Engineering Applications of Artificial Intelligence 55 (2016) 320–330

Contents lists available at ScienceDirect

Engineering Applications of Artificial Intelligence

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

Mining high-utility itemsets based on particle swarm optimization

Jerry Chun-Wei Lin [a,*], Lu Yang [a], Philippe Fournier-Viger [b], Jimmy Ming-Thai Wu [c], Tzung-Pei Hong [d,f], Leon Shyue-Liang Wang [e], Justin Zhan [c]

[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, University of Nevada, Las Vegas, USA
[d] Department of Computer Science and Information Engineering, National University of Kaohsiung, Kaohsiung, Taiwan, ROC
[e] Department of Information Management, National University of Kaohsiung, Kaohsiung, Taiwan, ROC
[f] Department of Computer Science and Engineering, National Sun Yat-sen University, Kaohsiung, Taiwan, ROC

ARTICLE INFO

Article history:
Received 18 June 2015
Received in revised form
19 May 2016
Accepted 31 July 2016

Keywords:
Evolutionary computation
High-utility itemsets
Particle swarm optimization
Discrete
Transaction-weighted utility model

ABSTRACT

High-utility itemset mining (HUIM) is a critical issue in recent years since it can be used to reveal the profitable products by considering both the quantity and profit factors instead of frequent itemset mining (FIM) or association-rule mining (ARM). Several algorithms have been presented to mine high-utility itemsets (HUIs) and most of the designed algorithms have to handle the exponential search space for discovering HUIs when the number of distinct items and the size of database are very large. In the past, a heuristic $HUPE_{umu}$-GRAM algorithm was proposed to mine HUIs based on genetic algorithm (GA). For the evolutionary computation (EC) techniques of particle swarm optimization (PSO), it only requires fewer parameters compared to the GA-based approach. Since the traditional PSO mechanism is used to handle the continuous problem, in this paper, the discrete PSO is adopted to encode the particles as the binary variables. An efficient PSO-based algorithm namely $HUIM\text{-}BPSO_{sig}$ is proposed to efficiently find HUIs. It first sets the number of discovered high-transaction-weighted utilization 1-itemsets (1-HTWUIs) as the size of a particle based on transaction-weighted utility (TWU) model, which can greatly reduce the combinational problem in evolution process. The *sigmoid* function is adopted in the updating process of the particles of the designed $HUIM\text{-}BPSO_{sig}$ algorithm. Substantial experiments on real-life datasets show that the proposed algorithm has better results compared to the state-of-the-art GA-based algorithm.

1. Introduction

Knowledge discovery in database (KDD) is an emerging issue since the potential or implicit information can be found from a very large database. Most of them, frequent itemset mining (FIM) or association-rule mining (ARM) has been extensively developed to mine the set of frequent itemsets in which their occurrence frequency of an itemset is no less than minimum support threshold or its confidence is no less than minimum confidence threshold (Agrawal and Srikant, 1994; Chen et al., 1996). Since only the occurrence frequency of itemsets is discovered whether in FIM or ARM, it is insufficient to identify the high profit itemsets especially when the itemset is rarely appeared but has high profit value. To solve the limitation of FIM or ARM, high-utility itemset mining (HUIM) (Yao et al., 2004; Yao and Hamilton, 2006) was designed to discover the "useful" and "profitable" itemsets from the quantitative databases. An itemset is considered as a high-utility itemset (HUI) if its utility value is no less than the user-specified minimum utility threshold.

The traditional algorithms of HUIM have to handle the "exponential problem" of a very huge search space while the number of distinct items or the size of database is very large. Evolutionary computation is an efficient way and able to find the optimal solutions using the principles of natural evolution (Cattral et al., 2009). The genetic algorithm (GA) (Holland, 1975) is an optimization approach to solve the NP-hard and non-linear problems and used to investigate a very large search spaces to find the optimal solutions based on the designed fitness functions with various operators such as selection, crossover, and mutation. Several GA-based algorithms have been extensively studied and applied to the variants of optimization problems (Kannimuthu and Premalatha, 2014; Martnez-Ballesteros et al., 2010; Salleb-Aouissi et al., 2007).

