Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

journal homepage: www.elsevier.com/locate/engappai

# A sanitization approach for hiding sensitive itemsets based on particle swarm optimization

Jerry Chun-Wei Lin[a,*], Qiankun Liu[a], Philippe Fournier-Viger[b], Tzung-Pei Hong[c], Miroslav Voznak[d], Justin Zhan[e]

[a] School of Computer Science and Technology, Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen, China
[b] School of Natural Sciences and Humanities, Harbin Institute of Technology, Shenzhen Graduate School, Shenzhen, China
[c] Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan
[d] Faculty of Electrical Engineering and Computer Science, VSB-Technical University of Ostrava, Ostrava-Poruba, Czech Republic
[e] Department of Computer Science, University of Nevada, Las Vegas, USA

A R T I C L E  I N F O

*Article history:*
Received 8 July 2015
Received in revised form
20 March 2016
Accepted 21 March 2016

*Keywords:*
PPDM
Sanitization
Evolutionary computation
Sensitive itemsets
PSO

A B S T R A C T

Privacy-preserving data mining (PPDM) has become an important research field in recent years, as approaches for PPDM can discover important information in databases, while ensuring that sensitive information is not revealed. Several algorithms have been proposed to hide sensitive information in databases. They apply addition and deletion operations to perturb an original database and hide the sensitive information. Finding an appropriate set of transactions/itemsets to be perturbed for hiding sensitive information while preserving other important information is a NP-hard problem. In the past, genetic algorithm (GA)-based approaches were developed to hide sensitive itemsets in an original database through transaction deletion. In this paper, a particle swarm optimization (PSO)-based algorithm called PSO2DT is developed to hide sensitive itemsets while minimizing the side effects of the sanitization process. Each particle in the designed PSO2DT algorithm represents a set of transactions to be deleted. Particles are evaluated using a fitness function that is designed to minimize the side effects of sanitization. The proposed algorithm can also determine the maximum number of transactions to be deleted for efficiently hiding sensitive itemsets, unlike the state-of-the-art GA-based approaches. Besides, an important strength of the proposed approach is that few parameters need to be set, and it can still find better solutions to the sanitization problem than GA-based approaches. Furthermore, the pre-large concept is also adopted in the designed algorithm to speed up the evolution process. Substantial experiments on both real-world and synthetic datasets show that the proposed PSO2DT algorithm performs better than the Greedy algorithm and GA-based algorithms in terms of runtime, fail to be hidden (F-T-H), not to be hidden (N-T-H), and database similarity (DS).

## 1. Introduction

With the rapid growth of information technology and e-commerce applications, it has become increasingly easy to discover useful information and interesting relationships in huge amount of data. Data mining, also called knowledge discovery in database (KDD), provides a set of techniques, commonly used to analyze relationships among purchased products for market basket analysis. Knowledge discovered using KDD techniques can be generally classified as association rules (Agrawal and Srikant, 1994b; Chen et al., 1996; Han et al., 2004), sequential patterns (Agrawal and Srikant, 1995; Mooney and Roddick, 2013; Zaki, 2001), clusters (Murty and Flynn, 1999), and classifications (Quinlan, 1993). Association rule mining (Agrawal and Srikant, 1994b; Chen et al., 1996) is a fundamental KDD task, which consists of discovering interesting information and knowledge in customer transactions.

As data mining techniques can be used to discover implicit information in very large databases, private or secure information may also be easily revealed by those techniques, such as credit card numbers, personal identification numbers, telephone numbers, and other confidential data. Besides, another important issue is that information shared among business collaborators may also be analyzed using data mining techniques to reveal sensitive knowledge that may then be leaked to competitors. Another risk is that a current collaborator may become a competitor, and that this latter may use the strategic knowledge obtained using data mining techniques to take better business decisions, and thus decrease the

* Corresponding author.
*E-mail addresses:* jerrylin@ieee.org (J.-W. Lin), qiankunliu@ikelab.net (Q. Liu), philfv@hitsz.edu.cn (P. Fournier-Viger), tphong@nuk.edu.tw (T.-P. Hong), miroslav.voznak@vsb.cz (M. Voznak), justin.zhan@unlv.edu (J. Zhan).

http://dx.doi.org/10.1016/j.engappai.2016.03.007
0952-1976/© 2016 Elsevier Ltd. All rights reserved.

business performance of the data provider as a result of increased competition. Because of these issues, privacy-preserving data mining (PPDM) has become a critical issue in recent years (Aggarwal et al., 2006; Dasseni et al., 2001; Evfimievski et al., 2002; Lindell and Pinkas, 2000; Verykios et al., 2004). The goal of PPDM is to sanitize a database, that is to hide and secure personal, confidential or sensitive information of participants, while still permitting data analysis. The most common way of hiding sensitive information in a collected database is to sanitize the database through deletion or addition operations. However, this approach may cause several side effects such as hiding non-sensitive patterns, or introducing new artificial patterns. It is a NP-hard (Aggarwal et al., 2006; Verykios et al., 2004) optimization problem to select an appropriate set of sanitization operations for hiding confidential information, while minimizing side effects.

Agrawal and Srikant first introduced PPDM (Aggarwal et al., 2006). Lindell and Pinkas (2000) addressed the issue of PPDM for decision tree learning with the ID3 algorithm. Clifton et al. (2003) presented a toolkit to address various problems in PPDM. To obtain a good balance between data privacy and data utility in PPDM, Pandya et al. (2014) proposed a multiplicative perturbation algorithm. Dwork et al. (2006) generalized previous work by considering both multi-attribute databases and vertically partitioned databases, and designed several algorithms for handling published noisy statistics. Several algorithms were also designed to hide sensitive frequent itemsets or sensitive association rules using custom sanitization procedures (Evfimievski et al., 2002; Hong et al., 2012; Lin et al., 2013; Wu et al., 2007).

Traditional PPDM algorithms have difficulty to cope with the challenge of finding an appropriate set of transactions/itemsets for sanitization that would minimize side effects especially when sensitive information overlaps with important but non sensitive information. Hiding and securing sensitive information may at the same time hide important information. Evolutionary computing is an efficient way of finding near optimal solutions to NP-hard problems. Genetic algorithms (GAs) (Goldberg, 1989; Holland, 1992) are a population-based approach that facilitates the search for good solutions by applying the principles of natural evolution. It has been extensively applied to handle problems having both discrete and continuous variables, nonlinear objectives, and constraint functions without gradient information. In the past, Lin et al. (2014, 2015a) proposed a GA-based algorithm to hide sensitive itemsets using a designed sanitization procedure. In this approach, choosing a set of transactions for deletion is done using a GA framework. It has been shown that GA-based approaches can provide better solutions to PPDM problems with lower side effects compared to traditional Greedy algorithms. Those algorithms still, however, require to manually set the number of transactions to be deleted. Besides, it is a non-trivial task to find appropriate values (rates) for parameters used by GAs such as mutation and crossover rates.

Particle swarm optimization (PSO) was invented by Kennedy and Eberhart (1995). It is inspired by bird flocking to find rich food sources. As GAs, PSO is a population-based search approach, designed to solve optimization problems. In PSO, each particle represents a solution and is evaluated by a predefined fitness function. The personal best (*pbest*) and global best (*gbest*) particles are used to update old particles and generate offsprings of the population, in the evolution process. Since the crossover and mutation operations in GAs are not used in PSO, it is easier to implement the PSO procedure for discovering near optimal solutions. Besides, particles in PSO can transmit information to other particles to speed up the evolution process. In this paper, a PSO-based PSO2DT algorithm is presented to find better sets of transactions to be deleted for hiding sensitive information. The key contributions of the designed algorithm are listed below.

1. In the past, few heuristic approaches have been proposed to sanitize databases for hiding sensitive information. Most of them utilize the GA framework. This is the first paper to address the problem of hiding sensitive itemsets using a PSO-based approach.
2. The designed PSO2DT algorithm is inspired by discrete PSO. It assigns particles and their velocities to set of transaction identifiers, representing transactions to be deleted for hiding sensitive itemsets. An advantage of the designed PSO2DT algorithm is that it has few parameters compared to previous approaches, and it still searches for near optimal solutions to the sanitization problem using a randomized evolutionary approach.
3. The pre-large concept is also adopted in the designed algorithm to avoid performing multiple database scans. This considerably speeds up the evaluation of particles in the evolution process.

The rest of this paper is organized as follows. Related work is reviewed in Section 2. Preliminaries and problem definition are mentioned in Section 3. The PSO2DT sanitization algorithm is presented in Section 4. An example illustrating the proposed algorithm is given in Section 5. Experimental results are reported in Section 6. A conclusion is drawn and future work is discussed in Section 7.

## 2. Related work

This section reviews related work about GAs, PSO and PPDM.

### 2.1. Genetic algorithm

In evolutionary computing, population-based approaches are widely used to find near optimal solutions to optimization problems. They are especially used for variations of NP-hard problems and related applications, where it is too expensive to find the best solution by evaluating all solutions. GAs are the most fundamental population-based approach. It has been developed in the early 1970s by Holland (1992). In GAs, a solution is called a chromosome, and it can be evaluated by a designed fitness function. Three operations named selection, crossover, and mutation are used by GAs and are described below.

1. *Crossover*: This operation swaps some bits among two chromosomes (individuals) to generate offsprings of the population. An offspring inherits attributes or characteristics of its two parent chromosomes.
2. *Mutation*: This operation randomly changes one or several bits of an offspring, which may produce variations of its parent characteristics. This operation is used to avoid being trapped in local optimal solutions, and is what allows the evolution process to find near optimal solutions.
3. *Selection*: This operation applies the fitness function to select the best offsprings as the surviving chromosomes. This operation ensures that characteristics of the best offsprings are likely transmitted to the next generation.

The main steps performed by a GA are the following. The first step is to define a type of chromosomes to represent possible solutions. Chromosomes are usually represented as bit strings. An initial population consisting of many chromosomes, also called individuals, is defined, and represents an initial set of possible solutions. The crossover, mutation, and selection operations are then applied to chromosomes to produce the next generation. Each chromosome is evaluated by the designed fitness function to assess the goodness of the chromosomes. This process is then repeated until a termination criterion is satisfied. Although GAs

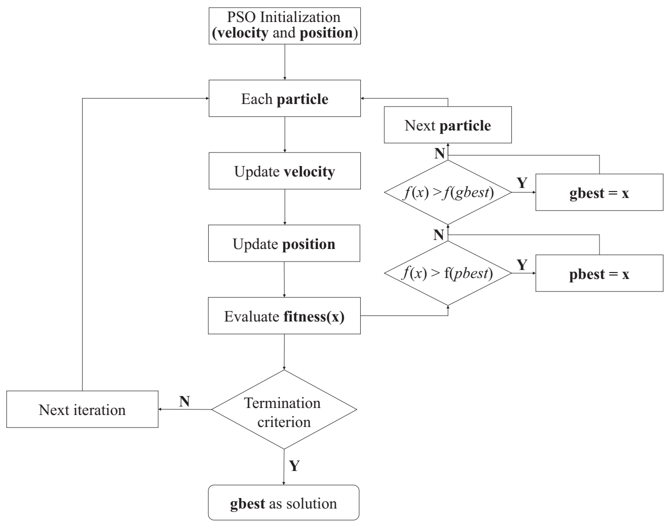

**Fig. 1.** Flowchart of traditional PSO.

generally perform well and have many applications, an important drawback of GAs is that it is a non-trivial task to set the appropriate crossover and mutation rates.

### 2.2. Particle swarm optimization

PSO is also a population-based approach, introduced in Kennedy and Eberhart (1995). PSO is inspired by the behavior of birds flocking to find better food sources. In PSO, particles are used to represent problem solutions, where each particle has a velocity representing a flying direction toward other solutions. The flowchart of traditional PSO is shown in Fig. 1. The PSO procedure first randomly initializes particles. Then, an iterative evolution process is performed. During each iteration, each particle is updated using its personal best value (*pbest*) and global best value (*gbest*) based on the designed fitness function. The *pbest* value is the personal best solution of a particle (according to the fitness function) so far, and the *gbest* value is the best solution among all *pbest* values in the population. Particles and their corresponding velocities are thus evaluated and updated using these two best values. The updating procedure of each particle is given below:

$$v_i(t+1) = w_1 \times v_i(t) + c_1 \times r_1 \times (pbest - x_i(t)) + c_2 \times r_2 \times (gbest - x_i(t)). \quad (1)$$

$$x_i(t+1) = x_i(t) + v_i(t+1). \quad (2)$$

In the above equations, $w_1$ is a factor balancing the importance of global search and local search. The notation $v_i$ denotes the velocity of the $i$-th particle in a population. $c_1$ and $c_2$ are constants respectively called the individual weight and social weight, which determines the importance of the personal best and global best solutions, and are usually both set to 2. Both $r_1$ and $r_2$ are random numbers generated by a uniform distribution in the range of [0, 1]. During an iteration, the velocity of each particle is first updated by Eq. (1). Then, each particle is updated toward a global solution by Eq. (2).

PSO was originally used to find solutions to continuous problems, and has been applied in numerous applications. Zuo et al. (2014) proposed a self-adaptive learning PSO-based framework for scheduling inter-cloud resources. Kuo et al. (2011) developed an approach to discover association rules in customer transaction databases. Bonam et al. (2014) addressed the issue of privacy preserving association rule mining by developing a PSO-based approach using data distortion. The IR value was proposed as a novel interesting measure to find meaningful association rules as an alternative to traditional measures such as the support and confidence. Real-world routing and scheduling is usually viewed as a discrete optimization problem. Thus, the traditional PSO approach cannot be directly applied to find a solution to this problem. To address this issue, Kennedy and Eberhart (1997) developed discrete PSO to find near optimal solutions to discrete optimization problems. Discrete PSO uses the traditional velocity-based updating approach of continuous PSO, but a *sigmoid* function is used to update the $j$-th vector of a particle, containing binary values (either 0 or 1). The *sigmoid* function is defined as

$$sig(v_{ij}(t+1)) = \frac{1}{1+e^{-v_{ij}(t+1)}} \quad (3)$$

where $v_{ij}(t+1)$ denotes the $j$-th vector of a velocity generated by Eq. (1). When updating a particle, a random number is generated in the [0, 1] interval for comparison. If the random number is less than $sig(v_{ij}(t+1))$, $x_{ij}(t+1) = 1$; otherwise, $x_{ij}(t+1) = 0$.

