represented by a specific letter.

TABLE I
AN EXAMPLE FOR TRANSACTION DATABASE.

TID	Items
1	a, b, c, d
2	c, d, e
3	a, b
4	a, c, e
5	a, b, e
6	a, b, c, f
7	a, c, d, e
8	a, c, d
9	a, b, c
10	a, b, c

Definition 1. An itemset *i* is a subset of *I*. The support count of an itemset *i* in a database *D* is the number of transactions consisting of *i*. Let the minimum support count be the product of the minimum support threshold and the number of transactions in the database. The support count of a frequent itemset f is larger than the minimum support count in the database and can be defined as:

$$sup\,(f) \geq |D| \times \delta \tag{7}$$

$sup\,(f)$ is the support count of f and δ is the minimum support threshold. And the set of frequent itemsets in the database *D* is denoted as $FI_s = \{f_1, f_2, \ldots, f_k\}$.

For example, assume the minimum support threshold δ is set to 40%. Thus, the minimum support count is calculated as $10 \times 0.4 = 4$. The support count of the itemset {a, b} is $sup\,(\{a, b\}) = 6$ in Table I, since it appears in six transactions. Therefore, {a, b} is a frequent itemset in this database due to the high support count.

Definition 2. $SIs = \{s_1, s_2, \ldots, s_p\}$ is the set of sensitive itemsets if and only if SIs is a subset of FI_s $(SIs \subseteq FI_s)$ and each $s_i \in SIs$ needs to be hidden in the database *D*.

Hiding as much sensitive information as possible in a database is the purpose of PPDM. And yet it is not only used to prune sensitive information from a database, but also to minimize side effects of the pruning process. There are three major side effects of the pruning process, which include the failure to hide some sensitive information (called fail to be hidden, F-T-H, or hiding failure), hiding information that is important but not sensitive (called not to be hidden, N-T-H, or missing cost), and the introduction of artificial information (called not to be generated, N-T-G, or artificial cost) [45], discussed in this section. Definitions and explanations of these three side effects are given below.

Definition 3. Let *D'* be a sanitized database, which was generated by pruning some transactions/itemsets from an original database *D*.

In Figure 1, the example illustrates the relationship between the original database *D* and the sanitized database *D'*. FI_s indicates the set of frequent itemsets in dataset *D*, FI'_s is the set of frequent itemsets in the sanitized database *D'*, SIs is the set of sensitive itemsets that needs to be hidden and $\sim SIs$ is the set of non-sensitive frequent itemsets in *D*.

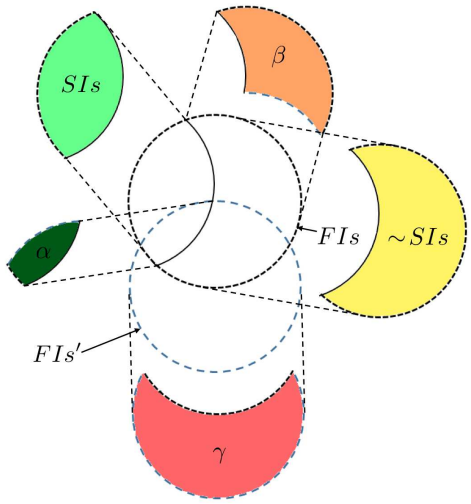

Fig. 1. Relationships of PPDM.

Definition 4. The side effect F-T-H (failure to hide some sensitive information) is denoted as α and defined as the number of sensitive itemsets appearing in the sanitized database *D'*, which is:

$$\alpha = |SIs \cap FIs'| \tag{8}$$

Definition 5. The side effect N-T-H (not to be hidden) is denoted as β and defined as the number of non-sensitive itemsets hidden in the sanitized database *D'*, which is:

$$\beta = |\sim SIs - FIs'| = |FIs - SIs - FIs'| \tag{9}$$

Definition 6. The side effect N-T-G (not to be generated) is denoted as γ and defined as the number of frequent itemsets in the sanitized database *D'* that was infrequent in the original database *D*, which is:.

$$\gamma = |FIs' - FIs| \tag{10}$$

B. Problem Statement

Due to hiding the sensitive itemsets $s_i \in SIs$ in a database *D*, the purpose of PPDM is to produce a sanitized database *D'* from *D* such that the support counts for all of the sensitive itemsets $s_i in SIs$ become less than the minimum support count, that is:

$$sup\,(s_i) < \delta \times |D| \tag{11}$$

The three aforementioned side effects are the standard measures used to assess the quality of a sanitization approach. High F-T-H means that too many sensitive patterns can still be found in the sanitized database. High N-T-H points out that some important information may be missing in the sanitized database. If the number of N-T-G is too high, it means that a huge amount of meaningless artificial information may have been generated by the sanitization process. There is a high possibility that important non-sensitive information would be hidden by the sanitization process if the amount of sensitive information which needs to be hidden is very substantial. The number of transactions would be decreased when some transactions were deleted in a database. Thus, the minimum support count would also be reduced. This causes several infrequent itemsets to become frequent as a result of the sanitization process. Thus, there is a trade-off relationship between the F-T-H, N-T-H, and N-T-G side effects. It is an NP-hard problem to find a solution to the problem of PPDM that minimizes the three side effects.

IV. Proposed ACS2DT Sanitization Algorithm

This section describes the proposed ant-based framework for efficiently hiding sensitive itemsets by transaction deletion. It is called ant colony system to delete transactions (ACS2DT). Each operator and detail algorithm of ACS2DT will be defined and explained in the following subsections. It contains the definitions of the ant routing graph, the termination conditions, the heuristic function and the pseudo code in this approach.

The proposed approach follows the traditional ACS process, which was introduced in section II-C, consists of the state transaction rule, pheromone updating rules, and selection process. Generally, ACS does not need to set a termination condition for each tour, since an ant would either finish its tour due to arriving at the destination or because no more following nodes could be selected. Because of the specifically defined ant routing graph in ACS2DT, some predefined termination conditions for each tour would be proposed to guide an ant toward finishing it's tour. This will be discribed in section IV-A and section IV-B. A well-defined heuristic function for hiding sensitive itemsets in ACS2DT will be proposed in section IV-D.

A. Ant Routing Graph

Before performing an ant-based algorithm, a routing graph would be encoded from the solution space. In ACS2DT, each node in the graph indicates a transaction in the dataset. Therefore, a completed tour for an ant is a subset of the input transaction dataset. And a solution produced from ACS2DT is an adjusted dataset which came from the original dataset's pruning of this subset. This is different from the traditional routing graph in ant-based algorithms, because there is no an obvious destination in the routing graph in ACS2DT. In ACS2DT, all of the nodes have an edge connecting to the nest node and they do not connect with each other. A simple example is shown in Figure 2.