* Corresponding author.
E-mail addresses: jerrylin@ieee.org (J.-W. Lin), luyang@ikelab.net (L. Yang), phifv@hitsz.edu.cn (P. Fournier-Viger), wmt@wmt35.idv.tw (J.-T. Wu), tphong@nuk.edu.tw (T.-P. Hong), slwang@nuk.edu.tw (L.-L. Wang), justin.zhan@unlv.edu (J. Zhan).

http://dx.doi.org/10.1016/j.engappai.2016.07.006
0952-1976/

In the past, Kannimuthu and Premalatha adopted the genetic algorithm approach and developed the high utility pattern extracting using genetic algorithm with ranked mutation using minimum utility threshold ($HUPE_{umu}$-GRAM) to mine HUIs (Kannimuthu and Premalatha, 2014). Another algorithm called $HUPE_{wumu}$-GRAM was also designed to mine HUIs without specified minimum utility threshold (Kannimuthu and Premalatha, 2014). Instead of GAs, Particle Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) is also a bio-inspired and population-based approach for finding the optimal solutions by adopting the velocity to update the particles.

In this paper, a binary PSO-based (BPSO) (Kennedy and Eberhart, 1997) algorithm is designed for mining the HUIs. The key contributions of this paper are described below:

1. Fewer algorithms have been developed to find the HUIs based on evolutionary computation. According to authors' knowledge, this is the first paper to mine the high-utility itemsets based on binary PSO mechanism.
2. A discrete PSO-based algorithm namely HUIM-$BPSO_{sig}$ is thus designed to find the HUIs by integrating the sigmoid updating strategy and TWU model. It first sets the number of discovered 1-HTWUIs as the particle size. This approach can find the potential HUIs based on TWU model, which can be used to reduce the combinational problem in traditional HUIM especially when the number of distinct items in the database or the size of database is very large.
3. Extensive experiments were conducted on real-life datasets to evaluate the performance of the proposed approach. Results showed that the proposed approach can efficiently identify the valid HUIs from a very condense datasets and has better results compared to the state-of-the-art GA-based algorithm.

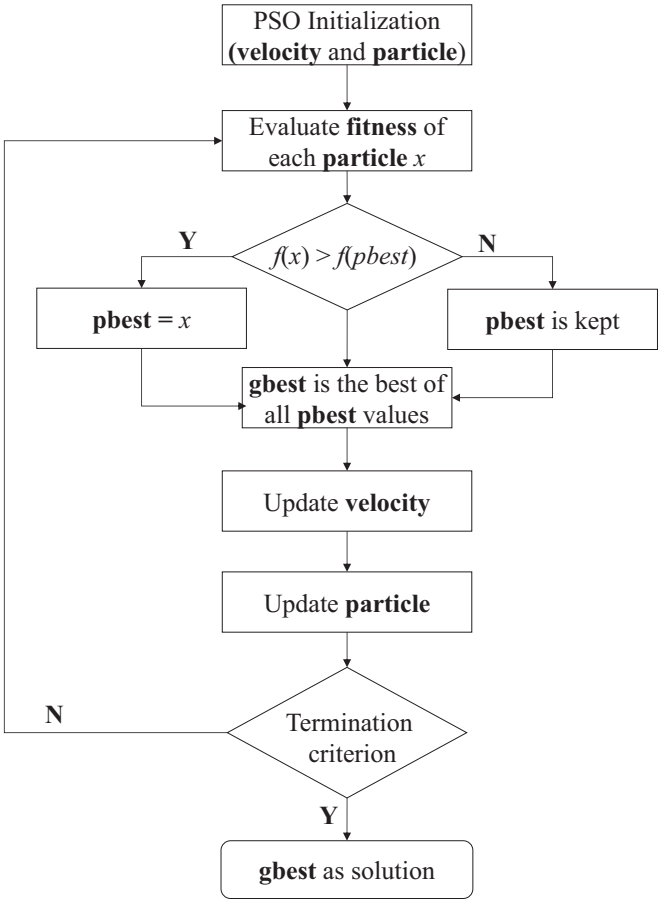

Fig. 1. Flowchart of standard PSO.

2. Related work

In this section, works related to particle swarm optimization (PSO) and high-utility itemset mining (HUIM) are briefly reviewed.

2.1. Particle swarm optimization

In the past, many heuristic algorithms have been facilitated to solve the optimization problems for discovering the necessary information in the evolutionary computation (Cattral et al., 2009; Martnez-Ballesteros et al., 2010). Kennedy and Eberhart first introduced Swarm Optimization (PSO) (Kennedy and Eberhart, 1995) in 1995.