Sarath and Ravi (2013) then designed a PSO optimization approach to mine association rules. Lin et al. (2015b) developed a binary PSO-based algorithm to efficiently mine high utility itemsets. Zhi et al. (2004) developed a discrete PSO method to address the generalized TSP problem. Shen et al. (2014) also applied discrete PSO to the multicast routing problem in communication networks. Tian et al. (2013) proposed a scheme that adopts discrete PSO to address the assembly scheduling problem. Designing PSO-based approaches to mine patterns in databases is an active research area (Menhas et al., 2011; Pears and Koh, 2002).

### 2.3. Privacy-preserving data mining

In recent years, data mining has become a key technology for businesses since it can easily reveal relationships between products purchased by customers. This information can then be used by managers or retailers to implement efficient and useful sales strategies. However, as data mining techniques are designed to reveal implicit relationships and knowledge, they may also be used to reveal confidential or sensitive information. To avoid this problem, confidential or secure information need to be hidden before a database is publicly published or shared with collaborators. For this reason, PPDM has recently emerged as a key research area (Aggarwal et al., 2006; Dasseni et al., 2001; Evfimievski et al., 2002; Lindell and Pinkas, 2000; Verykios et al., 2004). Agrawal and Srikant (2000) presented a reconstruction procedure that accurately estimate the distribution of original data values, and build classifiers to compare the accuracy of a sanitized database with respect to the corresponding original database. Verykios et al. (2004) gave an overview and proposed a hierarchical classification of PPDM techniques. Hajian et al. (2014) proposed a generalization-based approach to both preserve privacy and prevent discrimination. This approach can be extended to different privacy models and has good scalability. Lindell and Pinkas (2000) developed an approach to hide privileged information from decision tree learning with the well-known ID3 algorithm, for sharing data among several parties. Evfimievski et al. (2002) developed several algorithms to randomize numerical and categorical data for PPDM. They also defined a new privacy measure to consider the issue of privacy from a different angle than the conventional cryptographic approach. Clifton et al. (2003) provided a toolkit and several techniques for specific applications of PPDM. Islam and Brankovic (2011) proposed a framework to protect the privacy of individuals using noise addition, which both consider numerical and categorical data. Atallah et al. (1999) presented a heuristic approach for data perturbation, which modify an original database by decreasing the support of sensitive rules below a given user-specified threshold. Oliveira and Zaane (2002) developed a heuristic framework with several sanitization algorithms to hide frequent itemsets. The designed algorithms rely on an item-restriction approach, thus avoiding noise addition and limiting the deletion of real data. Sweeney (2002) designed a generalized $k$-anonymity algorithm to protect and suppress sensitive attributes.

Contrarily to traditional hiding algorithms developed for PPDM, some sanitization algorithms aim at minimizing side effects of sanitization. This problem can be considered as a NP-hard optimization problem (Aggarwal et al., 2006; Verykios et al., 2004). Few bio-inspired algorithms were proposed to find solutions to the problem of hiding sensitive information in PPDM. Han and Ng (2007) developed a secure protocol for discovering a set of rules while ensuring the non disclosure of their own private data, using GAs. In this work, the goodness of each decision rule is evaluated by multiplying the true positive rate by the true negative rate. Lin et al. proposed the sGA2DT, pGA2DT (Lin et al., 2015a), and cpGA2DT (Lin et al., 2014) algorithms for hiding sensitive itemsets by removing transactions, using GAs. Each chromosome encodes a solution consisting of a set of transactions to be deleted. The goodness of a chromosome is evaluated using a designed fitness function, which considers three side effects of sanitization. The sGA2DT algorithm uses simple GAs to find an appropriate set of transactions to be deleted for hiding sensitive itemsets. To speed up the evolution process, the pGA2DT algorithm extends the sGA2DT algorithm by adopting the pre-large concept (Hong et al., 2001). This optimization consists of maintaining a buffer of pre-large itemsets during the evolution process, to avoid performing multiple database scans for each iteration. To use traditional GA-based algorithms, a user must specify the number of chromosomes in a population. The cpGA2DT algorithm adopts a compact GA (Harik et al., 1999) approach to only generate two individuals per population for competition, thus reducing memory usage during the evolution process. However, several parameters still need to be specified by the user such as chromosome size, mutation rate, and crossover rate. It is not a trivial task to find appropriate values for these parameters in real-life situations.

## 3. Preliminaries and problem definition

This section introduces preliminaries and then defines the problem of hiding sensitive frequent itemsets while minimizing side effects.

### 3.1. Preliminaries

Let $I = \{i_1, i_2, \ldots, i_r\}$ be a finite set of $r$ distinct items. A database $D$ is a set of transactions $D = \{T_1, T_2, \ldots, T_n\}$, where for each transaction $T_q \in D$, $T_q$ is a subset of $I$, and $T_q$ has a unique identifier $q$,

**Table 1**
An original database.

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

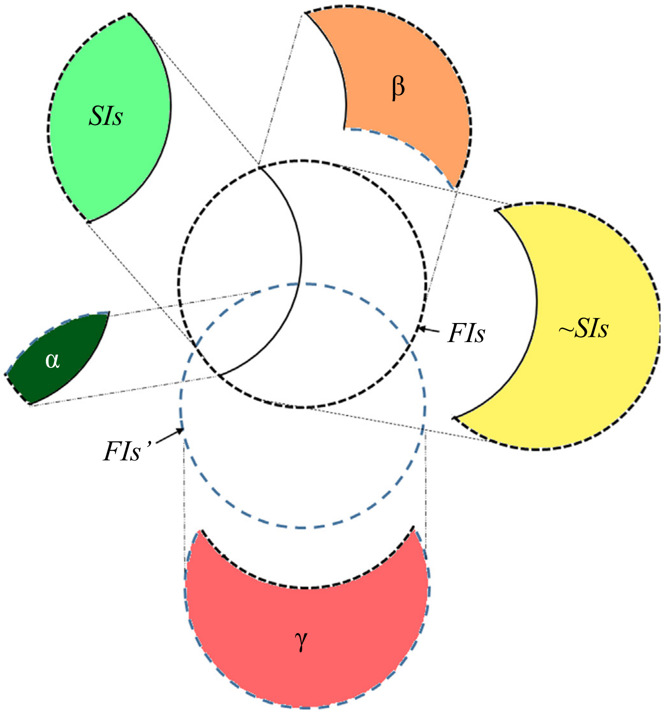

**Fig. 2.** Relationships of PPDM.

called its Transaction IDentifier (*TID*). A user-specific minimum support threshold $\delta$, is assumed to be manually set by users or experts (a percentage). In the following, the database shown in Table 1 will be used as running example. It contains 10 transactions where items are represented by letters.

**Definition 1.** An itemset $X$ is a set of items $(X \subseteq I)$. The support count of an itemset $X$ in a database $D$ is the number of transactions containing $X$. Let the minimum support count be the product of the minimum support threshold and the number of transactions in the database. The set of frequent itemsets is denoted as $FIs = \{f_1, f_2, \ldots, f_k\}$, and is defined as all itemsets having a support count that is no less than the minimum support count. Thus, an itemset $f_i$ is deemed a frequent itemset $(f_i \in FIs)$ if and only if:

$$sup(f_i) \geq |D| \times \delta. \tag{4}$$

For example, suppose that the minimum support threshold $\delta$ is set to 40%. The minimum support count is calculated as $(10 \times 0.4)(=4)$. From Table 1, the support count of the itemset $(bc)$ is $sup(bc)(=5)$, since it appears in five transactions. Since this value is greater than the minimum support count, $(bc)$ is a frequent itemset in this database.

**Table 2**
The projected database of $(bc)$ and $(ce)$.

| TID | Items |
|---|---|
| 2 | *b, c, e* |
| 3 | *a, b, c, e* |
| 5 | *b, c, e* |
| 6 | *b, c, e* |
| 7 | *b, c, d, e* |
| 10 | *c, e* |

**Definition 2.** The set of sensitive itemsets is denoted as $SIs = \{s_1, s_2, \ldots, s_p\}$, and defined as a subset of the set of *FIs* ($SIs \subseteq FIs$). This set can be specified by users or experts.

The goal of PPDM is to hide as much sensitive information as possible in a database so that confidential data cannot be discovered using data mining techniques. To hide sensitive itemsets, it is necessary to either remove transactions containing the sensitive itemsets or to remove itemsets containing confidential information from transactions. However, finding a set of transactions or itemsets to be deleted that minimize side effects is a NP-hard problem. Three important side effects are 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) (Wu et al., 2007). Definitions and explanations of these three side effects are given thereafter.

**Definition 3.** Let $D'$ be a sanitized database, such that $D'$ has been obtained by removing some transactions/itemsets from an original database $D$.

The relationship between the original database ($D$) and the sanitized database ($D'$) is depicted in Fig. 2. In Fig. 2, *FIs* denotes the set of frequent itemsets in $D$, *SIs* represents the set of sensitive itemsets to be hidden, $\sim SIs$ is the set of non-sensitive frequent itemsets in $D$, and $FIs'$ denotes the set of frequent itemsets in the sanitized database $D'$.

**Definition 4.** The side effect of F-T-H is denoted as $\alpha$, and defined as the number of sensitive itemsets appearing in the sanitized database $D'$, that is:

$$\alpha = |SIs \cap FIs'|. \tag{5}$$

**Definition 5.** The side effect of N-T-H is denoted as $\beta$, and defined as the number of non-sensitive itemsets that are hidden in the sanitized database $D'$, that is:

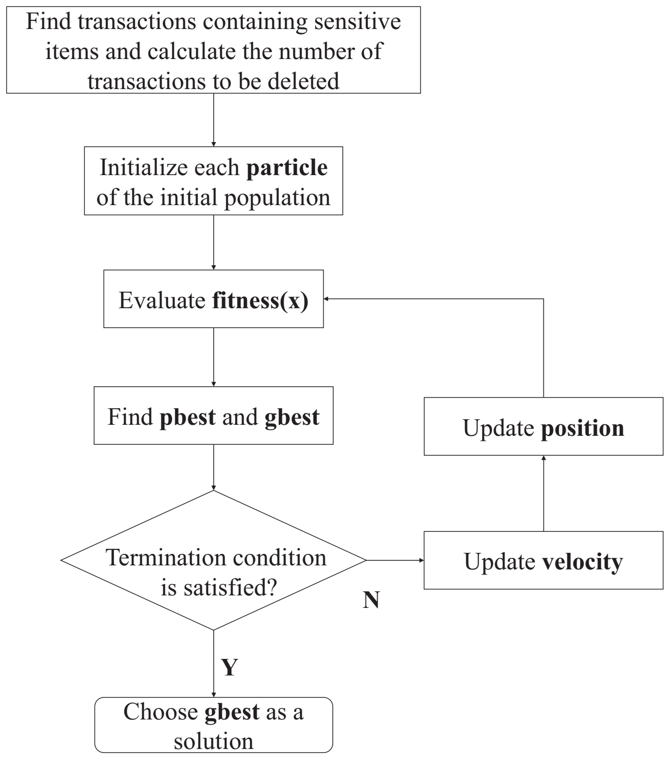
Find transactions containing sensitive items and calculate the number of transactions to be deleted
Initialize each particle of the initial population
Evaluate fitness(x)
Find pbest and gbest
Update position
Termination condition is satisfied?
Update velocity
N
Y
Choose gbest as a solution

**Fig. 3.** Flowchart of the designed PSO2DT algorithm.

**Table 3**
Discovered frequent itemsets.

| 1-itemset | Count | 2-itemset | Count | 3-itemset | Count |
|---|---|---|---|---|---|
| *a* | 5 | *ab* | 4 | *bce* | 5 |
| *b* | 8 | *bc* | 5 | | |
| *c* | 7 | *be* | 7 | | |
| *e* | 8 | *ce* | 6 | | |

**Table 4**
Pre-large itemsets.

| 2-itemset | Count | 3-itemset | Count |
|---|---|---|---|
| *ae* | 3 | *abe* | 3 |

**Table 5**
Initial particles and their fitness values.

| Particle | TIDs | Fitness |
|---|---|---|
| $p_1$ | [2, 5, 0, 7] | 1.1 |
| $p_2$ | [0, 3, 7, 0] | 1.0 |
| $p_3$ | [0, 2, 3, 3] | 1.0 |
| $p_4$ | [0, 5, 6, 0] | 0.9 |
| $p_5$ | [2, 3, 0, 10] | 1.6 |

$$\beta = |\sim SIs - FIs'| = |(FIs - SIs) - FIs'|. \tag{6}$$

**Definition 6.** The side effect of N-T-G is denoted as $\gamma$, and defined as the number of frequent itemsets in the sanitized database $D'$ that were infrequent in the original database $D$, that is:

$$\gamma = |FIs' - FIs|. \tag{7}$$

## 3.2. Problem statement

The problem of sanitizing a database $D$ for PPDM is to decrease the support count of e sensitive itemset $s_i \in SIs$ such that its support count becomes less than the minimum support count, that is:

**Table 6**
Updated velocities of the particles.

| Velocity | TIDs |
|---|---|
| $v_1$ | [6] |
| $v_2$ | [5, 6] |
| $v_3$ | [5, 6] |
| $v_4$ | [*null*] |
| $v_5$ | [5, 6] |

**Table 7**
Updated particles and their fitness values.

| Particle | TIDs | Fitness |
|---|---|---|
| $p_1$ | [6, 3, 0, 7] | 0.9 |
| $p_2$ | [5, 6, 0, 2] | 1.1 |
| $p_3$ | [5, 6, 2, 3] | 0.1 |
| $p_4$ | [2, 0, 5, 3] | 0.9 |
| $p_5$ | [5, 6, 2, 7] | 0.3 |

**Table 8**
Parameters of the datasets.

| | |
|---|---|
| #\|**D**\| | Total number of transactions |
| #\|**I**\| | Number of distinct items |
| **AvgLen** | Average transaction length |
| **MaxLen** | Maximal transaction length |
| **Type** | Dataset type |

**Table 9**
Characteristics of the datasets.

| **Dataset** | #\|**D**\| | #\|**I**\| | **AvgLen** | **MaxLen** | **Type** |
|---|---|---|---|---|---|
| Chess | 3196 | 74 | 37 | 37 | Dense |
| Mushroom | 8124 | 119 | 23 | 23 | Dense |
| Foodmart | 21,556 | 1559 | 4 | 11 | Sparse |
| T10I4D100K | 100,000 | 870 | 10.1 | 29 | Sparse |

$$sup(s_i) < \delta \times |D|. \tag{8}$$

The three aforementioned side effects are the standard measures used to assess the goodness of a sanitization approach. If the number of F-T-H is too high, it means that many sensitive patterns can still be discovered in the sanitized database. If the number of N-T-H is too high, it indicates that important decision making information may be missing in the sanitized database. If the number of N-T-G is too high, it means that a large amount of meaningless artificial information may have been introduced by the sanitization process. If the amount of sensitive information to be hidden is very large, there is a high possibility that important non-sensitive information may also be hidden by the sanitization process. Besides, note that when transactions are deleted in a database, the number of transactions changes, and thus also the minimum support count. Thus, numerous infrequent itemsets may 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 a NP-hard problem to find a solution to the problem of PPDM that minimize the three side effects. In this paper, we focus not only on hiding as much sensitive information as possible but also on minimizing the three side effects of sanitization.