In order to reduce the searching space, the proposed method does not map all of the transactions in the routing space. The proposed ACS2DT would extract a projected database D^* from the original database *D*. The process would only encode this projected database as the routing graph. The projected database D^* is the subset of transactions in *D* containing at least one sensitive itemset. It can be defined as Eq. (12), where *HS* is sensitive itemsets. A brief example of a projected database is shown in Table II.

$$D^* \leftarrow \{T_q \,|T_q \in D, \exists s_i \in HS, s_i \subseteq T_q\} \tag{12}$$

At the beginning of each tour, an ant would be put on the nest node. The ant will return back to the nest node after it has selected an edge and arrived at a node. An ant will finish

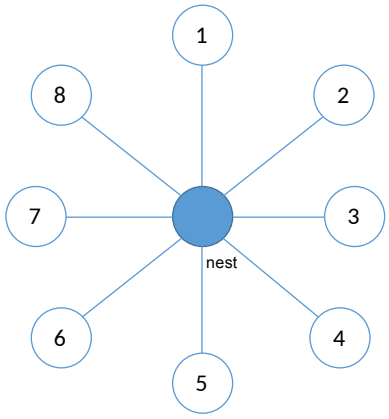

Fig. 2. A simple example for routing graph.

TABLE II
THE PROJECTED DATABASE OF $\{a, b\}$ AND $\{b, c\}$.

TID	Items
1	a, b, c, d
3	a, b
5	a, b, e
6	a, b, c, f
9	a, b, c
10	a, b, c

a tour when it selects several transactions (nodes) and reaches the termination condition. For example, the current transaction subset is {3, 5}, when an ant has been to the node C and the node D then back to the nest node in Figure 2.

B. Termination Condition for Each Tour

Due to the particular design of the routing graph, there is no obvious destination in ACS2DT. Therefore, a maximum number of selected transactions would be set in ACS2DT. An ant would select up to the maximum number of selected transactions. In PSO2DT, the appropriate number of transactions was defined to indicate the maximum number *m* of transactions to be deleted. The proposed method modifies the definition as Eq. (13):

$$m = n \times \left(\left\lfloor \frac{max_{s_i \in HS} sup\,(s_i) - \delta \times |D|}{1 - \delta} \right\rfloor + 1 \right) \qquad (13)$$

where $\left\lfloor \frac{max_{s_i \in HS} sup(s_i) - \delta \times |D|}{1 - \delta} \right\rfloor + 1$ is the minimum number of selected transactions theoretically, if the process would hide all of the sensitive itemsets. *n* is a number which is larger than 1 and dependent on different kinds of datasets. If the process can always select a transaction which contains the sensitive itemsets with the highest support counts to be deleted, *n* can be set as a number that approaches 1. If it cannot, it needs to be set as a number larger than 1 in order to hide all of the sensitive itemsets.

In Table II, it is a corresponding projected database D^* from Table I, if there are two sensitive itemsets {a,b} and {b, c} need to be hidden and assume $\delta = 0.4$. For this example, since $n = 1$, $sup\,(a, b) = 6$ and $sup\,(b, c) = 5$, therefore $max_{s_i \in HS} sup\,(s_i) = 6$. The maximum number of transactions to be deleted is calculated as: $m = \left\lfloor \frac{6 - 0.4 \times 10}{1 - 0.4} \right\rfloor + 1 = 4$. Then, an ant would select a least 4 nodes (transactions).

After determining the minimum number of selected transactions, the proposed method would compare the value of heuristic function (introduced in section IV-D) with current fitness function before performing pseudo random proportional rule. If there is no node in which the value of heuristic function is better than current fitness value, the selecting process would be interrupted and the ant would finish the tour.

C. Fitness Function

A fitness function in an ant-based algorithm is used to estimate each tour (solution) produced by the ants. Generally, ACS2DT follows the same fitness function from PSO2DT, but a rule is used to compare two tours if their fitness values are the same. The fitness function proposed from PSO2DT is the weighted sum of the three side effects, and is defined as:

$$fitness\,(p_i) = w_1 \times \alpha + w_2 \times \beta + w_3 \times \gamma \qquad (14)$$

where α is the number of F-T-H itemsets; β is the number of N-T-H itemsets, and γ is the number of N-T-G itemsets. The w_1, w_2 and w_3 are the weights of the relative importance of each side effect, which are adjusted by the user. In general, people always want to hide as much sensitive information as possible, even if this causes some non-sensitive information to go missing or some artificial information to be generated. Therefore, w_1 is always set at a higher number than the other two weighted numbers. Besides, the density of a database also influences the fitness of a tour. Since the data distribution is condensed in dense databases, if a sensitive itemset is successfully hidden, many non-sensitive but frequent itemsets may also be hidden as a side-effect. Thus, for dense databases, w_1 should be set higher to obtain a good balance between the three side effects. On the contrary, the distribution of data in sparse databases is scarce. Thus, if a sensitive itemset is hidden, fewer non-sensitive but frequent itemsets will be hidden as a side-effect. Hence, w_1 can be set lower for sparse datasets.

In ACS, the global pheromone updating rule would update the density of pheromones on the best tour. Sometimes, there are several tours that have the same fitness value and are better than the other tours. In this kind of situation, the proposed method would compare the size of selected transactions for these solutions. The proposed method would choose one solution as the best solution if it prunes fewer numbers of transactions from the original dataset. In this way, the proposed approach can always determine the solution which has higher similarity to the original dataset. Therefore, the process would help the proposed method to reduce side effects and increase similarity.

D. Heuristic Function

In ACS, a well-defined heuristic function is very important to guide an ant searching for a better direction. The proposed method would calculate the fitness value and record the current

situation after selecting each transaction. According to the current situation, the proposed method can select a suitable transaction to adjust or improve the current solution. As an example, presume the current selected transactions for deletion is transactions 1, 3 and 10. There are two sensitive itemsets $\{a, b\}$ and $\{c, d\}$ no hidden, one frequent itemset $\{e\}$ deleted and one itemset $\{f, g\}$ generated. The proposed method would prefer selecting a transaction which includes $\{a, b\}$, $\{c, d\}$ and $\{f, g\}$ but does not include $\{c, d\}$. Assume x is the number of non-hidden transactions which are included in the selected transaction, y is the number of hidden frequent itemsets in the selected transaction and z is the number of fake frequent itemsets in the selected transaction. The value of heuristic function is defined below:

$$H(e) = x \times \alpha - y \times \beta + z \times \gamma \tag{15}$$

where *H* is the heuristic function, *e* is a transaction which can be selected and α, β, γ are set as the same value with fitness function. By this function, the proposed method always selects a suitable transaction to adjust the current solution and reduce side effects effectively. Therefore, ants can refer the selected transactions so far and select the next transaction by this heuristic function to effectively reduce the side effects. This is the major difference and enhancement from previous works. The previous works did not consider the relationship between the selected transactions. However, the proposed method does not select the transactions separately; it tries to further obtain a suitable combination of the transactions.

E. The proposed Ant Colony System to Delete Transactions

The ACS2DT algorithm is proposed for efficiently hiding sensitive itemsets by transaction deletion. The pseudocode is provided in Algorithm 5.