Many individuals (particles) are stochastically generated in the search space of PSO in the evolution process. Each particle is represented as an optimized solution by composing a set of velocities in the evolution process. Instead of GA, each particle has memories to keep its previous best particle (personal best, *pbest*) and its previous best particle by considering its neighborhoods (global best, *gbest*). The flowchart of standard PSO is shown in Fig. 1.

The PSO was originally defined to solve the continuous problems. In the updating process of PSO, the corresponding particle and velocity are described as follows:

$$v_i^j(t+1) = w_1 \times v_i^j(t) + c_1 \times rand() \times (pbest_i^j - x_i^j(t)) + c_2 \times rand() \times (gbest^j - x_i^j(t)). \quad (1)$$

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

In the Eqs. (1) and (2), t represents current number of iterations; w_1 plays a balancing role between global search and local search; $v_i^j(t)$ is represented the velocity of j-th position in particle i; $x_i^j(t)$ is represented the item of j-th position in particle i; $rand()$ is the random number in range of (0, 1); c_1 is the individual factor and c_2 is the social factor, which are usually set as 2. In the past, PSO has been adopted to various real-world applications. Kuo et al. designed an algorithm to mine the association rules (ARs) from the investor's stock purchase behavior by using the designed itemset range (IR) value instead of the specified minimum support and minimum confidence thresholds (Kuo et al., 2011). Pears and Koh presented a feasible method to mine the weighted association rule mining based on PSO (Pears and Koh, 2012). Kennedy and Eberhart also designed a discrete (binary) PSO (BPSO) (Kennedy and Eberhart, 1997) to solve the limitation of continuous PSO. Each particle in BPSO is represented as a set of binary variables. The velocity in the BPSO is updated in terms of probabilities by adopting sigmoid function. Sarath and Ravi also designed a BPSO optimization approach to discover ARs (Sarath and Ravi, 2013). Other applications by adopting PSO to mine the required information are still in progress (Menhas et al., 2011; Nouaouria et al., 2013).

2.2. High-utility itemset mining

High-utility itemset mining (HUIM) (Yao et al., 2004; Yao and Hamilton, 2006) is a critical issue and an emerging topic in recent decades, which can be concerned as the extension of frequent itemset mining (FIM) but more factors such as quantity and profit are considered in it. Many algorithms were respectively developed to mine the set of complete HUIs (Chan et al., 2003; Yao et al., 2004; Yao and Hamilton, 2006). Liu et al. first developed a two-phase (TWU) model and designed the transaction-weighted downward closure (TWDC) property to early prune the unpromising HUIs but still can discover the complete HUIs (Liu et al., 2005). In the past, Lin et al. designed a high-utility-pattern (HUP)-tree for discovering HUIs (Lin et al., 2011). Tseng et al. presented the UP-tree structure and UP-growth mining algorithm (Tseng et al., 2010) for discovering HUIs.

Since the tree-based algorithm limits the memory usage, Lan et al. developed an projection-based algorithm based on the index mechanism to fast mine HUIs (Lan et al., 2013). Fournier-Viger et al. then presented an algorithm namely FHM to quickly mine HUIs based on the built Estimated Utility Co-occurrence Structure (EUCS) (Fournier-Viger et al., 2014) and the list-based approach (Liu and Qu, 2012). Other related algorithms are also developed in the progress (Zihayat and An, 2014; Zida et al., 2015). Instead of traditional HUIM, Kannimuthu and Premalatha first designed the GA-based algorithm to mine HUIs with the ranked mutation (Kannimuthu and Premalatha, 2014). In their approach, it is not easy to find the initial 1-HTWUIs as the chromosome to find the valid HUIs. A very huge computation is necessary to initially set the appropriate chromosomes for mining HUIs. Moreover, some specific parameters are required in the evolution process of GAs, which is a non-trivial task to set the appropriate values of them.