## 4. Proposed PSO2DT sanitization algorithm

Because PPDM is NP-hard, heuristic approaches should be used to find nearly optimal solutions rather than attempting to find truly optimal solutions. In this paper, the concept of discrete PSO from evolutionary computing is adopted in the designed algorithm to efficiently find a set of transactions to be deleted that minimize the three side effects. An important strength of the proposed approach is that few parameters need to be set compared to GA-based approaches for PPDM, but the proposed approach still searches for solutions using a randomized evolutionary approach. The designed algorithm performs the following steps.

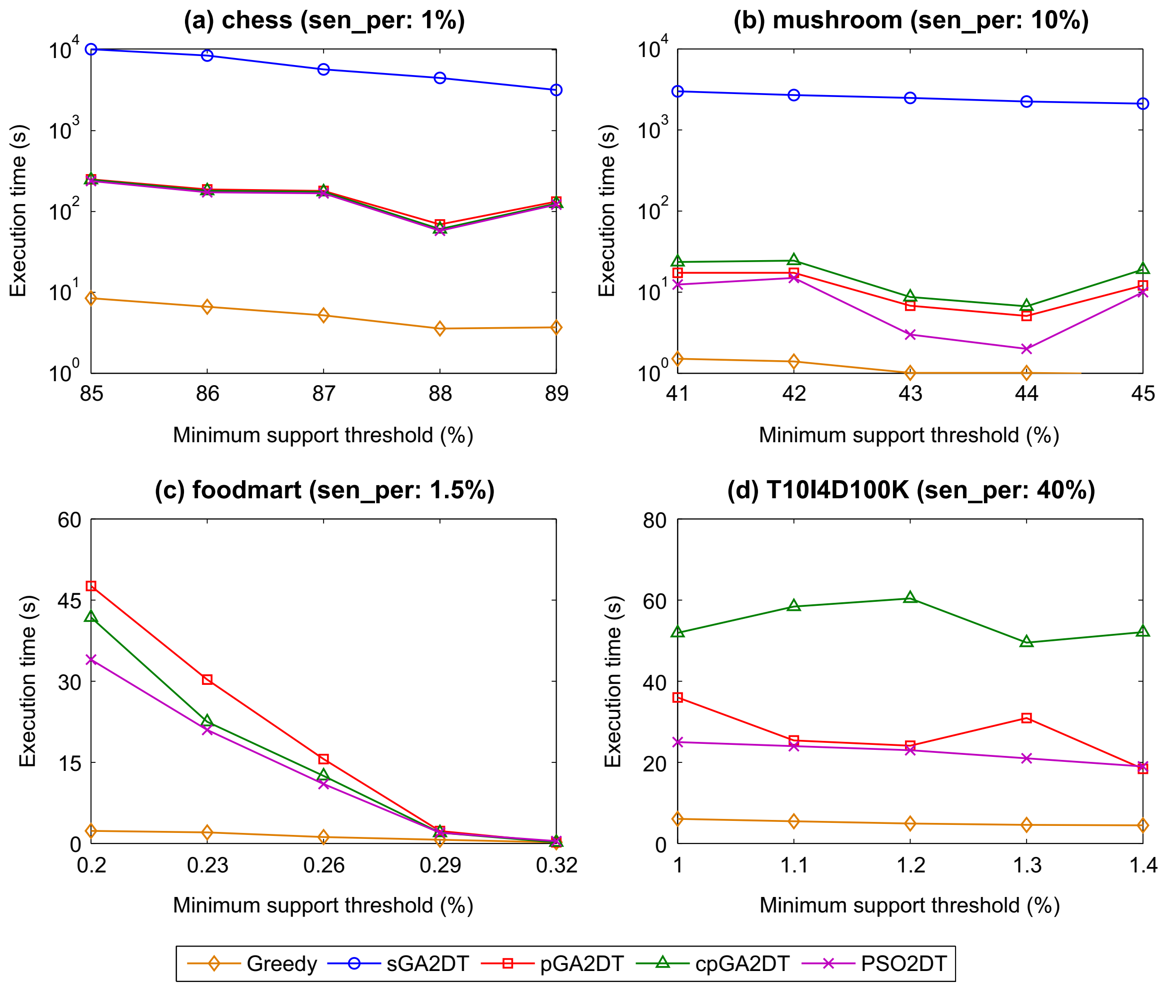

**Fig. 4.** Runtime w.r.t. various minimum support threshold values.

### 4.1. Initialization

To hide sensitive itemsets through transaction deletion, only transactions containing at least one sensitive itemset (an itemset in the set *SIs*) should be considered for deletion.

**Definition 7.** The first step is to extract a projected database $D^*$ from the original database *D*. The projected database $D^*$ is the set of transactions in *D* containing at least one sensitive itemset, that is:

$$D^* \leftarrow \{T_q | T_q \in D, \exists\, s_i \in HS, s_i \subseteq T_q\}. \quad (9)$$

For example, assume that two sensitive itemsets $(bc)$ and $(ce)$ need to be hidden. The corresponding projected database $D^*$ for $(bc)$ and $(ce)$ is shown in Table 2.

In the designed PSO2DT algorithm, each particle represents a solution, and is evaluated by a designed fitness function. Particle size is set to an appropriate number of transactions to be deleted, for hiding the sensitive itemsets.

**Definition 8.** The appropriate number of transactions to be deleted is denoted as *m*, and is defined as the difference between the largest support count among all sensitive itemsets in *SIs* and the minimum support count, that is:

$$m = \left\lceil \frac{Max\_sup(s_i) - \delta \times |D|}{1 - \delta} \right\rceil. \quad (10)$$

**Definition 9.** A particle $p_i$ is an *m* dimensional vector, where each dimension represents a transaction to be deleted $T_q \in D^*$, and stores its TID. Note that a dimension may optionally contain the value *null*, representing no transaction.

For example, since $sup(bc)(=5)$ and $sup(ce)(=6)$, the appropriate number of transactions to be deleted is calculated as: $m\left( = \left\lceil \frac{6 - 0.4 \times 10}{1 - 0.4} \right\rceil \right)( = 4)$. Thus, four transactions will be deleted in the projected database $D^*$ to hide the sensitive itemset $(ce)$. By deleting four transactions, the support count of $(ce)$ can be updated as $sup(ce)( = 6 - 4)( = 2)$, and the size of the sanitized database will become $(10 - 4)( = 6)$. It can be observed that $(ce)$ is successfully hidden when four transactions containing $(ce)$ are deleted from the original database since $sup(ce)( = 2 < 0.4 \times 6 = 2.4)$. Thus, the designed equation (10) is reasonable and acceptable. The size of each particle in the evolution process is hence set to 4. Each particle is randomly initialized. In this example, a particle could be initialized as: $p_i = [2, 5, 6, 7]$.

To evaluate the goodness of a particle, a flexible fitness function is proposed, which measures three commonly used side effects of sanitization. In this function, weights are used to indicate the relative importance of each side effect to the user, and can be set according to each user's preferences.

**Definition 10.** The proposed fitness function is the weighted sum of the three side effects, and is defined as:

$$fitness(p_i) = w_1 \times \alpha + w_2 \times \beta + w_3 \times \gamma \quad (11)$$

where $\alpha$ is the number of F-T-H itemsets; $\beta$ is the number of N-T-H itemsets; and $\gamma$ is the number of N-T-G itemsets; $w_1$, $w_2$ and $w_3$ are

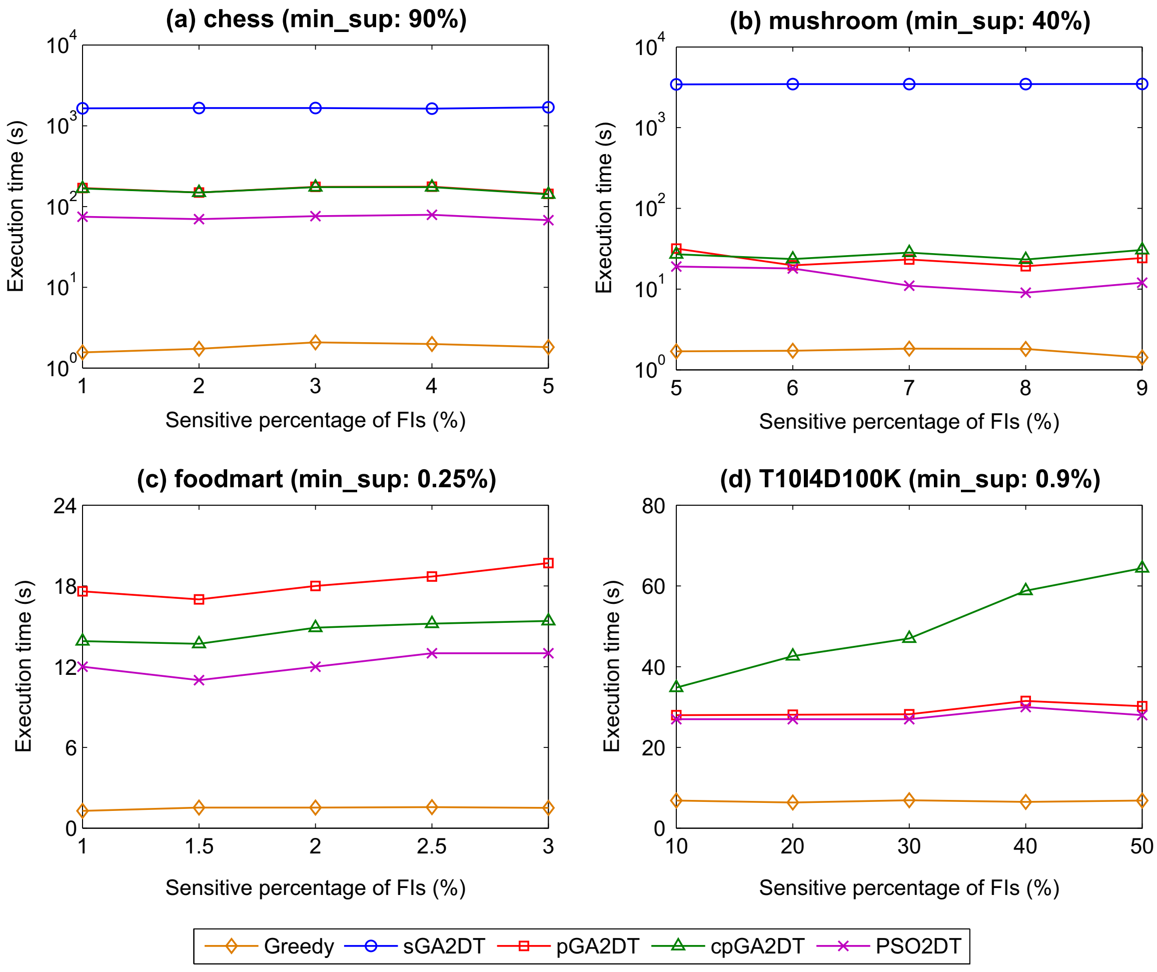

**Fig. 5.** Runtime w.r.t. various sensitive percentages of FIs.

weights indicating the relative importance of each side-effect, which can be adjusted by the user.

Thus, if the user wants to hide as much sensitive information as possible, $w_1$ should be set higher. If the user wants to ensure that important non-sensitive information is preserved for decision making, $w_2$ should be set higher. And if the user wants to reduce the introduction of artificial information, $w_3$ should be set higher. Note that the density of a database also influences the fitness of an evaluated particle. 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 sparse. 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 the designed PSO2DT algorithm, a particle and its velocity are sets of transactions to be deleted in $D^*$ for hiding sensitive information, and are represented as sets of TIDs. To update the velocity of a particle, the difference and union operations are applied on the particle with *pbest* and *gbest*, according to Eq. (12), where the difference operation calculates the difference between two particles, and the union operator is applied to obtain the union of two sets forming a new particle. By using this updating mechanism, the designed PSO2DT algorithm handles the problem of sanitization as a discrete problem. For this reason, several parameters used in traditional PSO such as $r_1$, $r_2$, $c_1$, and $c_2$ in Eq. (1) are unnecessary in the designed approach. To update a particle, TIDs contained in the elder particle or *null* are randomly selected to fill the dimensions of the updated particle. The result is then summed with the updated velocity of the particle, as shown in Eq. (13). This process increases the randomization of the evolution process, and thus its exploration ability, compared to parameters used in traditional PSO. The full updating process of particles and their velocities in the designed PSO2DT algorithm is shown below:

$$v_i(t+1) = (pbest - x_i(t)) \cup (gbest - x_i(t)). \tag{12}$$

$$x_i(t+1) = rand(x_i(t), \textbf{null}) + v_i(t+1). \tag{13}$$

For example, consider that the particle size is 4, that the updating process is applied to a particle $x(t) = [1, 4, 7, 5]$, that $pbest = [2, 1, 0, 5]$, and $gbest = [0, 3, 0, 4]$. In this example, 0 denotes the *null* value. According to Eq. (12), the velocity of the updated particle is calculated as $v(t+1) = \{[2, 1, 0, 5] - [1, 4, 7, 5]\} \cup \{[0, 3, 0, 4] - [1, 4, 7, 5]\} = [2, 0] \cup [0, 3] = [0, 2, 3]$. Since there is an empty dimension in the updated particle, it is then filled by randomly selecting a value from its elder particle or the *null* value [=0]. Suppose that $rand([1, 4, 7, 5], 0) = [1]$. The updated particle is thus [0, 2, 3, 1].