The proposed ACS2DT algorithm first generated a projected dataset in order to generate an ant routing graph (line 1). Find the frequent itemsets by data mining algorithm (line 3). Initialize the pre-large frequent itemsets to speed up the mining process (line 4). Before starting the iterations, the process would calculate the maximum number of selected transactions as the termination condition (line 5). In each iteration, an ant would generate its own tour from line 8 to line 23. In general, an ant-based algorithm would estimate a solution after finishing a tour. But it is different in ACS2DT; the proposed approach would calculate the fitness value for each selection (line 19). Therefore, during a tour in ACS2DT, several suitable solutions would be produced. If a produced solution is better than the current global solution, it would be updated as the new global solution (line 21). Finally, if the process reaches the termination condition, the approach outputs the current global best solution and interrupts the process (line 27).

V. AN ILLUSTRATING EXAMPLE

In this section, an example for an ant to generate its own solution using the database shown in Table I is given to illustrate the proposed ACS2DT algorithm, step by step. In this example, the minimum support threshold is set to 40%. The discovered frequent itemsets (FIs) are shown in Table III. In the following description, the example focuses on selection process of the proposed method and ignores the ACS operators. Assume the itemsets $\{a, b\}$ and $\{b, c\}$ are the sensitive itemsets and these two frequent itemsets need to be hidden in the adjusted dataset. The same projected database of the sensitive itemsets is shown in Table II. At the beginning, an ant would select some of the transactions from the projected database to hide the sensitive itemsets. The maximum number of selected transactions would be calculated as: $\left\lfloor \frac{6-0.4\times10}{1-0.4} \right\rfloor + 1 = 4$. Then, the process would calculate the value of the heuristic function for each record in the projected database (Assume α is 0.9, β is 0.1 and δ is 0.1 in Eq. (14) and Eq. (15)). The results are shown below:

TABLE III
DISCOVERED FREQUENT ITEMSETS.

1-itemset	Count	2-itemset	Count	3-itemset	Count
{a}	9	{a,b}	6	{a,b,c}	4
{b}	6	{a,c}	5		
{c}	5	{b,c}	4		
{d}	4	{c,d}	4		
{e}	4				

1. a, b, c, d : 1.8
3. a, b : 0.9
5. a, b, e : 0.9
6. a, b, c, f : 1.8
9. a, b, c : 1.8
10. a, b, c: 1.8

In an ant-based algorithm, an edge (a candidate transaction in the projected database) would be selected with higher probability if it has higher fitness value. Assume the ant first selects transaction 1 for deletion from the original database. It causes the sensitive itemset $\{b, c\}$ to hidden but the frequent itemsets $\{d\}$, $\{c, d\}$ and $\{a, b, c\}$ are also removed. Thus, the fitness value for the current solution is $0.9 \times 1 + 0.1 \times 3 = 1.2$. The values of the heuristic function for the remaining transactions in the projected database are calculated below:

3. a, b : 0.9
5. a, b, e : 0.9
6. a, b, c, f : 0.8
9. a, b, c : 0.8
10. a, b, c : 0.8

The second selection of this ant is transaction 3. The process evaluates the current situation again. The sensitive itemset $\{b, c\}$ is still hidden and the itemset $\{a, b\}$ also exists in the frequent itemsets. The itemsets $\{d\}$, $\{c, d\}$ and $\{a, b, c\}$ are still removed from the frequent itemsets. Thus, the fitness value for the current solution is 1.2, the same as the previous stage. The updated values of the heuristic function for the remaining transactions in the projected database are calculated below:

5. a, b, e : 0.9
6. a, b, c, f : 0.8
9. a, b, c : 0.8
10. a, b, c : 0.8

The ant selects transaction 5 in the following third stage. This time, the current solution does not hide any sensitive itemsets

Algorithm 5 ACS2DT

Input: D^*, the set of projected transactions; SIs, a set of sensitive itemsets to be hidden; FI_s, the set of frequent itemsets in D^*; δ, the minimum support threshold; and *M*, the number of ant to be used for each iteration.
Output: *D*', a sanitized database.

1: *construct* the ant routing graph from transactions by section IV-A;
2: *set* $s_b = \emptyset$, $g = 0$;
3: *perform* data mining algorithm to find the frequent itemsets;
4: *initialize* the pre-large frequent itemsets to speed up mining process.
5: *calculate* the maximum number of selected transactions m_t by section IV-B;
6: **while** the ants which have not built their solution **do**
7: *choose* an ant which has not finished its tour;
8: *set* $g_b = \emptyset$ indicates the best solution of this tour;
9: **if** the number of selected transactions $= m_t$; **then**
10: the ant has already finished its tour;
11: the g_b is indicated the solution generated by this ant;
12: *update* the global best solution G_b;
13: **continuous;**
14: **end if**
15: *calculate* the value of heuristic function for all of candidate removed transactions by section IV-D;
16: *select* an edge from the candidate removed transactions to the next stage according to the pseudo random proportional rule by section II-C2;
17: *perform* pheromone local updating rule;
18: *set* current feasible solution as s_c;
19: *calculate* the fitness value $f(s_c)$ for current feasible solution s_c by section IV-C;
20: **if** $f(s_c) > f(g_b)$ **then**
21: $g_b \leftarrow s_c$;
22: **end if**
23: **go to** line 9;
24: **end while**
25: $g = g + 1$;
26: **if** $g = G$ **then**
27: **return** G_b;
28: **end if**
29: *process* pheromone global updating process;
30: **go to** line 6 for next iteration;

but the itemsets $\{d\}$, $\{c, d\}$ and $\{a, b, c\}$ are still deleted from the frequent itemsets. Thus, the fitness value is increased to 1.8 + 0.3 = 2.1. The values of the heuristic function for the following three transactions are calculated in the below:
6. a, b, c, f : 0.9
9. a, b, c : 0.9
10. a, b, c : 0.9
In the last selection, the ant selects transaction 10 for deletion from the database. Finally, the current solution hides all of the sensitive itemsets, removes two frequent itemsets $\{b\}$ and $\{a, b, c\}$ and generates threes fake transactions $\{a, e\}$, $\{c, d\}$ and $\{c, e\}$. The fitness value is updated as $0.1 \times 2 + 0.1 \times 3 = 0.5$.

The ant has already finished its tour and generated 4 candidate solutions. Comparing these 4 solutions, the last suitable solution (removing transactions 1, 3, 5 and 10) is the best as its the fitness value (0.5) is lower than others. Durning the proposed ant-based method, each ant would generate its own solution in the same way using the traditional ant operators (crossover, mutation, selecting and pheromone updating). When the process reaches the pre-defined maximum iteration, the proposed ACS2DT would output the best solution it had generated up to this point.

VI. Experimental Results

Substantial experiments were conducted to compare the effectiveness and efficiency of the proposed ACS2DT algorithm with PSO2DT [35], GA-based sGA2DT, pGA2DT, cpGA2DT approaches [32], [33] and the non-evolutionary Greedy sanitization algorithm [31]. The proposed ACS2DT algorithm, like PSO2DT, adopts the pre-large concept to avoid performing multiple database scans to evaluate side effects during the process. The algorithms in the experiments were implemented in Java language on the supercomputer Cherry-creek with Intel Xeon E5-2697v2 CPU and each experiment was performed by one core and 15GB ram, running the 64-bit CentOS 6.6 Linux operating system. There are three real-world datasets called chess [2], mushroom [2] and foodmart [1] used in these experiments. A synthetic dataset named T10I4D100K has also been generated using the IBM database generator

[3]. Parameters and characteristics of the datasets used in the experiments are respectively shown in Table IV and Table V.