3. Preliminaries and problem statement

3.1. Preliminaries

Let $I=\{i_1, i_2, \ldots, i_m\}$ be a finite set of m distinct items. A quantitative database is a set of transactions $D=\{T_1, T_2, \ldots, T_n\}$, where each transaction $T_q \in D$ $(1 \le q \le n)$ is a subset of I and has a unique identifier q, called its *TID*. Besides, each item i_j in a transaction T_q has a purchase quantity (internal utility) denoted as $q(i_j, T_q)$. A profit table $ptable=\{pr_1, pr_2, \ldots, pr_m\}$ indicates the profit value pr_j of each item i_j. A set of k distinct items $X=\{i_1, i_2, \ldots, i_k\}$ such that $X \subseteq I$ is said to be a k-itemset, where k is the length of the itemset. An itemset X is said to be contained in a transaction T_q if $X \subseteq T_q$. A minimum utility threshold is set as δ according to users' preference.

Table 1
A quantitative database.

TID	Transaction (item, quantity)
T_1	*a*:1, *c*:18, *e*:1,
T_2	*b*:6, *d*:1, *e*:1, *f*:1
T_3	*a*:2, *c*:1, *e*:1
T_4	*d*:1, *e*:1
T_5	*c*:4, *e*:2
T_6	*b*:1, *f*:1
T_7	*b*:10, *d*:1, *e*:1
T_8	*a*:3, *c*:25, *d*:3, *e*:1
T_9	*a*:1, *b*:1, *f*:3
T_{10}	*b*:6, *c*:2, *e*:2, *f*:4

An illustrated example is stated in Table 1 as the running example in this paper. In Table 1, it has 10 transactions and 6 distinct items, denoted from (*a*) to (*f*). The profit value (external utility) of each item is set as { *pr*(*a*): 3, *pr*(*b*): 9, *pr*(*c*): 1, *pr*(*d*): 5, *pr*(*e*): 6, *pr*(*f*): 1}, and the minimum utility threshold is set as (δ=30%).

Definition 1. The utility of an item i_j in a transaction T_q is denoted as $u(i_j, T_q)$, and is defined as:

$$u(i_j, T_q) = q(i_j, T_q) \times pr(i_j). \tag{3}$$

For example, the utility of item (*a*) in transaction T_1 is calculated as $u(a, T_1) = q(a, T_1) \times pr(a)$ $(=1 \times 3)(=3)$.

Definition 2. The utility of an itemset X in transaction T_q is denoted as $u(X, T_q)$, and defined as:

$$u(X, T_q) = \sum_{i_j \subseteq X \wedge X \in T_q} u(i_j, T_q). \tag{4}$$

For example, the utility of an itemset (ac) in transaction T_1 is calculated as $u(ac, T_1)=u(a, T_1) + u(c, T_1)(= 1 \times 3 + 18 \times 1)(= 21)$.

Definition 3. The utility of an itemset X in a database D is denoted as $u(X)$, and defined as:

$$u(X) = \sum_{X \subseteq T_q \wedge T_q \in D} u(X, T_q). \tag{5}$$

For example, the utility of an itemset (b) in D is calculated as $u(b)= u(b, T_2) + u(b, T_6) + u(b, T_7) + u(b, T_9) + u(b, T_{10})$ $(=54+9+90+9+54)(=216)$.

Definition 4. The transaction utility of a transaction T_q is denoted as $tu(T_q)$, and defined as:

$$tu(T_q) = \sum_{X \subseteq T_q} u(X, T_q). \tag{6}$$

For example, $tu(T_1)= u(a, T_1) + u(c, T_1) + u(e, T_1)$ $(=3+18+6)$ $(=27)$.

Definition 5. The transaction-weighted utility of an itemset X in a database D is denoted as $twu(X)$, and defined as:

$$twu(X) = \sum_{X \subseteq T_q \wedge T_q \in D} tu(T_q). \tag{7}$$

For example, $twu(b)= tu(T_2) + tu(T_6) + tu(T_7) + tu(T_9) + tu(T_{10})(=66+10+101+15+72)(=264)$.

Definition 6. The total utility of a database D is denoted as TU, and defined as:

$$TU = \sum_{T_q \in D} tu(T_q). \tag{8}$$

For example, the total utility in a database D is calculated as TU $(=27+66+13+11+16+10+101+55+15+72)(=386)$.

Definition 7. An itemset X in a database D is a high transaction-weighted utilization itemset (HTWUI) iff its *twu* is no less than the minimum utility value as:

$$HTWUI \leftarrow \{X | twu(X) \ge TU \times \delta\}. \tag{9}$$

For example, the itemset (a) is not a HTWUI since its $twu(a)(=110<386\times0.3=115.8)$. The itemset (b) is a HTWUI since its $twu(b)(=264>115.8)$.