## 4.2. Proposed PSO2DT algorithm

The designed PSO2DT algorithm is proposed for efficiently hiding sensitive itemsets by transaction deletion. The pseudocode is given in Algorithm 1.

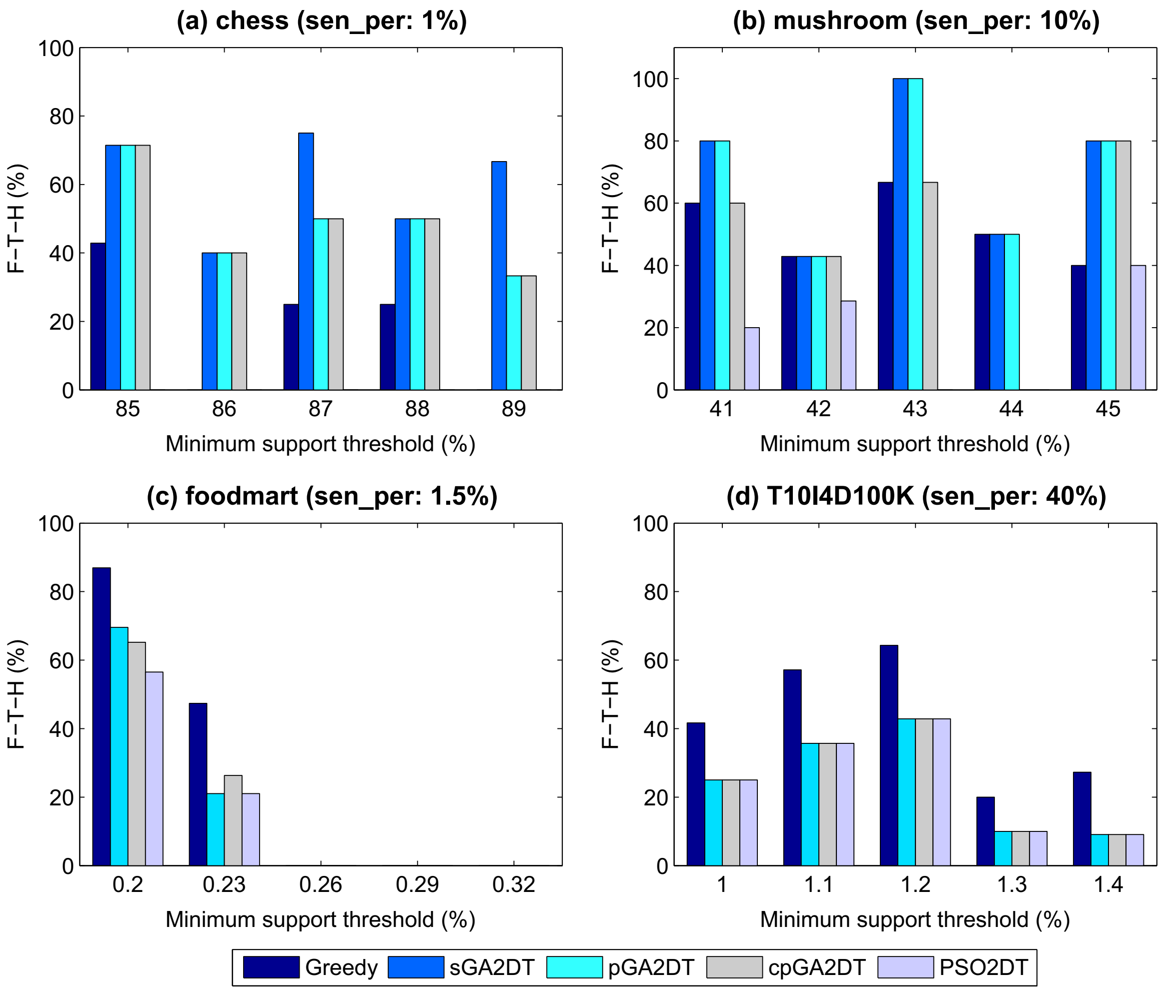

**Fig. 6.** F-T-H w.r.t. various minimum support threshold values.

The designed PSO2DT algorithm first calculates particle size as the difference between the support count of the most frequent sensitive itemset and the minimum support count, by applying Eq. (10) (Line 1). Then, $M$ particles are created. as initial population for the iterative evolution process.

**Algorithm 1.** Proposed PSO2DT algorithm.

```
   Input: $D^*$, the set of projected transactions; $SIs$, a set of sensitive itemsets to be
          hidden; $FIs$, the set of frequent itemsets in $D$; $\delta$, the minimum support
          threshold; and $M$, the number of particles to be used for each iteration
   Output: $D'$, a sanitized database.
 1 calculate the size of a particle as $m$;                        // by equation (10)
 2 for $i \leftarrow 1, M$ do
 3     for $i \leftarrow 1, m$ do
 4         $p_i(t) \leftarrow p_i(t) \cup rand(D^*)$;
 5 while termination criterion is not met do
 6     for $i \leftarrow 1, M$ do
 7         if $fitness(p_i(t)) \leq pbest_i(t)$ then
 8             $pbest_i(t) \leftarrow p_i(t)$;
 9         if $pbest_i(t) \leq gbest$ then
10             $gbest \leftarrow pbest_i(t)$;
11     for $i \leftarrow 1, M$ do
12         update the $v_i(t+1)$ velocities of $M$ particles;       // by equation (12)
13         update the $p_i(t+1)$ for $M$ particles;                 // by equation (13)
14     set $t \leftarrow t+1$;
15 delete transactions in $gbest$ from $D$;
16 return the sanitized database $D'$;
```

Each of these particles contains $m$ distinct TIDs randomly selected from $D^*$ (Lines 2 and 3). The iterative process then starts and is repeated until a predefined termination criterion is satisfied. During each iteration, the fitness value of each particle is evaluated using the fitness function. Then, the personal best $pbest_i$ of each particle is calculated as well as the global best *gbest* from the $M$ *pbest* values (Lines 6–10). After that, each particle is updated according to Eqs. (12) and (13) (Lines 11–14). This iterative process is then repeated until the termination criterion is met (Lines 5–14). After that, transactions in *gbest* are selected as the transactions to be deleted for sanitization (Line 15). Finally, the resulting sanitized database is returned (Line 16). The flowchart of the designed PSO2DT algorithm is shown in Fig. 3.

### 4.3. An improved strategy

The evolution process of the designed PSO2DT algorithm performs multiple database scans to evaluate side effects when transactions are deleted, especially to evaluate the N-T-G side-effect. To speed up the calculation of this side effect, and thus also the evolution process, the pre-large concept (Hong et al., 2001) is adopted in the designed algorithm. According to the pre-large concept, some infrequent itemsets having a high support may become frequent when transactions are deleted. These almost frequent itemsets are kept in a buffer for later evaluation, and are called pre-large itemsets. Using this buffer, all N-T-G itemsets can be identified from pre-large itemsets without scanning the database, as it will be explained. To determine which itemsets should be added to the buffer, a lower support threshold is defined. In previous work, the pre-large concept has been mainly used for incremental data mining to keep track of frequent itemsets when a database is frequently updated using insertion, deletion, or modification operations. The pre-large concept for transaction deletion is presented thereafter.

**Definition 11.** Let there be two support thresholds $S_u$ and $S_l$. It is unnecessary to scan the original database for detecting N-T-G itemsets if the number of deleted transactions is less than the safety bound $(f)$, defined as:

$$f = \frac{(S_u - S_l) \times |D|}{S_u}. \tag{14}$$

Generally, the upper support threshold in the pre-large concept is defined as the minimum support threshold. Recall that the designed PSO2DT algorithm selects transactions to be deleted from the database to hide sensitive itemsets. The size of each particle is set to the difference between the support count of the most frequent sensitive itemset and the minimum support threshold. In PSO2DT, this value is used as the safety bound $(f)$ to avoid performing multiple database scans. Thus, the pre-large concept can be modified to obtain the lower support threshold as follows:

$$f(=m) = \frac{(S_u - S_l) \times |D|}{S_u} \Rightarrow S_l = \frac{S_u \times (|D| - f(=m))}{|D|}. \tag{15}$$

For example in Table 1, the upper support threshold $S_u$ is 40% and the number of transactions to be deleted is 4. The lower support threshold can thus be calculated as: $S_l\left(=\frac{40\% \times (10-4)}{10}\right)(=24)\%$. During the initial process for discovering *FIs*, each itemset having a support between $S_l$ and $S_u$ is said to be a pre-large itemset, and is

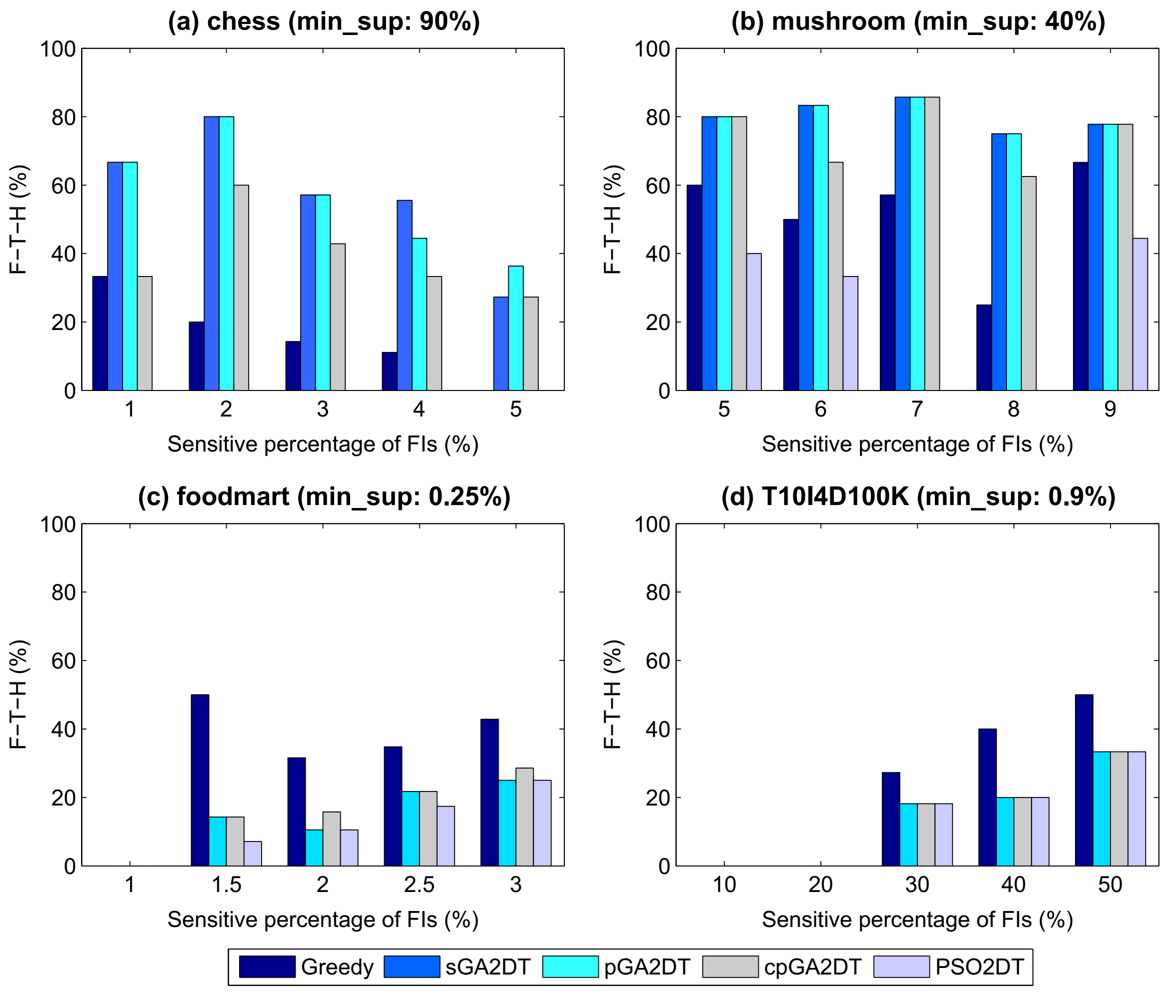

**Fig. 7.** F-T-H w.r.t. various sensitive percentages of FIs.

hence inserted into the buffer. For example, the support count of itemset $(ae)$ in Table 1 is 3. Thus, this itemset is a pre-large itemset. Thanks to the pre-large concept, the algorithm can avoid scanning the database multiple times, and this thus speed up the evolution process.

## 5. An illustrating example

In this section, an example using the database shown in Table 1 is given to illustrate the proposed PSO2DT algorithm, step by step. In this example, the minimum support threshold is set to 40%. The discovered frequent itemsets $(FIs)$ are shown in Table 3.

Consider that itemsets $(bc)$ and $(ce)$ are the sensitive itemsets that need to be hidden for the purpose of PPDM. First, the number of transactions to be deleted to completely hide sensitive itemsets is calculated as : $\left\lceil \frac{6-0.4\times 10}{1-0.4} \right\rceil$ $(=4)$. Thus, the lower support threshold of the pre-large concept can be calculated as: $\frac{40\% \times (10-4)}{10}$ $(=24\%)$. The original database is then scanned to find the pre-large itemsets, which will be used to avoid performing multiple database scans during the evolution process, especially to calculate the number of N-T-G itemsets. The pre-large itemsets are shown in Table 4.