TABLE IV
PARAMETERS OF USED DATASETS.

$\#\|\mathbf{D}\|$	Total number of transactions
$\#\|\mathbf{I}\|$	Number of distinct items
AvgLen	Average transaction length
MaxLen	Maximal length transactions
Type	Dataset type

TABLE V
CHARACTERISTICS OF USED DATASETS.

Dataset	$\#\|\mathbf{D}\|$	$\#\|\mathbf{I}\|$	AvgLen	MaxLen	Type
chess	3196	76	37	37	dense
mushroom	8124	120	23	23	dense
foodmart	21557	1559	4	11	sparse
T10I4D100K	100000	870	10.1	29	sparse

The maximum number of iterations is set at 1,000 and the size of the population is set at 10 in the conducted experiments for the previous GA-based and PSO-based approaches. Due to the well-defined heuristic function, the proposed ACS2DT makes it unnecessary to perform many iterations to obtain a good solution. Thus, the maximum number of iterations is set as 4 and the size of the population is set as 2 for the proposed ACS2DT. Fewer iterations and population can also reduce the runtime of the proposed algorithm but can still obtain the optimal results. All experiments were executed five times and solutions having the smallest fitness values were used to determine the transactions to be deleted for hiding the sensitive itemsets. A certain percentage of the frequent itemsets found in each database were randomly selected to be the sensitive itemsets. This percentage is called the percentage of sensitive itemsets. In the following, the minimum support threshold and the percentage of sensitive itemsets are respectively denoted as *min_sup* and *sen_per*. In practice, the values of *min_sup* and *sen_per* can be specified by the users or experts. To evaluate the performance of the designed approach, the *min_sup* and *sen_per* values have been adjusted for each dataset, to ensure that the number of frequent itemsets and sensitive itemsets is appropriate. Thus, in the conducted experiments, the *min_sup* and *sen_per* parameters are different for each dataset, and were adjusted based on each datasets characteristics. According to different datasets, the parameters of fitness functions and *n* value of maximum number of deleted transactions are set in Table VI.

A. Fitness Value

A first experiment was conducted to compare the fitness value of the designed algorithm with the Greedy, sGA2DT, pGA2DT, cpGA2DT and PSO2DT algorithms. The weights of the three side effects in the fitness function were set in Table VI. In order to hide as much sensitive information as possible, w_1 is set much higher than the other two weights in our experiments. The fitness values obtained for various minimum support threshold values are shown in Figure 3.

TABLE VI
EXPERIMENTAL PARAMETERS FOR EACH DATASETS.

Dataset	*min_sup*	*sen_per*	α	β	γ	*n*
chess	0.9	0.01	0.98	0.01	0.01	1.01
chess	0.9	0.02	0.98	0.01	0.01	1.03
chess	0.9	0.03	0.98	0.01	0.01	1.02
chess	0.9	0.04	0.98	0.01	0.01	1.02
chess	0.9	0.05	0.98	0.01	0.01	1.03
chess	0.85	0.01	0.98	0.01	0.01	1.03
chess	0.86	0.01	0.98	0.01	0.01	1.08
chess	0.87	0.01	0.98	0.01	0.01	1.03
chess	0.88	0.01	0.98	0.01	0.01	1.08
chess	0.89	0.01	0.98	0.01	0.01	1.02
foodmart	0.0025	0.01	0.8	0.1	0.1	8
foodmart	0.0025	0.015	0.8	0.1	0.1	10
foodmart	0.0025	0.02	0.8	0.1	0.1	8.5
foodmart	0.0025	0.025	0.8	0.1	0.1	8.5
foodmart	0.0025	0.03	0.8	0.1	0.1	8.5
foodmart	0.002	0.015	0.8	0.1	0.1	8.5
foodmart	0.0023	0.015	0.8	0.1	0.1	8.5
foodmart	0.0026	0.015	0.8	0.1	0.1	8.5
foodmart	0.0029	0.015	0.8	0.1	0.1	8.5
foodmart	0.0032	0.015	0.8	0.1	0.1	8.5
mushroom	0.4	0.05	0.98	0.01	0.01	2
mushroom	0.4	0.06	0.98	0.01	0.01	2
mushroom	0.4	0.07	0.98	0.01	0.01	2
mushroom	0.4	0.08	0.98	0.01	0.01	2
mushroom	0.4	0.09	0.98	0.01	0.01	2
mushroom	0.41	0.1	0.98	0.01	0.01	2
mushroom	0.42	0.1	0.98	0.01	0.01	2
mushroom	0.43	0.1	0.98	0.01	0.01	2
mushroom	0.44	0.1	0.98	0.01	0.01	2
mushroom	0.45	0.1	0.98	0.01	0.01	2
T10I4D100K	0.009	0.1	0.8	0.1	0.1	8
T10I4D100K	0.009	0.2	0.8	0.1	0.1	8
T10I4D100K	0.009	0.3	0.8	0.1	0.1	8
T10I4D100K	0.009	0.4	0.8	0.1	0.1	8
T10I4D100K	0.009	0.5	0.8	0.1	0.1	8
T10I4D100K	0.01	0.4	0.8	0.1	0.1	8
T10I4D100K	0.011	0.4	0.8	0.1	0.1	8
T10I4D100K	0.012	0.4	0.8	0.1	0.1	8
T10I4D100K	0.013	0.4	0.8	0.1	0.1	8
T10I4D100K	0.014	0.4	0.8	0.1	0.1	8

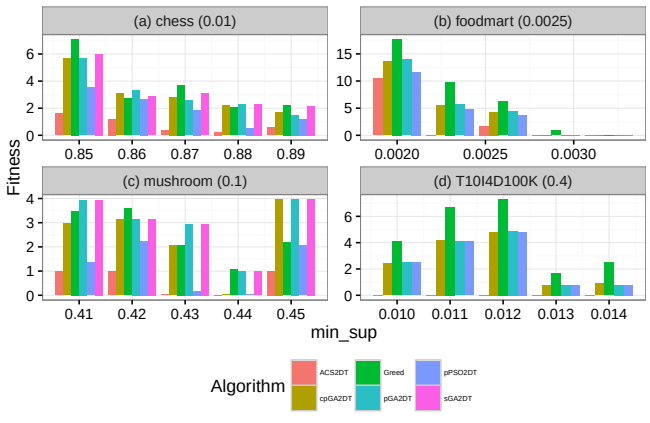

Fig. 3. Fitness values for various minimum support threshold values.