Definition 8. An itemset X in a database D is a high-utility itemset (HUI) iff its utility is no less than the minimum utility value as:

$$HUI \leftarrow \{X|u(X)\geq TU\times\delta\}. \quad (10)$$

For example, the utility of itemsets (b) and (bc) are respectively calculated as $u(b)$ $(=216)$ and $u(bc)$ $(=56)$. Thus, the itemset (b) is a HUI since $u(b)(=216>115.8)$. The itemset (bc) is not a HUI since $u(bc)(=56<115.8)$.

3.2. Problem statement

Let D be a quantitative transactional database, its profit table *ptable*, and a user-specified minimum utility threshold δ. The problem of HUIM from D is to find the set of high-utility itemsets, in which the utility of an itemset X is no less than $(TU\times\delta)$.

4. Proposed BPSO-based framework of HUIM

In the designed binary PSO (BPSO)-based model (Kennedy and Eberhart, 1997) for mining HUIs, it consists of pre-processing, particle encoding, fitness evaluation and the updating phases to mine HUIs. Details of four phases are respectively described below and the designed HUIM-BPSO$_{sig}$ is shown in Algorithm 1.

4.1. Pre-processing phase

In the designed HUIM-BPSO$_{sig}$ algorithm, the TWU model (Liu et al., 2005) of traditional HUIM is first adopted to discover high-transaction-weighted utilization 1-itemsets (1-HTWUIs). Based on the transaction-weighted downward closure (TWDC) property of TWU model, the unpromising items can be early pruned. First, the utility of items of each transaction is calculated as the transaction utility (*tu*) (Lines 1-3 in Algorithm 1). The transaction-weighted utility (*twu*) of an item is thus calculated by sum up the transaction utility if an item appearing in the transaction (Line 4 in Algorithm 1). This process is used to estimate the upper-bound value of an item based on the TWU model. If the transaction-weighted utility of an item is no less than the minimum utility value, it is considered as a HTWUI (Lines 5-6 in Algorithm 1). In this example, the minimum utility value is calculated as $(386\times0.3=115.8)$. The $twu(a)$ is calculated as: $twu(a)(=27+13+55+15)(=110<115.8)$; (a) is not a 1-HTWUI. The resting items are calculated in the same way. After that, the discovered 1-HTWUIs respectively are {b: 264, c: 183, d: 233, e: 361, f: 163}.

4.2. Particle encoding phase

Each particle in the designed approach is to present a set of items forming the potential HUI. The size of the particle is the number of 1-HTWUIs found in the first pre-processing phase (Line 7 in Algorithm 1). Each particle is composed by a set of binary variables as 0 or 1 (Line 11 in Algorithm 1), which indicates that a corresponding item is present or absent in a particle. If the corresponding j-th position of a particle is set as 1, it indicates that the an item in j-th position is presented as a potential HUI; otherwise, this item is not included and cannot be a potential HUI in the evolution process. Note that the discovered 1-HTWUIs are sorted in *alphabetic-ascending* order corresponding to the positions in the particle. Initially, the discovered *twu* values of 1-HTWUIs will be normalized as the probabilities to initialize the particles in the population (Line 8 in Algorithm 1). The velocities are randomly generated in the range of (0, 1) (Line 12 in Algorithm 1). In this example, the size of a particle is set as 5 and the particles are represented in Fig. 2.

	b	*c*	*d*	*e*	*f*
p_1	1	0	0	1	1
p_2	0	1	0	0	1
p_3	0	0	1	0	0
⋮					
p_n	0	1	0	1	0

Fig. 2. Encoding of particles.

4.3. Fitness evaluation phase

The fitness value in the designed algorithm is utilized to evaluate the utility value of each particle by the designed fitness function. The fitness function for discovering HUIs is the same as the traditional HUIM, which is stated as:

$$fitness(p_i)=u(X), \quad (11)$$

in which X is the union of the j-th item in the particle if its value is set as 1. In this example, the itemset of particle p_1 is (bef), and the particle p_2 is (cf). Thus, if the utility value of a particle is no less than the minimum utility count, it is concerned as a HUI and will be putted into the set of HUIs (Lines 13 and 14 in Algorithm 1).