Then, transactions containing at least one sensitive itemset are identified to create the projected database. The projected database contains all transactions that will be considered for deletion by the evolution process. In the running example, these transactions have the identifiers $\{2, 3, 5, 6, 7, 10\}$, and are shown in Table 2. Then, particles are initialized in the initial population by randomly selecting values from the set of transactions to be deleted and the *null* value. Assume that the weights for $\alpha$, $\beta$ and $\gamma$ in the fitness function are respectively set to 0.8, 0.1, and 0.1, which can be adjusted according to the user's preferences. The number of particles in a population is set to 5, and the number of iterations is set to 10. The evolution process will utilize pre-large itemsets to avoid performing multiple database scans, especially for evaluating the N-T-G side effect. Consider particle $p_1$ in Table 5 as an example to illustrate the evolution process. The particle $p_1$ represents the solution of deleting transactions [2, 5, 7] in the original database. In this example, the support ratios of $(bc)$ and $(ce)$ are respectively calculated as $(2/7 = 28.5\% < 40\%)$ and $(3/7 = 42.8\% > 40\%)$. Thus, according to this solution, $(bc)$ is completely hidden but $(ce)$ is a F-T-H. Since the frequent itemsets have been already discovered, it is easy to maintain the N-T-H. In this example, the support ratios of $\{(b), (c), (e), (bc), (be), (ce), (bce)\}$ are shown in Table 3 and are respectively updated to consider the deletion of transactions. Only $(bce)$ is a N-T-H since its support ratio after deletion is calculated as: $(2/7 = 28.5\% < 40\%)$. The side effect of N-T-H can also be easily maintained using the pre-large itemsets shown in Table 4. In this example, the support ratios of $(ae)$ and $(abe)$ increase and are both updated as: $(3/7 = 42.8\% > 40\%)$. Thus, $(ae)$ and $(abe)$ are N-T-G. In this example, the fitness value of particle $p_1$ is calculated as $fitness(p_1) = (0.8 \times 1 + 0.1 \times 1 + 0.1 \times 2) = 1.1$. The other particles are processed in the same way and the initially generated particles and their fitness values are shown in Table 5.

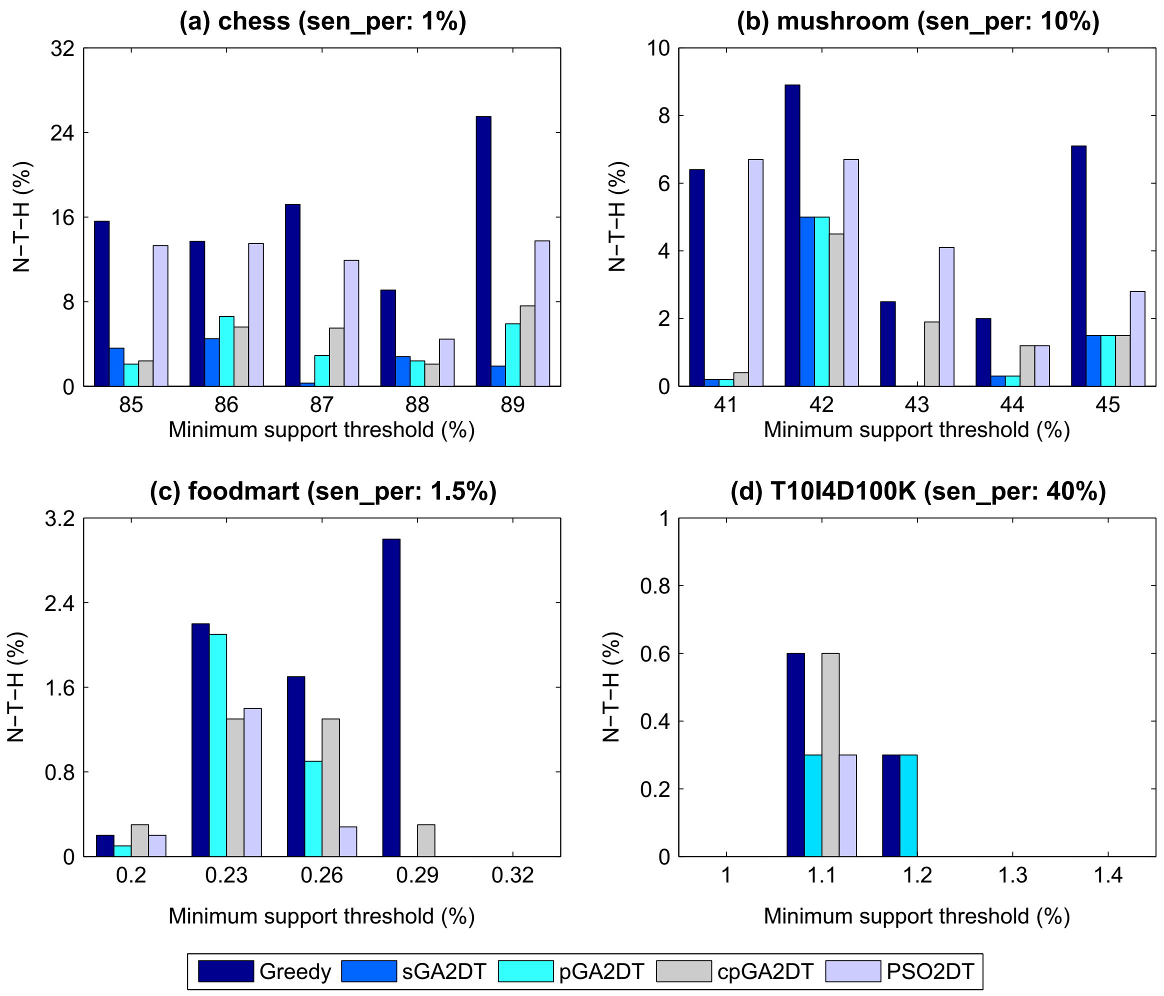

**Fig. 8.** N-T-H for various minimum support threshold values.

From the results of Table 5, it can be observed that $p_4$ has the smallest fitness value. $p_4$ is thus updated using the global best (*gbest*) of this iteration and its personal best (*pbest*). According to Eq. (12), the set difference between *pbest* and each particle, and the difference between the *gbest* and each particle, are respectively calculated. Then, the union of the results of the two set differences is calculated for each particle, and is used as the velocity of the particle for the next iteration. Consider $p_1$ in Table 5 as example to illustrate these steps. In this example, the *pbest* of $p_1$ is itself, and the *gbest* is $p_4$. The set difference between $p_1$ and *pbest* is calculated as: $[2, 5, 0, 7] - [2, 5, 0, 7] = [$***null***$]$. The set difference between $p_1$ and *gbest* is calculated as $[0, 5, 6, 0] - [2, 5, 0, 7] = [6]$. The union of the two set differences is $[$***null***$] \cup [6] = [6]$. The velocity $v_1$ of $p_1$ for the next iteration is thus calculated as [6]. The remaining velocities for the other particles are calculated in the same way. The updated velocities for all particles are shown in Table 6.

Based on Eq. (13), the TIDs from the elder particle and the *null* value are randomly selected to preserve the exploration ability of the particle. Consider $v_1$ in Table 6 as example to illustrate this process. In this example, the size of $v_1$ is 1 and the size of the particle is 4. Three transactions including the *null* value are randomly selected from the elder particle $p_1$. The result is [6, 3, 0, 7]. This process is repeated for every other particles in the same way. The particles obtained after applying this updating mechanism are shown in Table 7.

In Table 7, it can be observed that $p_3$ has the smallest fitness value. This latter is less than the current *gbest*. Thus, $p_3$ is set as the new *gbest*. The *pbest* of each particle is also updated during this iteration. The above evolution process is then repeated until the termination condition is satisfied. Note that the three side effects of the processed particle can be easily calculated without performing multiple database scans (especially the side effect of N-T-G using the discovered pre-large itemsets). The final *gbest* is thus set as the solution and the transactions corresponding to the TIDs in the final *gbest* are respectively deleted in the original database to hide the sensitive itemsets. In this example, transactions $\{2, 3, 5, 6\}$ are selected for deletion to hide the sensitive itemsets $(bc)$ and $(ce)$. The side effects F-T-H, N-T-H, and N-T-G are respectively 0, 1, and 0. The fitness value is thus calculated as $(0.8 \times 0 + 0.1 \times 1 + 0.1 \times 0)(=0.1)$.

## 6. Experimental results

Substantial experiments were conducted to compare the effectiveness and efficiency of the proposed PSO2DT algorithm with the state-of-the-art GA-based sGA2DT, pGA2DT, cpGA2DT approaches (Lin et al., 2015a, 2014), and the non-evolutionary Greedy sanitization algorithm (Lin et al., 2013). The proposed PSO2DT algorithm adopts the pre-large concept to avoid performing multiple databases scans to evaluate side-effects during the evolution process. The algorithms in the experiments were implemented in Java. Experiments were carried on a PC equipped with an Intel Core 2 i3-4160 processor and 4 GB of RAM, running under the 64-bit Microsoft Windows 7 operating system. Three real-world datasets called chess (Frequent Itemset Mining Dataset

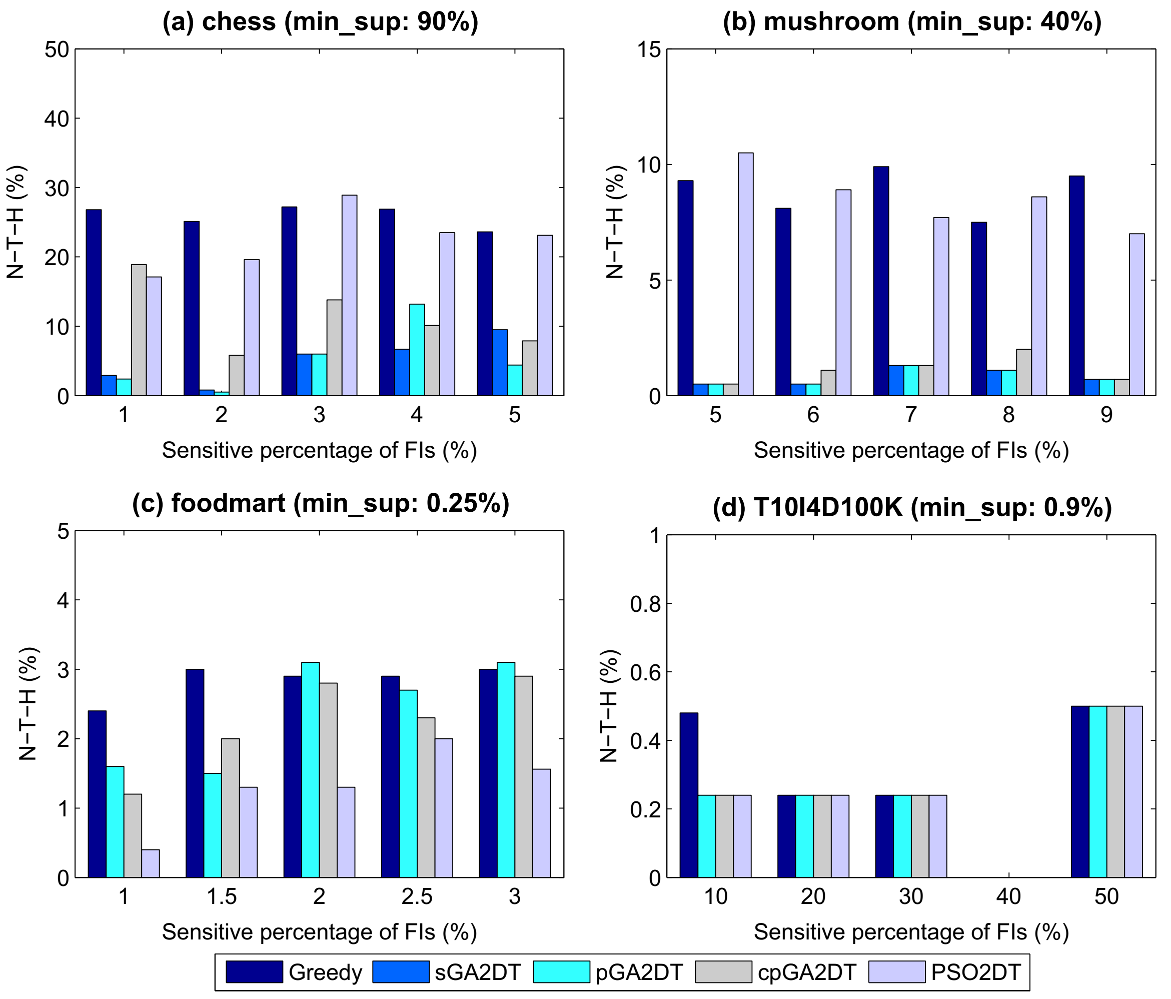

**Fig. 9.** N-T-H w.r.t. various sensitive percentages of FIs.

Repository, 2012), mushroom (Frequent Itemset Mining Dataset Repository, 2012), and foodmart (Microsoft) were used in the experiments. A synthetic dataset named T10I4D100K has also been generated using the IBM database generator (Agrawal and Srikant, 1994a). Parameters and characteristics of the datasets used in the experiments are respectively shown in Tables 8 and 9.

In these experiments, the number of iterations was set to 100, and 10 particles were used in each population. 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 dataset's characteristics.

## 6.1. Runtime

A first experiment was conducted to compare the runtime of the designed algorithm with the Greedy, sGA2DT, pGA2DT, and cpGA2DT algorithms, when the minimum support threshold and sensitive percentage are varied. Results are shown in Figs. 4 and 5. 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 Figs. 4 and 5. In fact, evolutionary approaches should not be compared with evolutionary approaches in term of runtime, as these two types of approaches are very different. It is also meaningless to compare results of non-evolutionary algorithm with evolutionary algorithms using a two-way ANOVA analysis. Thus, in the following, we only compare results of GA-based algorithms with the designed algorithm using a two-way ANOVA analysis.