In Figure 3, the fitness values of the proposed ACS2DT are obviously better than previous approaches. Due to the well-defined heuristic function in ACS2DT, the proposed method not only hides the sensitive itemsets effectively but reduces the other two side effects successfully. Therefore, the proposed method can further decrease the value of fitness function. The experimental results (in the following subsection) showed that ACS2DT can resolve the trade-off issue between F-T-H, N-T-H, and N-T-G in previous algorithms. Previous algorithms always focused on hiding sensitive itemsets, but ignored the other side effects in their solutions. However, ACS2DT can also hide more sensitive itemsets than other previous methods in most situations (dense type or sparse type datasets) and has lower N-T-G and N-T-H side effects. The fitness values obtained for various sensitive percentages of FIs are shown in Figure 4.

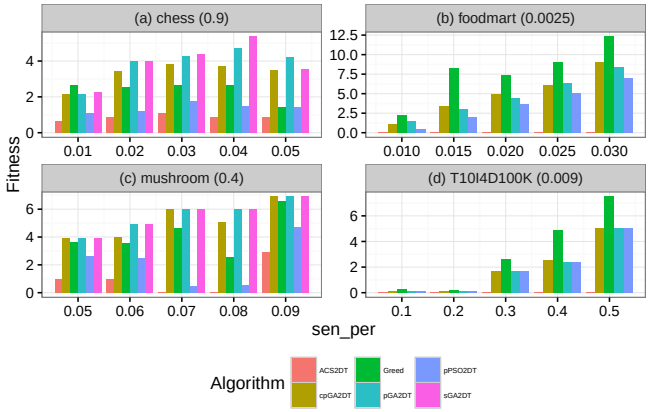

Fig. 4. Fitness values for various sensitive percentages of FIs.

In Figure 4, results are similar to Figure 3. The fitness values of ACS2DT are all better than other previous algorithms and approaches or reach zero. In T10I4D100K dataset, the proposed method generates almost no side effects and hides all of the sensitive itemsets in all kinds of situations. From the above discussion, it can be concluded that the proposed ACS2DT algorithm considers the three side effects simultaneously and can obtain better results than previous works.

B. Side Effects

In this section, the three side effects (F-T-H, N-T-H and N-T-G) are discussed to show the performance for proposed ACS2DT and the other previous algorithms. The standard way of comparing sanitization algorithm in PPDM is to compare the number of occurrence of the three side effects that each algorithm generates. The number of occurrence of the three side effects are shown in Figure 5 and Figure 6.

In the experimental results, the side effects in PSO2DT and ACS2DT are both less than the GA-based algorithms and Greedy algorithm. PSO2DT showed a greater ability to hide the sensitive itemsets than the GA-based algorithms and Greedy algorithm, but it also generated too much N-T-H and N-T-G side effects in some datasets. This means PSO2DT usually avoids N-T-H and N-T-G side effects in order to reduce

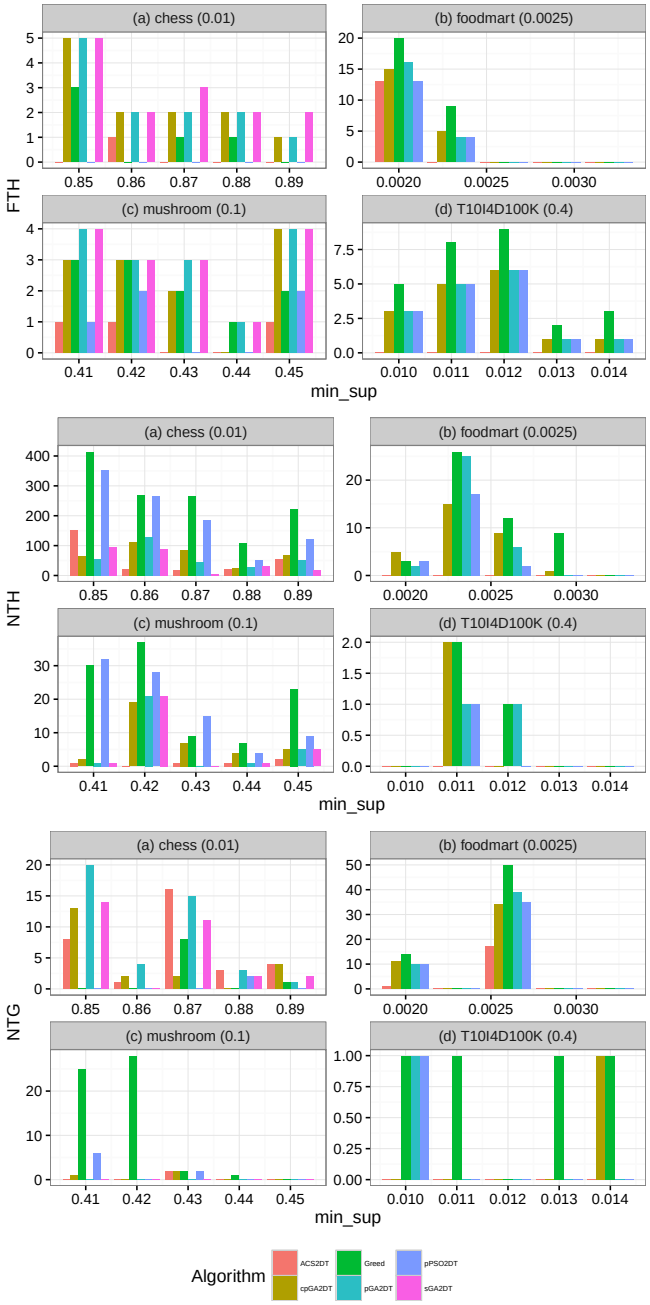

Fig. 5. Side effects w.r.t various minimum support threshold values.

F-T-H side effect as much as possible. Even ACS2DT can hide more sensitive itemsets than PSO2DT in most of the datasets, but the proposed method does not generate a large number of the other side effects compared to PSO2DT. Due to the well-defined heuristic function and the characteristics of ACS, ACS2DT selects the deleted transactions one-by-one by always referring to the previous selected transactions then choosing a suitable transaction to reduce the three side effects in the current situation. It is worth noting that the result of the chess dataset with parameters (*sen_per* = 0.01 and *min_sup* = 0.86) in Figure 5. This is strong evidence to show that PSO2DT focuses on hiding sensitive itemsets but avoids the others side effects. Even PSO2DT hides all of the sensitive