4.4. Updating phase

After the evaluation process of the particles in a population, the velocity of the particles is updated according to the traditional PSO approach shown in the Eq. (1) (Line 20 in Algorithm 1). The particles are updated, however, based on *sigmoid* function used in BPSO approach (Kennedy and Eberhart, 1997) (Line 21 in Algorithm 1). The obtained equation for updating the particles is shown as follows:

$$x_i^j(t+1)=\begin{cases}1 & rand()<sig(v_i^j(t+1))=\dfrac{1}{1+e^{-v_i^j(t+1)}}\\ 0 & otherwise\end{cases} \quad (12)$$

For the above Eq. (12), it uses the *sigmoid* function as the normalization function and uses the uniform *rand*() number in the range of (0, 1) as the probability that j-th position of a particle will change to either 1 or 0. When the generated *rand*() is less than the $sig(v_i^j(t+1))$, the value of the corresponding j-th position of a particle is set as 1; otherwise, it is set as 0. If the fitness (utility) of a particle is no less than the minimum utility value, the items in this particle is then concerned as a high-utility itemset (Lines 22 and 23 in Algorithm 1). This process is then repeated until the termination criteria is achieved (Lines 17–26 in Algorithm 1). The flowchart of the designed algorithm is shown in Fig. 3.

Algorithm 1: HUIM-BPSO$_{sig}$

Input: D, a quantitative database; $ptable$, a profit table; δ, the minimum utility threshold; M, the number of particles of each iteration.
Output: $HUIs$, a set of high-utility itemsets.

```
for each T_q ∈ D do
    for each i_j ⊆ T_q do
        tu(T_q) = q(i_j, T_q) × ptable(i_j);
calculate twu(i_j) = Σ_{i_j ⊆ T_q} tu(T_q);
TU = Σ_{T_q ∈ D} tu(T_q);
find 1-HTWUIs← {i_j | twu(i_j) ≥ TU × δ};
set k = |1-HTWUIs|;                                   // set particle size
PV ← normalize(1-HTWUIs);                             // normalize the probabilities for 1-HTWUIs
for i ← 1, M particles do
    for j ← 1, k do
        initialize x_i^j(t) = either 1 or 0;          // initial the particles from PV
        initialize v_i^j(t) = rand();                 // initial the velocity of particles in (0, 1)
    if fitness(x_i(t)) ≥ TU × δ then
        HUIs ← GetItem(x_i(t)) ∪ HUIs;                // find a HUI
    find pbest(t) of each M particle;                 // update pbest
    find gbest(t) among M pbest particles and gbest(t);   // update gbest
while termination criteria is not reached do
    for i ← 1, M particles do
        for j ← 1, k do
            update the velocity v_i^j(t + 1) ;        // by equation (1)
            update the x_i^j(t + 1) ;                 // by equation (12)
        if fitness(x_i(t + 1)) ≥ TU × δ then
            HUIs ← GetItem(x_i(t + 1)) ∪ HUIs;        // find a HUI
        find pbest(t + 1) of each M particle;         // update pbest
        find gbest(t + 1) among M pbest particles and gbest(t);   // update gbest
    set t ← t + 1;
return HUIs;
```

4.5. Time complexity analysis

For the designed algorithm, the time complexity is analyzed as $O(N \times M \times m)$, in which N is the number of iterations, M is the number of particles at each iteration and m is the size of each particle. Note that m is equal to the size of 1-HTWUIs. The variables of N and M can be adjusted by user, and the size of 1-HTWUIs is determined by the minimum utility threshold. The time complexity of the state-of-the-art HUPE$_{umu}$-GRAM algorithm is analyzed as $O(N \times M \times (M + m))$. In GA-based approach, it requires $O(M)$ by roulette wheel selection to select two chromosomes for later operations. After that, the GA-based approach requires $O(m)$ to perform the crossover and mutation operations for new chromosomes. Thus, it requires $O(M + m)$ for generating new offsprings and the total time complexity becomes $O(N \times M \times (M + m))$. From the above analysis, our proposed algorithm outperforms the state-of-the-art GA-based HUPE$_{umu}$-GRAM for mining HUIs.