In Fig. 4(c) and (d), the runtimes of the sGA2DT algorithm are not shown since they exceed 1000 s for a single iteration. It can be observed from these results that the proposed algorithm outperforms GA-based algorithms on the four datasets. Another observation is that as the minimum support threshold is increased, the number of FIs decreases, and less sensitive itemsets needs to be hidden. Thus, runtimes of the compared algorithms decrease when the minimum support threshold is increased. Based on a two-way ANOVA analysis, the runtime of the sGA2DT algorithm is significantly different from the runtimes of the other algorithms ($F = 24.66$, $p < 0.001$, shown in Fig. 4(a); $F = 252.395$, $p < 0.001$, shown in Fig. 4(b)). For the foodmart dataset shown in Fig. 4(c), there is no significant difference between the runtimes of all compared algorithms ($F = 4.040$, $p = 0.061 > 0.05$). For the T10I4D100K dataset shown in Fig. 4(d), the runtime of cpGA2DT is significantly different from the other two algorithms

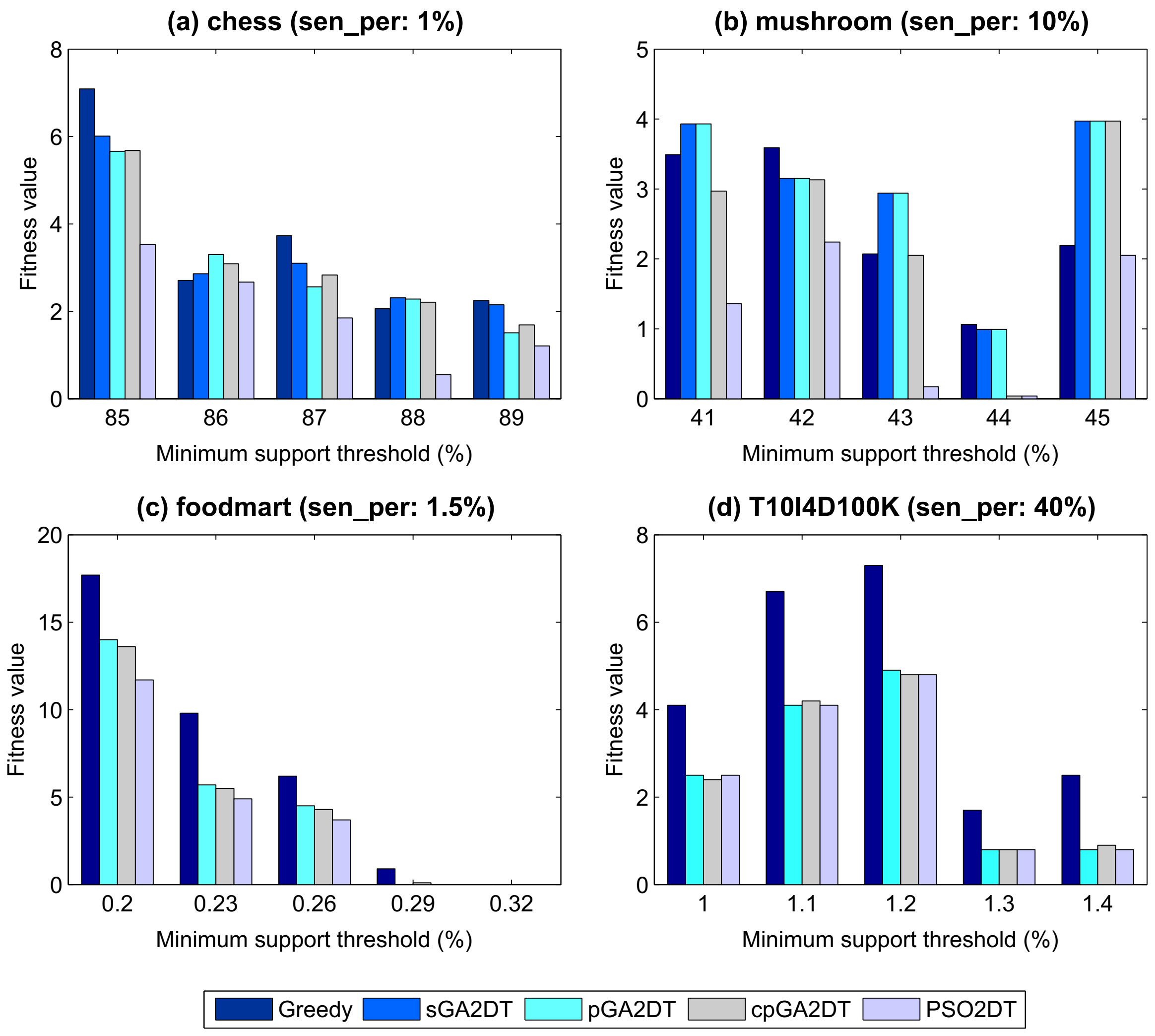

**Fig. 10.** Fitness values obtained for various minimum support threshold values.

($F = 65.295$, $p < 0.001$). In Fig. 4, it can also be found that the minimum support threshold has a small influence on the runtimes of the compared algorithms. Overall, the proposed PSO2DT algorithm always outperforms the other GA-based algorithms, i.e. it can find a solution more quickly than the other algorithms. Fig. 5 shows results obtained by the algorithms when the sensitive percentage of FIs is varied.

The runtimes of the sGA2DT algorithm are not shown in Fig. 5 (c) and (d) since they again exceed 1000 s for a single iteration. In Fig. 5, it can be observed that the proposed PSO2DT algorithm still outperforms GA-based algorithms when the sensitive percentage of FIs is varied, on the four datasets. The proposed PSO2DT algorithm is up to three times faster than the compared pGA2DT and cpGA2DT algorithms in Fig. 5(a) and (b). Besides, according to a two-way ANOVA analysis, there is a significant difference between the runtimes of the designed algorithm and the other algorithms ($F = 10146.414$, $p < 0.001$, shown in Fig. 5(a)). The difference between the sGA2DT algorithm and the other compared algorithms is also significant ($F = 229189.397$, $p < 0.001$). For the foodmart and T10I4D100K datasets shown in Fig. 5(c) and (d), the runtime of the proposed PSO2DT algorithm is up to 20% or 30% less than the other approaches. Besides, the runtime of the designed PSO2DT algorithm is significantly different from the cpGA2DT and pGA2DT algorithms ($F = 493.437$, $p < 0.001$, shown in Fig. 5(c)). There is no significant difference between runtimes of the PSO2DT and pGA2DT algorithms but there is one for the cpGA2DT algorithm ($F = 18.450$, $p = 0.001 < 0.05$). Overall, the proposed PSO2DT algorithm outperforms the state-of-the-art GA-based algorithms.

## 6.2. Side effects evaluation

The standard way of comparing sanitization algorithms in PPDM is to compare the number of occurrences of the three side effects that each algorithm generates. In this section, the F-T-H and N-T-H side effects are measures to compare the efficiency of the designed algorithm with the other algorithm. Since the N-T-G side effect is quite rare for all compared algorithms, it is unnecessary to compare the performance of the algorithms w.r.t this side-effect.

### 6.2.1. Fail To be hidden, F-T-H

The F-T-H side-effect has been measured to compare how much of the sensitive information has not been successfully hidden by each sanitization algorithm. Recall that this side effect is defined as:

$$\text{F} - \text{T} - \text{H} = \frac{|SIs^{*}|}{|SIs|}, \tag{16}$$

where $|SIs|$ is the number of sensitive itemsets in the original database and $|SIs^{*}|$ is the number of sensitive itemsets still appearing in the sanitized database. Results for various minimum support threshold values are shown in Fig. 6.

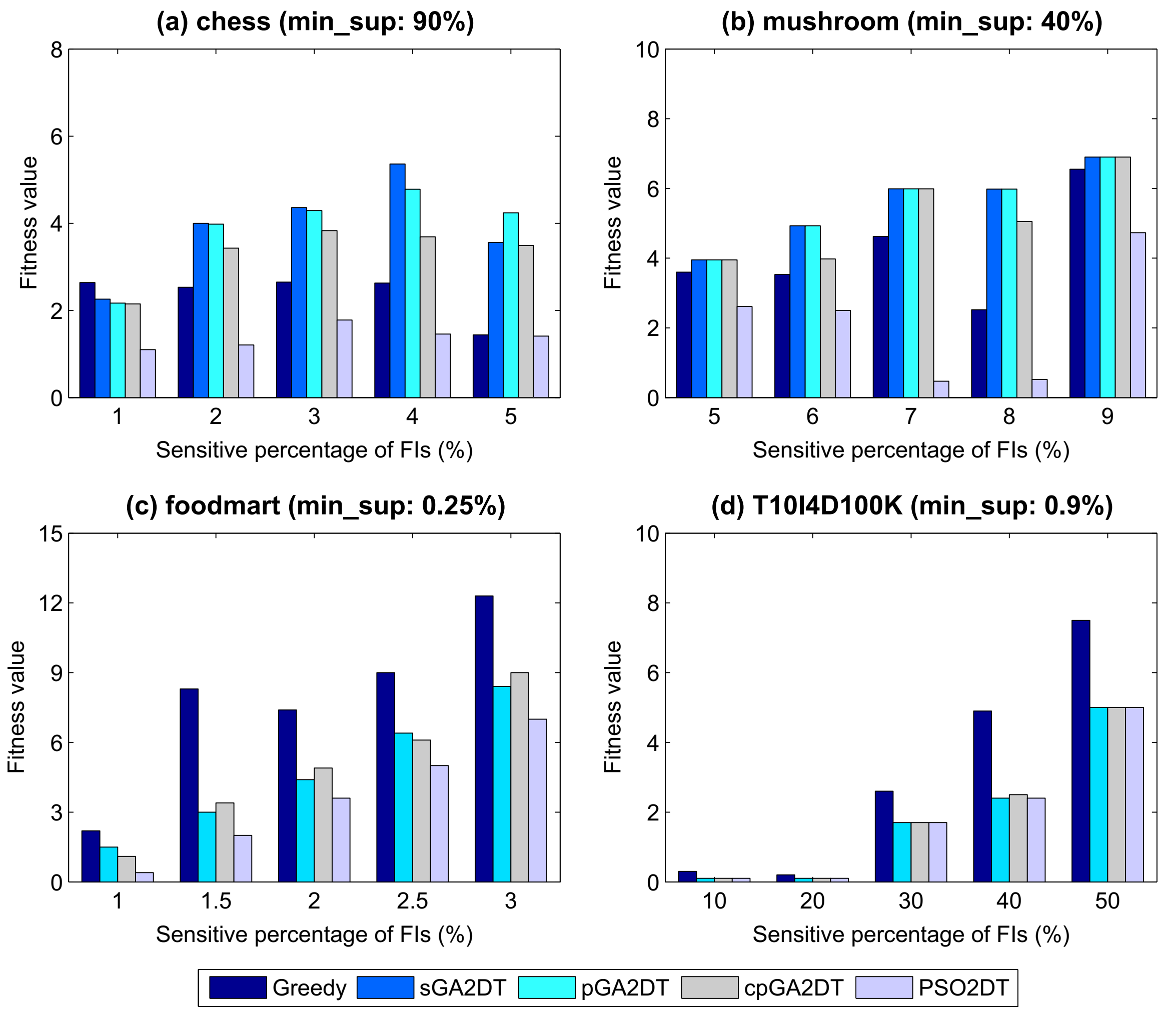

**Fig. 11.** Fitness values for various sensitive percentages of FIs.

In Fig. 6, it can be observed that the proposed PSO2DT algorithm achieves better results in terms of F-T-H compared to GA-based algorithms and the Greedy algorithm. For example, the proposed PSO2DT algorithm completely hides all sensitive itemsets for the chess dataset as shown in Fig. 6(a), and also successfully hides all sensitive itemsets when the minimum support threshold is set to 43% and 44% on the mushroom dataset, as shown in Fig. 6(b). The proposed algorithm has a clear advantage over the Greedy algorithm for all tested minimum threshold values and all databases, as shown in Fig. 6(a)–(d). On the two dense chess and mushroom datasets, results for Greedy are generally better than GA-based algorithms in terms of F-T-H, while on the two sparse foodmart and T10I4D100K datasets, results for Greedy are the worst. Based on a two way ANOVA analysis, results obtained by the PSO2DT algorithm are significantly different from the other algorithms except for the Greedy algorithm ($F = 15.130$, $p < 0.001$), as shown in Fig. 6(a). In Fig. 6(b), results for the proposed PSO2DT algorithm are significantly different from the other algorithms ($F = 10.030$, $p < 0.05$). For sparse datasets such as foodmart and T10I4D100K, results obtained by the proposed PSO2DT algorithm are similar to the compared algorithms, as shown in Fig. 6(d), but PSO2DT still outperforms the pGA2DT and cpGA2DT algorithms in Fig. 6(c) when the minimum support threshold is set to 0.2%. For sparse datasets, there is no significant difference between the compared algorithms ($F = 2.418$, $p = 0.117 > 0.05$, shown in Fig. 6(c)). However, results achieved by the proposed PSO2DT algorithm are significantly different from the Greedy algorithm ($F = 34.571$, $p < 0.001$, shown in Fig. 6(d)). Results obtained in terms of F-T-H, when varying the sensitive percentage of FIs are shown in Fig. 7.