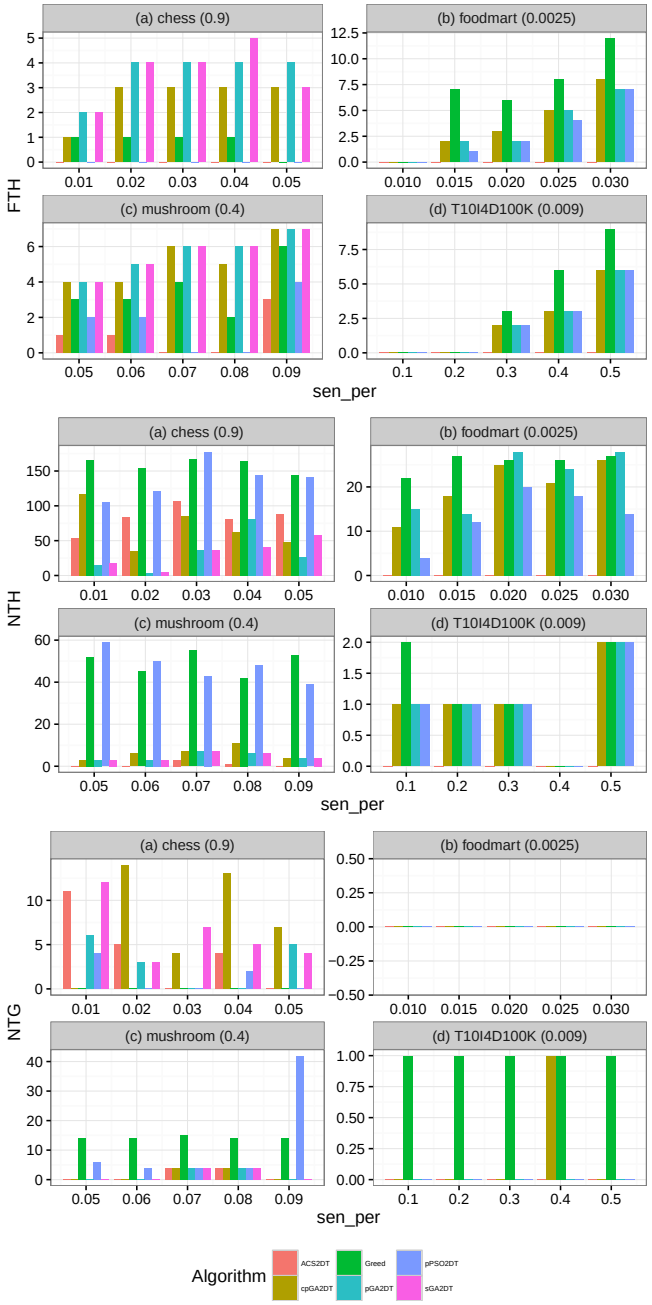

Fig. 6. Side effects w.r.t various sensitive percentages of FIs.

itemsets in this situation but generated 267 itemsets with N-T-H side effect in its solution (fitness: 2.67). ACS2DT fails to hide a sensitive itemset in the same experiment but only generated 22 itemsets in N-T-H and N-T-G side effects (fitness: 1.2). Therefore, the proposed method is the only method in the compared approaches that can efficiently consider all of three side effects in PPDM data mining.

C. Database Similarity

Besides the three side effects, the database similarity (DS) between an original database and a sanitized database should be evaluated to examine the size difference between these two databases. If the difference is similar for an algorithm than for the other algorithms, it means that the algorithm has selected a better set of transactions for deletion, and thus avoids deleting irrelevant transactions that may result in hiding more non-sensitive but frequent itemsets. The DS is defined as:

$$DS = \frac{|D^*|}{|D|} \tag{16}$$

where $|D|$ is the number of transactions in the original database D, and $|D^*|$ is the number of transactions in the sanitized database D^*.

Result w.r.t various minimum support threshold values and sensitive percentage of FIs are respectively shown in Figure 7 and Figure 8.

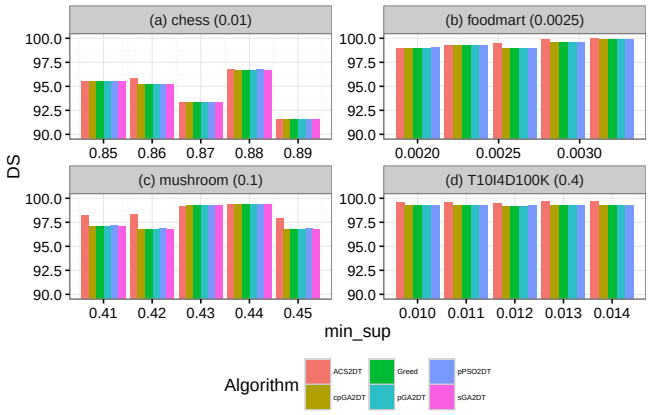

Fig. 7. DS w.r.t various minimum support threshold values.

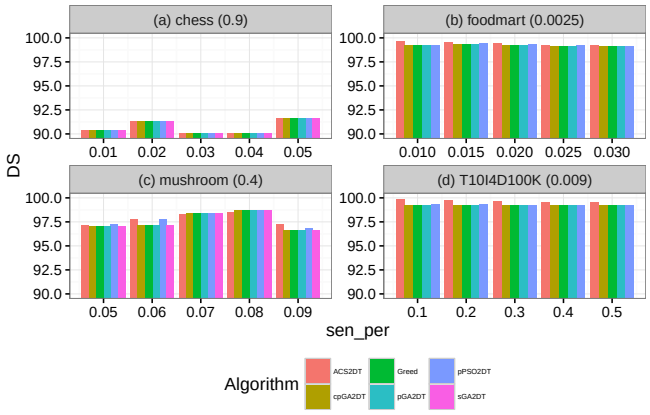

Fig. 8. DS w.r.t various sensitive percentages of FIs.

In ACS2DT, the algorithm records the best solution during the selecting process. It would retain a solution that contains fewer selected transactions if two suitable solutions have the same fitness value. Thus, ACS2DT not only improves the fitness value but reduces the number of selected transactions. In a chess dataset, the termination conditions are very strict. Thus, DS values for all compared algorithms are almost the same. In a mushroom dataset, ACS2DT selects more transactions in order to get a better fitness values in some experiments. However, ACS2DT also can get higher DS value in some experiments. Finally, ACS2DT can both get a better fitness

value and higher DS value in foodmart and T10I4D100K databases. In summary, the proposed ACS2DT can obtain higher DS value and also get a better fitness value than the previous algorithms.

D. Runtime

To further compare the designed algorithm with PSO-based, GA-based, and Greedy algorithms, the runtime value has also been compared. Results are shown in Figure 9 and Figure 10. Since Greedy does not use an evolutionary approach, this algorithm is always executed only once. The runtimes of Greedy are less than evolutionary approaches in Figure 9 and 10.

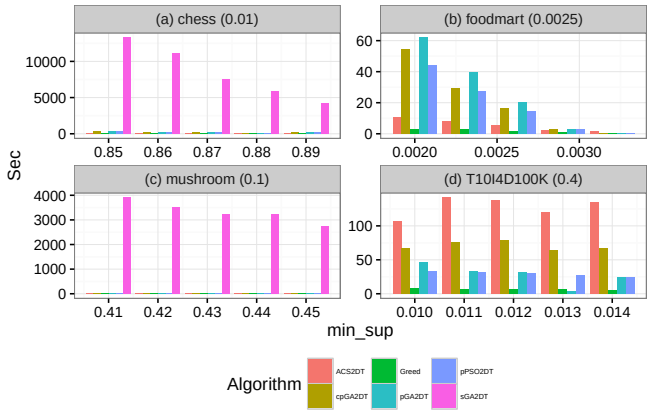

Fig. 9. Runtime w.r.t various minimum support threshold values.