5. An illustrated example

In this section, a database shown in Table 1 and its profit table are used as the running example to illustrate the procedure of the designed algorithm. The minimum utility threshold is set as δ (=30%). Thus, the minimum utility value is calculated as ($TU \times \delta$) ($=386 \times 0.3$)($=115.8$). The quantitative database in Table 1 is first scanned to find the 1-HTWUIs. In this example, the discovered 1-HTWUIs are {b: 264, c: 183, d: 233, e: 361, f: 163}. Thus, the size of each particle in the designed algorithm is set as 5, which is equal to the number of discovered 1-HTWUIs. Based on the transaction-weighted downward closure (TWDC) property, this process can be used to eliminate the combinational process for discovering HUIs. After that, the number of particles in the population is initially set as 10, as well as the number of iterations. Note that those two parameters can be adjusted by users' preferences.

The positions of each particle are corresponding to the items in 1-HTWUIs. The initial particles are generated by the roulette wheel selection from the TWU values of the 1-HTWUIs. Thus, if the TWU value of an item in 1-HTWUIs is high, it has higher probability to be selected in the particle and set its corresponding position (item) as 1. The results are shown in Fig. 4. The velocities of particles in the population are randomly generated in the range of (0, 1). The results are shown in Fig. 5.

After that, the fitness of each particle is then evaluated to find its utility. In this example, the fitness of p_1 is calculated as 135, which is no less than the minimum utility value (=115.8); the itemset (bf) is considered as a high-utility itemset and put it in the set of HUIs. The other particles are also processed in the same way. In this example, only the fitness of p_1 is no less than the minimum utility value; the *pbest* of p_1 is set as (1, 0, 0, 0, 1) as well as the *gbest*. The velocity of each particle is then updated according to the traditional PSO approach shown in Eq. (1). After that, the particles are then updated according to Eq. (12). In this step, the *sigmoid* function is adopted in the designed algorithm to assign whether

the item in the particle is represented either 0 or 1. The above processes for updating the velocities and particles are then repeated until the termination criteria is achieved. After that, the final discovered HUIs with their utilities (fitness values) are {(*b*: 216), (*bd*: 154), (*be*: 222), (*bf*: 135), (*bde*: 166), (*bef*: 131)}.

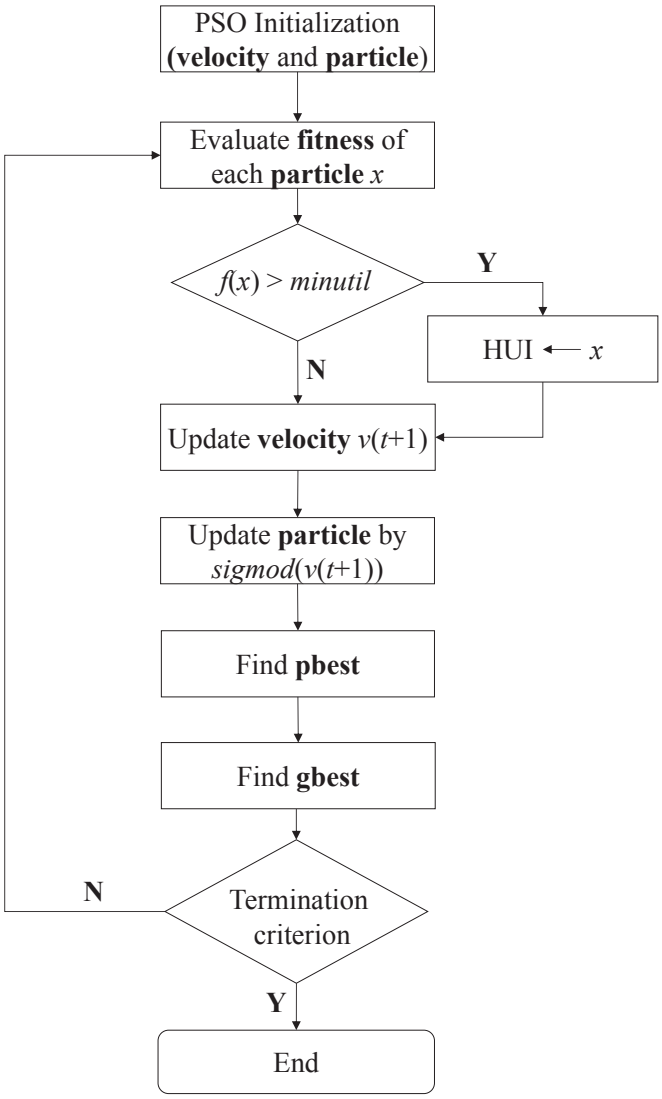

Fig. 3. Flowchart of the designed algorithm.