For the chess dataset (results shown in Fig. 7(a)), the proposed PSO2DT algorithm successfully hides sensitive itemsets while GA-based algorithms and the Greedy algorithm do not. For the mushroom dataset, the proposed PSO2DT algorithm also achieves better results than the other algorithms, as it can be observed in Fig. 7(b). The reason is that the sets of transactions to be deleted for hiding each sensitive itemset highly overlap. Thus, when the proposed PSO2DT algorithm deletes a transaction, several sensitive itemsets may be hidden at the same time. According to a two-way ANOVA analysis, results for the proposed PSO2DT algorithm are significantly different from the other algorithms, except for the Greedy algorithm ($F = 35.221$, $p < 0.001$), as shown in Fig. 7(a). For the mushroom dataset, results for the PSO2DT algorithm are significantly different from the other algorithms ($F = 18.469$, $p < 0.001$), as shown in Fig. 7(b). For sparse datasets such as foodmart and T10I4D100K, results achieved by the proposed algorithm are nearly identical to those of GA-based algorithms. These results are reasonable since sensitive itemsets in sparse datasets often do not appear in the same transactions, and as a result the proposed PSO2DT algorithm cannot completely hide sensitive itemsets while considering the three side effects. But the proposed algorithm still outperforms the GA-based pGA2DT and cpGA2DT algorithms, as shown in Fig. 7(c). Moreover, results of the PSO2DT algorithm are significantly different from the Greedy

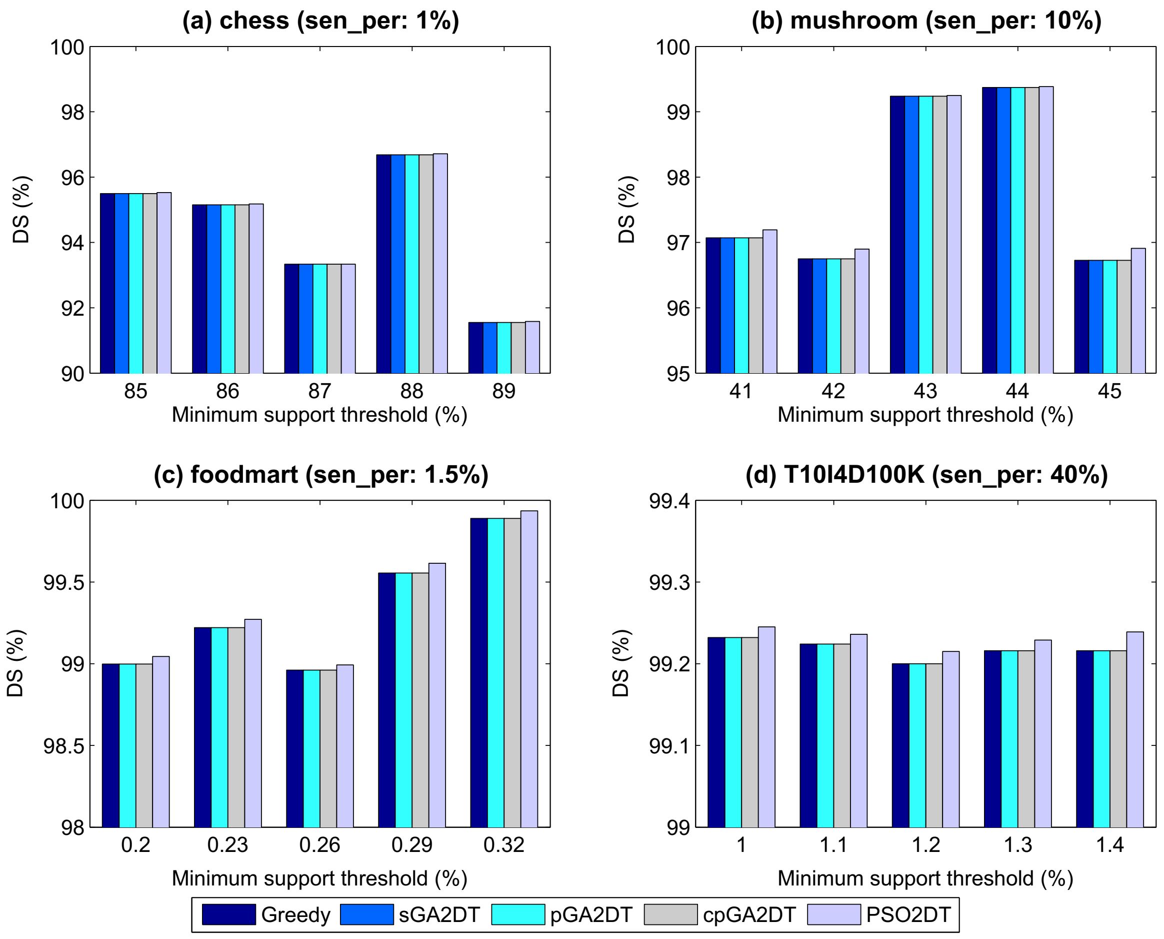

**Fig. 12.** DS w.r.t. various minimum support threshold values.

algorithm ($F = 13.029$, $p < 0.001$, shown in Fig. 7(c)), but not from the other compared algorithms ($F = 4.261$, $p = 0.029 > 0.05$, shown in Fig. 7(d)).

### 6.2.2. Not To be hidden, N-T-H

The side effect of N-T-H is used to evaluate how many non-sensitive frequent itemsets are hidden by the sanitization process, that is:

$$\mathrm{N-T-H} = \frac{|FIs - SIs - FIs^{*}|}{|FIs - SIs|}, \tag{17}$$

where *FIs* is the set of frequent itemsets discovered in the original database, and *SIs* is the set of sensitive itemsets. Therefore the term $|FIs - SIs|$ is the number of non-sensitive frequent itemsets in the original database. The notation $FIs^{*}$ denotes the frequent itemsets still appearing in the sanitized database. Therefore, the term $|FIs - SIs - FIs^{*}|$ is the number of non-sensitive frequent itemsets that are hidden by the sanitization process. The influence of varying the minimum support threshold on N-T-H is presented in Fig. 8.

In Fig. 8, it can be observed that the proposed PSO2DT algorithm generates more N-T-H than GA-based algorithms, especially in Fig. 8(a) and (b). The reason is that for dense datasets such as chess and mushroom, the proposed PSO2DT algorithm completely hides all sensitive itemsets in most cases, as it was respectively shown in Fig. 6(a) and (b). However, there is a trade-off relationship between the F-T-H and N-T-H since if more sensitive itemsets are hidden, more frequent itemsets will be missed. Thus, some non-sensitive information may be hidden by the sanitization process through transaction deletion. But this rule does not hold for Greedy. It can be observed that the N-T-H of Greedy is generally the worse among all algorithms. The F-T-H of the three GA-based algorithms on foodmart and T10I4D100K, and the F-T-H of the designed PSO2DT algorithm on all datasets, are less than Greedy. This is because Greedy selects transactions to be deleted without considering the minimization of side effects. Based on a two-way ANOVA analysis, the number of N-T-H for the PSO2DT algorithm is significantly different from the other algorithms except for the Greedy algorithm ($F = 13.794$, $p < 0.05$, shown in Fig. 8(a)), but there is no significant difference between all compared algorithms ($F = 0.999$, $p > 0.05$, shown in Fig. 8(b)). For sparse datasets, it can be observed that the proposed PSO2DT algorithm outperforms the GA-based algorithms and generates nearly zero N-T-H in some cases compared to the GA-based algorithms, as shown in Fig. 8 (c) and (d). However, there are no significant differences between the compared algorithms ($F = 2.648$, $p = 0.097 > 0.05$, shown in Fig. 8(c); $F = 1$, $p = 0.426 > 0.05$, shown in Fig. 8(d)). Results obtained in terms of N-T-H, when varying the sensitive percentage of FIs are shown in Fig. 9.

In Fig. 9, it can be observed that the proposed PSO2DT algorithm also generates more N-T-H compared to the GA-based sGA2DT, pGA2DT, and cpGA2DT algorithms but less than Greedy, in most cases. This can be observed in Fig. 9(a) and (b) for dense datasets. The reason is the same as for Fig. 8. For sparse datasets such as foodmart (results shown in Fig. 9(c)), the proposed PSO2DT algorithm still outperforms GA-based algorithms and results are similar for the other algorithms (as shown in Fig. 9(d)). According to a two-way ANOVA analysis, results for the proposed PSO2DT algorithm are significantly different from the other algorithms except for the Greedy algorithm ($F = 33.525$, $p < 0.05$, shown in Fig. 9(a); $F = 119.184$, $p < 0.001$, shown in Fig. 9(b); $F = 11.88$, $p < 0.05$, shown in Fig. 9(c)). However, there is no significant difference with the compared algorithms in Fig. 9(d) ($F = 1.000$, $p = 0.426 > 0.05$), as results are the same for the compared algorithms. Overall, the proposed PSO2DT algorithm achieves better results on sparse datasets.

### 6.2.3. *Fitness value*

To further compare the designed algorithm with GA-based algorithms and Greedy, the fitness values of the final solutions have also been compared. The weights of the three side effects in the fitness function can be adjusted according to the user's preferences. To hide as much sensitive information as possible, $w_1$ is set much higher than the other two weights in our experiments. Thus, $w_1$, $w_2$, and $w_3$ are respectively set to 0.98, 0.01 and 0.01 for the dense chess and mushroom datasets. The $w_1$, $w_2$, and $w_3$ weights are respectively set to 0.8, 0.1 and 0.1 for the sparse foodmart and T104D100K datasets. The fitness values obtained for various minimum support threshold values are shown in Fig. 10.

In Fig. 10, it is obvious that the fitness values obtained by the designed algorithm are generally better than those obtained by the four other algorithms. For the chess dataset (Fig. 10(a)), the reason is the F-T-H side-effect, since the designed PSO2DT algorithm can always hide all sensitive itemsets successfully for that dataset. Although Greedy can hide more sensitive itemsets than the three GA-based algorithms for the two dense datasets (Figs. 6 and 7), its fitness values are much higher than those of GA-based algorithms in most cases for the chess and mushroom datasets. This is because Greedy generates a large amount of N-T-H itemsets. Since there is a trade-off relationship between F-T-H and N-T-H, the N-T-H side effect has appeared. This problem occurs for very dense datasets. Although the number of F-T-H itemsets produced by the designed PSO2DT algorithm is similar to other approaches, PSO2DT can still obtain less N-T-H and N-T-G and the fitness values of its final solutions are smaller for the two sparse datasets. This can be observed in Fig. 10(a) and (b). According to a two-way ANOVA analysis, results for the PSO2DT algorithm are significantly different from the other algorithms ($F = 6.767$, $p < 0.05$, shown in Fig. 10(a); $F = 9.283$, $p < 0.05$, shown in Fig. 10(b)). For sparse datasets (results shown in Fig. 10(c) and (d)), results for the proposed PSO2DT algorithm are not significantly different from the other algorithms except for the Greedy algorithm ($F = 6.197$, $p < 0.05$, shown in Fig. 10(c); $F = 35.7$, $p < 0.001$, shown in Fig. 10(d)). The fitness values obtained for various sensitive percentages of FIs are shown in Fig. 11.

In Fig. 11, results similar to Fig. 10 can be observed. The fitness values of the designed algorithm is up to about two or three times less than the other algorithms, especially for dense datasets. Based on a two-way ANOVA analysis, fitness values obtained by the proposed PSO2DT algorithm are significantly different from the other algorithms except for the Greedy algorithm ($F = 17.267$, $p < 0.001$, shown in Fig. 11(a)). Besides, results for the proposed PSO2DT algorithm are significantly different from all compared algorithms ($F = 11.644$, $p < 0.05$, shown in Fig. 11(b)). For the sparse foodmart and T10I4D100K datasets, the proposed PSO2DT algorithm still outperforms GA-based algorithms, as shown in Fig. 11(c), and has nearly the same fitness values as the other approaches, as shown in Fig. 11(d). The fitness values of the PSO2DT algorithm are significantly different from the Greedy

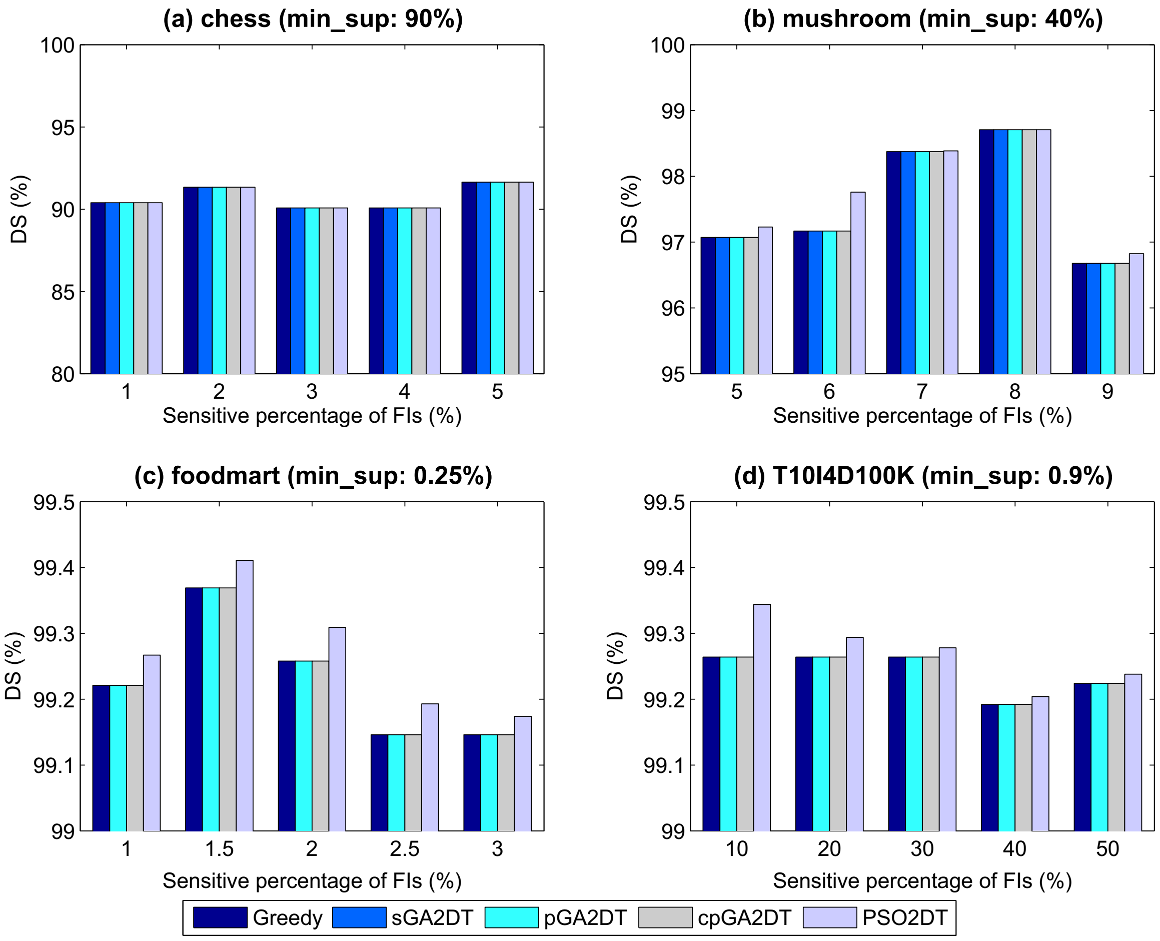

**Fig. 13.** DS w.r.t. various sensitive percentages of FIs.

algorithm but not others ( $F = 24.013$, $p < 0.001$, shown in Fig. 11(c); $F = 5.438$, $p < 0.05$, shown in Fig. 11(d)). From the above discussion, it can be concluded that the proposed PSO2DT algorithm can obtain better results because it considers the three side effects, unlike the other GA-based algorithms.