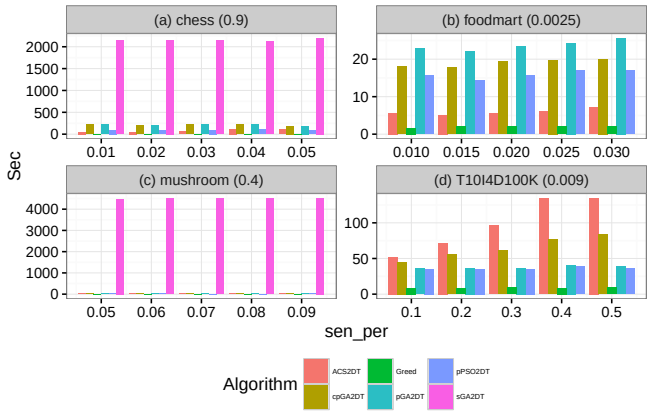

Fig. 10. Runtime w.r.t various sensitive percentages of FIs.

In ACS2DT, the proposed algorithm would perform fitness function one time after selecting a new transaction in the current solution. ACS2DT also needs to calculate the value of the heuristic function for all candidate transactions. Therefore, the maximum number of selected transactions and the size of the projected dataset have a big influence over runtime in ACS2DT. ACS2DT spends less or similar computation time in chess, mushroom, and foodmart datasets. But it spends more runtime in T10I4D100K dataset, because it needs to calculate more fitness functions and evaluate more heuristic functions.

VII. Conclusion and Future Work

In the past, several kinds of evolutionary computations, including GA-based algorithms and PSO2DT, have been proposed to hide sensitive itemsets through transaction deletion. This is the first paper that designs an ACS-based approach to hide sensitive itemsets through transaction deletion, while minimizing side effects. In the proposed ACS2DT algorithm, a termination condition is proposed to determine the number of transactions that would be deleted. The proposed algorithm is more effective than previous GA-based, PSO-based and the non-evolutionary Greedy algorithms. Furthermore, the designed ASC2DT algorithm also applied the pre-large concept to avoid performing multiple database scans and avoid spending similar computation time with PSO2DT, even though it performs more complicated computation. Substantive experiments have been carried out to compare the performance of the designed algorithm with the previous evolutionary computation approaches and Greedy algorithm in terms of the three side effects and database similarity (integrity). Experimental results show that the designed algorithm can generate a smaller number of these three side-effects (N-T-H, N-T-G and F-T-H) compared to PSO-based, GA-based and Greedy approaches in all kinds of datasets.

In this paper, an ACS-based approach was designed. But it remains possible to design parallel framework based on cloud computing such as Map-Reduce framework, to increase performance for hiding sensitive itemsets. However, it is a non-trivial task to define the formulas for hiding sensitive itemsets through transaction deletion using a parallel framework. Another research opportunity for future work is to consider the privacy preserving issue discussed in this paper as a multi-objective optimization problem where criteria such as F-T-H, N-T-H, N-T-G, and DS are considered as an object. Besides GA-based, PSO-based or ACO-based approaches, the NSGA2, SPEA, or Pareto optimization approaches can be further involved to solve the problems of PPDM, which can be used to find the optimized balance between different side effects.

Acknowledgment

We gratefully acknowledge the United States Department of Defense (DoD grant # W911NF-17-1-0088, W911NF-16-1-0416) the National Science Foundation (NSF grant # 1710716, 1625677), and Oak Ridge National Laboratory (ORNL Contract # 4000144962) and United Hall Foundation (UHF # 1592) for their support and finance for this project.

References

[1] Example database foodmart of Microsoft analysis services. URL: http://msdn.microsoft.com/en-us/library/aa217032(SQL.80).aspx.
[2] Frequent Itemset Mining Dataset Repository. URL: http://fimi.ua.ac.be/data/.
[3] IBM quest synthetic data generator. URL: https://sourceforge.net/projects/ibmquestdatagen/.
[4] Charu C Aggarwal, Jian Pei, and Bo Zhang. On privacy preservation against adversarial data mining. In *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 510–516, 2006.
[5] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *The International Conference on Very Large Data Bases*, vol. 1215, pp. 487–499, 1994.

[6] Rakesh Agrawal and Ramakrishnan Srikant. Mining sequential patterns. In *IEEE International Conference on Data Engineering*, pp. 3–14, 1995.

[7] Rakesh Agrawal and Ramakrishnan Srikant. Privacy-preserving data mining. In *ACM SIGMOD Record*, vol. 29, pp. 439–450, 2000.

[8] Muhammad Zubair Hasan Ansari, Sohail Asghar, Yan Zhuang, and Simon Fong. A Data Modification Technique in PPDM Based on Ant Colony Optimization Approach. In *Advances in Digital Technologies: Proceedings of the 6th International Conference on Applications of Digital Information and Web Technologies 2015*, vol. 275, pp. 139, 2015.

[9] Mike Atallah, Elisa Bertino, Ahmed Elmagarmid, Mohamed Ibrahim, and Vassilios Verykios. Disclosure limitation of sensitive rules. In *The Workshop on Knowledge and Data Engineering Exchange*, pp. 45–52, 1999.

[10] Ming-Syan Chen, Jiawei Han, and Philip S Yu. Data mining: an overview from a database perspective. *IEEE Transactions on Knowledge and Data Engineering*, vol. 8(6), pp. 866–883, 1996.

[11] Chris Clifton, Murat Kantarcioglu, Jaideep Vaidya, Xiaodong Lin, and Michael Y Zhu. Tools for privacy preserving distributed data mining. *ACM Sigkdd Explorations Newsletter*, vol. 4(2), pp. 28–34, 2002.

[12] Alberto Colorni, Marco Dorigo, and Vittorio Maniezzo. Distributed optimization by ant colonies. In *The first European conference on artificial life*, vol. 142, pp. 134–142, 1991.

[13] Elena Dasseni, Vassilios S Verykios, Ahmed K Elmagarmid, and Elisa Bertino. Hiding association rules by using confidence and support. In *International Workshop on Information Hiding*, pp. 369–383, 2001.

[14] Marco Dorigo and Luca Maria Gambardella. Ant colony system: a cooperative learning approach to the traveling salesman problem. *IEEE Transactions on Evolutionary Computation*, vol. 1(1), pp. 53–66, 1997.

[15] Marco Dorigo, Vittorio Maniezzo, and Alberto Colorni. Ant system: optimization by a colony of cooperating agents. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, vol. 26(1), pp. 29–41, 1996.

[16] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In *Theory of Cryptography Conference*, pp. 265–284, 2006.

[17] Alexandre Evfimievski, Ramakrishnan Srikant, Rakesh Agrawal, and Johannes Gehrke. Privacy preserving mining of association rules. *Information Systems*, vol. 29(4), pp. 343–364, 2004.

[18] Philippe Fournier-Viger, Jerry Chun-Wei Lin, Rage Uday Kiran, Yun Sing Koh, and Rincy Thomas. A Survey of Sequential Pattern Mining. *Data Science and Pattern Recognition*, vol. 1(1), pp. 54–77, 2017.

[19] David E Golberg. Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989.

[20] Sara Hajian, Josep Domingo-Ferrer, and Oriol Farrs. Generalization-based privacy preservation and discrimination prevention in data publishing and mining. *Data Mining and Knowledge Discovery*, vol. 28(5-6), pp. 1158–1188, 2014.