### 6.3. 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 smaller for an algorithm than for the other algorithms, it means that the algorithm has selected a better set of transactions for deletion, and thus avoid deleting irrelevant transactions that may result in hiding more non-sensitive but frequent itemsets. The DS is defined as:

$$DS = \frac{|D^*|}{|D|}, \tag{18}$$

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

Results w.r.t. various minimum support threshold values and sensitive percentages of FIs are respectively shown in Figs. 12 and 13.

From the results shown in Figs. 12 and 13, it can be observed that all compared algorithms achieve decent database integrity since DS values are never less than 90%. Since the given maximum number of deleted transactions for the Greedy algorithm is the number of transactions that GA-based algorithms delete, the DS of Greedy is the same as GA-based algorithms. The proposed PSO2DT algorithm outperforms GA-based algorithms in terms of DS. This indicates that the proposed PSO2DT algorithm can completely hide sensitive itemsets while still preserving good database integrity. Based on the conducted experiments, the designed PSO2DT algorithm can find better transactions for deletion, that minimize the three side effects, and also maintain high database similarity through transaction deletion. Based on a two-way ANOVA analysis, results for the PSO2DT algorithm are significantly different from the other algorithms ( $F = 15.980$, $p < 0.001$, shown in Fig. 12(a); $F = 7.431$, $p < 0.05$, shown in Fig. 12(b); $F = 107.233$, $p < 0.001$, shown in Fig. 12(c)). However, there is no significant difference between the compared algorithms since all of them preserve good database integrity when hiding sensitive itemsets ( $F = 3.093$, $p = 0.223 > 0.05$, shown in Fig. 13(a); $F = 0.057$, $p > 0.05$, shown in Fig. 13(b)). Besides, DS for the PSO2DT algorithm is not significantly different from the compared algorithms for sparse datasets ($F = 116.381$, $p < 0.001$, shown in Fig. 13(c); $F = 5.396$, $p < 0.05$, shown in Fig. 13(d)). In summary, the proposed PSO2DT algorithm is thus acceptable for hiding sensitive itemsets through transaction deletion.

## 7. Conclusion and future work

In the past, GA-based algorithms have been proposed to hide sensitive itemsets through transaction deletion. This is the first paper that design a PSO-based approach to hide sensitive itemsets through transaction deletion, while minimizing side effects. In the designed PSO2DT algorithm, the maximum number of transactions to be deleted is automatically determined and set as the particle size. The proposed algorithm is more flexible than previous GA-based approaches and the non-evolutionary Greedy algorithm, as much less parameters need to be set by the user. Furthermore, the designed PSO2DT algorithm adopts the pre-large concept to avoid performing multiple database scans, and thus speed up the evolution process. Substantive experiments have been carried to compare the performance of the designed algorithm with the state-of-the-art GA-based approaches and Greedy algorithm in terms of the three side effects and database similarity (integrity). Experimental results show that the designed algorithm is much faster and generates a smaller N-T-H side-effect compared to GA-based approaches. Since there is a trade-off relationship between the F-T-H and N-T-H side effects, the proposed algorithm generates more N-T-H than GA-based approaches for dense datasets but still has good results for sparse datasets. Besides, compared with the Greedy approach, the designed PSO2DT algorithm provides excellent results in all cases. Moreover, the proposed PSO2DT algorithm achieves greater database integrity compared to GA-based approaches on all datasets.

In this paper, a PSO-based approach was designed. But it remains possible to design sanitization algorithms based on other evolutionary approaches such as ant colony optimization (ACO) and covariance matrix adaptation evolution strategy (CMA-ES) to find better solutions for hiding sensitive itemsets. However, it is a non-trivial task to define the formulas for hiding sensitive itemsets through transaction deletion using an ACO framework (especially the process for updating pheromone). 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, and DS are considered as an object. Evolutionary algorithms such as GA, PSO, ACO, and CMA-ES are suitable for this issue. In addition, the Pareto approach used in multi-objective optimization problems can be considered to find balanced solutions satisfying the criteria of interest. A multi-component vector function could also be considered as another solution for PPDM since it could be easier to implement and integrate in the PSO approach. The above issues will be considered in our future work.

## Acknowledgment

This research was partially supported by National Natural Science Foundation of China (NSFC) under Grant no. 61503092, by the Natural Scientific Research Innovation Foundation in Harbin Institute of Technology under Grant HIT.NSRIF.2014100, and by the Shenzhen Strategic Emerging Industries Program under Grant ZDSY20120613125016389.

## References

Agrawal, R., Srikant, R., 1994a. Quest Synthetic Data Generator. IBM Almaden Research Center 〈http://www.Almaden.ibm.com/cs/quest/syndata.html〉.

Agrawal, R., Srikant, R., 1994b. Fast algorithms for mining association rules in large databases. In: The International Conference on Very Large Data Base, San Francisco, CA, USA, pp. 487–499.

Agrawal, R., Srikant, R., 1995. Mining sequential patterns. In: The International Conference on Data Engineering, pp. 3–14.

Agrawal, R., Srikant, R., 2000. Privacy-preserving data mining. ACM SIGMOD Rec. 29, 439–450.

Aggarwal, C.C., Pei, J., Zhang, B., 2006. On privacy preservation against adversarial data mining. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 510–516.

Atallah, M., Bertino, E., Elmagarmid, A., Ibrahim, M., Verykios, V., 1999. Disclosure limitation of sensitive rules. In: The Workshop on Knowledge and Data Engineering Exchange, pp. 45–52.

Bonam, J., Reddy, A.R., Kalyani, G., 2014. Privacy preserving in association rule mining by data distortion using PSO. In: ICT and Critical Infrastructure: Proceedings of the 48th Annual Convention of Computer Society of India—vol. II, pp. 551–558.

Chen, M.S., Han, J., Yu, P.S., 1996. Data mining: an overview from a database perspective. IEEE Trans. Knowl. Data Eng. 8 (6), 866–883.

Clifton, C., Kantarcioglu, M., Vaidya, J., Lin, X., Zhu, M.Y., 2003. Tools for privacy preserving distributed data mining. ACM SIGKDD Explor. 4, 1–7.

Dasseni, E., Verykios, V.S., Elmagarmid, A.K., Bertino, E., 2001. Hiding association rules by using confidence and support. In: International Workshop on Information Hiding, pp. 369–383.

Dwork, C., McSherry, F., Nissim, K., Smith, A., 2006. Calibrating noise to sensitivity in private data analysis. In: Lecture Notes in Computer Science, vol. 3876, pp. 265–284.

Evfimievski, A., Srikant, R., Agrawal, R., Gehrke, J., 2002. Privacy preserving mining of association rules. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 217–228.

Frequent Itemset Mining Dataset Repository ⟨http://fimi.ua.ac.be/data/⟩, 2012.

Goldberg, D.E., 1989. Genetic Algorithms in Search, Optimization and Machine Learning. Addison-Wesley Longman Publishing Co., Inc, Boston, MA, USA.

Hajian, S., Domingo-Ferrer, J., Farrs, O., 2014. Generalization-based privacy preservation and discrimination prevention in data publishing and mining. Data Min. Knowl. Discov. 28 (5–6), 1158–1188.

Han, J., Pei, J., Yin, Y., Mao, R., 2004. Mining frequent patterns without candidate generation: a frequent-pattern tree approach. Data Min. Knowl. Discov. 8 (1), 53–87.

Han, S., Ng, W.K., 2007. Privacy-preserving genetic algorithms for rule discovery. In: Lecture Notes in Computer Science, pp. 407–417.

Harik, G.R., Lobo, F.G., Goldberg, D.E., 1999. The compact genetic algorithm. IEEE Trans. Evol. Comput. 3 (4), 287–297.

Holland, J.H., 1992. Adaptation in Natural and Artificial Systems. MIT Press.

Hong, T.P., Wang, C.Y., Tao, Y.H., 2001. A new incremental data mining algorithm using pre-large itemsets. Intell. Data Anal. 5, 111–129.

Hong, T.P., Lin, C.W., Yang, K.T., Wang, S.L., 2012. Using TF-IDF to hide sensitive itemsets. Appl. Intell. 38 (4), 502–510.

Islam, Z., Brankovic, L., 2011. Privacy preserving data mining: a noise addition framework using a novel clustering technique. Knowl.-Based Syst. 24 (8), 1214–1223.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: IEEE International Conference on Neural Networks, pp. 1942–1948.

Kennedy, J., Eberhart, R., 1997. A discrete binary version of particle swarm algorithm. In: IEEE International Conference on Systems, Man, and Cybernetics, pp. 4104–4108.

Kuo, R.J., Chao, C.M., Chiu, Y.T., 2011. Application of particle swarm optimization to association rule mining. Appl. Soft Comput. 11 (1), 326–336.

Lindell, Y., Pinkas, B., 2000. Privacy preserving data mining. In: The Annual International Cryptology Conference on Advances in Cryptology, pp. 36–54.

Lin, C.W., Hong, T.P., Chang, C.C., Wang, S.L., 2013. A greedy-based approach for hiding sensitive itemsets by transaction insertion. J. Inf. Hiding Multimed. Signal Process. 4, 201–227.

Lin, C.W., Hong, T.P., Yang, K.T., Wang, S.L., 2015a. The GA-based algorithms for optimizing hiding sensitive itemsets through transaction deletion. Appl. Intell. 42 (2), 210–230.

Lin, C.W., Yang, L., Fournier-Viger, P., Wu, M.T., Hong, T.P., Wang, S.L., 2015b. A swarm-based approach to mine high-utility itemsets. Multidiscip. Soc. Netw. Res., 572–581.

Lin, C.W., Zhang, B., Yang, K.T., Hong, T.P., 2014. Efficiently hiding sensitive itemsets with transaction deletion based on genetic algorithms. Sci. World J., 13 (Article ID 398269).

Menhas, M.I., Fei, M., Wang, L., Fu, X., 2011. A novel hybrid binary PSO algorithm. In: Lecture Notes in Computer Science, vol. 6728, pp. 93–100.

Microsoft. Example Database Foodmart of Microsoft Analysis Services ⟨http://msdn.microsoft.com/en-us/library/aa217032(SQL.80).aspx⟩.

Murty, M.N., Flynn, P.J., 1999. Data clustering: a review. ACM Comput. Surv. 31 (3), 264–323.

Mooney, C.H., Roddick, J.F., 2013. Sequential pattern mining—approaches and algorithms. ACM Comput. Surv. 45 (2), 1–39.

Oliveira, S.R.M., Zaane, Osmar R., 2002. Privacy preserving frequent itemset mining. In: IEEE International Conference on Privacy, Security and Data Mining, pp. 43–54.

Pandya, B.K., Dixit, K., Singh, U.K., Bunkar, K., 2014. Effectiveness of multiplicative data perturbation for privacy preserving data mining. Int. J. Adv. Res. Comput. Sci. 5 (6), 112–115.

Pears, R., Koh, Y.S., 2012. Weighted association rule mining using particle swarm optimization. In: Lecture Notes in Computer Science, vol. 7104, pp. 327–338.

Quinlan, J.R., 1993. C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers Inc, San Francisco, CA, USA.

Sweeney, L., 2002. k-anonymity: a model for protecting privacy. Int. J. Uncertain. Fuzziness Knowl.-Based Syst., 557–570.

Sarath, K.N.V.D., Ravi, V., 2013. Association rule mining using binary particle swarm optimization. Eng. Appl. Artif. Intell. 26, 1832–1840.

Shen, M., Zhan, Z.H., Chen, W.N., Gong, Y.J., Zhang, J., Li, Y., 2014. Bi-velocity discrete particle swarm optimization and its application to multicast routing problem in communication networks. IEEE Trans. Ind. Electron. 61 (12), 7141–7151.

Tian, Y., Liu, D., Yuan, D., Wang, K., 2013. A discrete PSO for two-stage assembly scheduling problem. Int. J. Adv. Manuf. Technol. 66 (1–4), 481–499.

Verykios, V.S., Bertino, E., Fovino, I.N., Provenza, L.P., Saygin, Y., Theodoridis, Y., 2004. State-of-the-art in privacy preserving data mining. ACM SIGMOD Rec. 33, 50–57.

Wu, Y.H., Chiang, C.M., Chen, A.L.P., 2007. Hiding sensitive association rules with limited side effects. IEEE Trans. Knowl. Data Eng. 19, 29–42.

Zaki, M., 2001. SPADE: an efficient algorithm for mining frequent sequences. Mach. Learn. 42 (1–2), 31–60.

Zuo, X., Zhang, G., Tan, W., 2014. Self-adaptive learning PSO-based deadline constrained task scheduling for hybrid IaaS cloud. IEEE Trans. Autom. Sci. Eng. 11 (2), 564–573.

Zhi, X.H., Xing, X.L., Qang, Q.X., Zhang, L.H., 2004. A discrete PSO method for generalized TSP problem. In: IEEE International Conference on Machine Learning and Cybernetics, pp. 2378–2383.