[21] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, vol. 8(1), pp. 53–87, 2004.

[22] Shuguo Han and Wee Keong Ng. Privacy-preserving genetic algorithms for rule discovery. In *International Conference on Data Warehousing and Knowledge Discovery*, pp. 407–417, 2007.

[23] Georges R Harik, Fernando G Lobo, and David E Goldberg. The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation*, vol. 3(4), pp. 287–297, 1999.

[24] Tzung-Pei Hong, Chun-Wei Lin, Kuo-Tung Yang, and Shyue-Liang Wang. Using TF-IDF to hide sensitive itemsets. *Applied Intelligence*, vol. 38(4), pp. 502–510, 2013.

[25] Tzung-Pei Hong, Ching-Yao Wang, and Yu-Hui Tao. A new incremental data mining algorithm using pre-large itemsets. *Intelligent Data Analysis*, vol. 5(2), pp. 111–129, 2001.

[26] Md Zahidul Islam and Ljiljana Brankovic. Privacy preserving data mining: a noise addition framework using a novel clustering technique. *Knowledge-Based Systems*, vol. 24(8), pp. 1214–1223, 2011.

[27] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM Computing Surveys*, vol. 31(3), pp. 264–323, 1999.

[28] Holland John. *Adaptation in natural and artificial systems*. MIT Press, Cambridge, MA, 1992.

[29] James Kenndy and RC Eberhart. Particle swarm optimization. In *IEEE International Conference on Neural Networks*, vol. 4, pp. 1942–1948, 1995.

[30] James Kennedy and Russell C Eberhart. A discrete binary version of the particle swarm algorithm. In *IEEE International Conference on Systems, Man, and Cybernetics*, vol. 5, pp. 4104–4108, 1997.

[31] Chun-Wei Lin, Tzung-Pei Hong, Chia-Ching Chang, and Shyue-Liang Wang. A greedy-based approach for hiding sensitive itemsets by transaction insertion. *Journal of Information Hiding and Multimedia Signal Processing*, vol. 4(4):, pp. 201–227, 2013.

[32] Chun-Wei Lin, Tzung-Pei Hong, Kuo-Tung Yang, and Shyue-Liang Wang. The GA-based algorithms for optimizing hiding sensitive itemsets through transaction deletion. *Applied Intelligence*, vol. 42(2), pp. 210–230, 2015.

[33] Chun-Wei Lin, Binbin Zhang, Kuo-Tung Yang, and Tzung-Pei Hong. Efficiently hiding sensitive itemsets with transaction deletion based on genetic algorithms. *The Scientific World Journal*, vol. 2014, 2014.

[34] Jerry Chun-Wei Lin, Qiankun Liu, Philippe Fournier-Viger, Tzung-Pei Hong, Miroslav Voznak, and Justin Zhan. A sanitization approach for hiding sensitive itemsets based on particle swarm optimization. *Engineering Applications of Artificial Intelligence*, vol. 53, pp. 1–18, 2016.

[35] Jerry Chun-Wei Lin, Lu Yang, Philippe Fournier-Viger, Ming-Thai Wu, Tzung-Pei Hong, and Leon Shyue-Liang Wang. A swarm-based approach to mine high-utility itemsets. In *Multidisciplinary Social Networks Research*, pp. 572–581, 2015.

[36] Yehuda Lindell and Benny Pinkas. Privacy preserving data mining. In *Annual International Cryptology Conference*, pp. 36–54, 2000.

[37] Carl H Mooney and John F Roddick. Sequential pattern mining approaches and algorithms. *ACM Computing Surveys*, vol. 45(2), pp. 19, 2013.

[38] S Narmadha. Privacy Preserving Data Mining based on Ant Colony Optimization. *International Journal of Computer Applications*, vol. 83(8), 2013.

[39] Stanley RM Oliveira and Osmar R Zaiane. Privacy preserving frequent itemset mining. In *IEEE International Conference on Privacy, Security and Data Mining*, pp. 43–54, 2002.

[40] Bhupendra Kumar Pandya, Keerti Dixit, Umesh Kumar Singh, and Kamal Bunkar. Effectiveness of multiplicative data perturbation for privacy preserving data mining. *International Journal of Advanced Research in Computer Science*, vol. 5(6), 2014.

[41] J Ross Quinlan. C4.5: programming for machine learning. *Morgan Kauffmann*, pp. 38, 1993.

[42] P Tamil Selvan and S Veni. Social Ant based Sensitive Item Hiding with Optimal Side Effects for Data Publishing. *Indian Journal of Science and Technology*, vol. 9(2), 2016.

[43] Latanya Sweeney. k-anonymity: a model for protecting privacy. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 10(05), pp. 557–570, 2002.

[44] Vassilios S Verykios, Elisa Bertino, Igor Nai Fovino, Loredana Parasiliti Provenza, Yucel Saygin, and Yannis Theodoridis. State-of-the-art in privacy preserving data mining. *ACM Sigmod Record*, vol. 33(1), pp. 50–57, 2004.

[45] Yi-Hung Wu and Chia-Ming Chiang. Hiding sensitive association rules with limited side effects. *IEEE Transactions on Knowledge and Data Engineering*, vol. 19(1), pp. 29–42, 2007.

[46] Mohammed J Zaki. SPADE: an efficient algorithm for mining frequent sequences. *Machine Learning*, vol. 42(1-2), pp. 31–60, 2001.

[47] Xiao-Hu Zhi, XL Xing, QX Wang, LH Zhang, XW Yang, CG Zhou, and YC Liang. A discrete PSO method for generalized TSP problem. In *IEEE International Conference on Machine Learning and Cybernetics*, vol. 4, pp. 2378–2383, 2004.

Jimmy Ming-Tai Wu is a Postdoctoral Research Fellow in the Department of Computer Science, University of Nevada, Las Vegas. He was a Postdoctoral Research Fellow in the Department of Computer Science, National University of Kaohsiung, Taiwan. His research interests include data mining, artificial intelligence, bioinformation and big data.

Justin Zhan is the Director of Big Data Hub and Director of ILAB. He is a faculty member are the Department of Computer Science, University of Nevada, Las Vegas. He has been previously been a faculty member at the Carnegie Mellon University, National Center for the Protection of Financial Infrastructure in Dakota State University and North Carolina A&T State University. His research interests include big data, information assurance, social computing, and medical informatics. His research has been sponsored by the National Science Foundation, Department of Defense, and National Institute of Health.

Jerry Chun-Wei Lin is currently working as an Associate Professor at School of Computer Science and Technology, Harbin Institute of Technology-Shenzhen, China. He has published more than 200 research papers in referred journals and international conferences. His research interests include data mining, soft computing, artificial intelligence, social computing, multimedia and image processing, and privacy-preserving and security technologies. He is also the project leader of SPMF: An Open-Source Data Mining Library, which is a toolkit to collect and implement the library for variants of data mining algorithms. He also serves as the Editor-in-Chief of an international journal of Data Science and Pattern Recognition