6. Experimental results

Substantial experiments were conducted to verify the effectiveness and efficiency of the proposed HUIM-BPSO$_{sig}$ algorithm compared to the state-of-the-art evolutionary HUPE$_{umu}$-GRAM algorithm. The original HUPE$_{umu}$-GRAM algorithm requires the HUIs as the initial particles for later evolution process, which needs very large computations to find the initial particles. We have thus improved this approach by adopting our designed pre-processing phase. The algorithms in the experiments were implemented in C++ language, performing on a PC with an Intel Core2 i3–4160 CPU and 4 GB of RAM, running the 64-bit Microsoft Windows 7 operating system. Four dense real-world datasets called chess (Agrawal and Srikant, 1994), mushroom (Agrawal and Srikant, 1994), connect (Agrawal and Srikant, 1994) and accidents (Agrawal and Srikant, 1994) are used in the experiments. The used four datasets in the experiments are widely used in the issues of high-utility itemsets (Fournier-Viger et al., 2014; Fournier-Viger and Zida, 2015; Lin et al., 2011; Liu and Qu, 2012; Tseng et al., 2010; Zida et al., 2015). A simulation model (Liu et al., 2005) was developed to generate the quantities and profit values of items in transactions for all datasets. A log-normal distribution was used to randomly assign quantities in the [1,5] interval, and item profit values in the [1,1000] interval. Parameters and characteristics of the datasets used in the experiments are respectively shown in Tables 2 and 3.

In the conducted experiments for mining HUIs, the termination criteria is set as 10,000 iterations and the population size is set as 20. The algorithms are then compared in terms of execution time, number of HUIs, number of patterns, and the convergence. The results are analyzed as follows.

6.1. Runtime

In the conducted experiments of runtime in four datasets, the state-of-the-art HUPE$_{umu}$-GRAM algorithm of HUIM is compared to the designed HUIM-BPSO$_{sig}$ algorithm. The results are shown in Fig. 6.

From Fig. 6, it can be seen that the proposed PSO-based algorithm has better performance in term of execution time compared to the improved HUPE$_{umu}$-GRAM algorithm w.r.t. variants of minimum utility thresholds in four datasets. For example, the chess dataset shown in Fig. 6(a) with 25% minimum utility

	b	*c*	*d*	*e*	*f*		*b*	*c*	*d*	*e*	*f*
p_1	1	0	0	0	1	p_6	1	0	1	1	0
p_2	0	1	1	0	1	p_7	1	0	0	0	1
p_3	1	1	1	0	0	p_8	1	0	0	0	1
p_4	0	0	1	0	1	p_9	1	1	1	1	0
p_5	0	1	0	1	0	p_{10}	0	1	1	1	1

Fig. 4. Initial particles.

	b	*c*	*d*	*e*	*f*		*b*	*c*	*d*	*e*	*f*
v_1	0.63	0.69	0.50	0.40	0.49	v_6	0.78	0.61	0.85	0.83	0.62
v_2	0.20	0.78	0.61	0.39	0.82	v_7	0.73	0.50	0.92	0.48	0.82
v_3	0.55	0.56	0.35	0.35	0.13	v_8	0.06	0.28	0.58	0.38	0.26
v_4	0.66	0.27	0.69	0.17	0.51	v_9	0.71	0.32	0.03	0.91	0.82
v_5	0.63	0.55	0.52	0.04	0.86	v_{10}	0.72	0.20	0.41	0.35	0.92

Fig. 5. Initial velocities.

threshold, the runtime of improved HUPE$_{umu}$-GRAM algorithm and the proposed HUIM-BPSO$_{sig}$ algorithms were respectively calculated as 10,764, 10,568 s at 10,000 iteration. Overall, the proposed PSO-based algorithm outperforms the HUPE$_{umu}$-GRAM algorithm.

6.2. Number of HUIs

In this section, the number of HUIs is evaluated to show the performance of the compared algorithms. The traditional algorithm of HUIM (Fournier-Viger et al., 2014; Liu et al., 2005) is used to discover the actual and complete HUIs from the quantitative datasets. Experiments are conducted and shown in Fig. 7.

From Fig. 7, it can be seen that the proposed binary PSO-based algorithm can generate nearly the same number of HUIs compared to the state-of-the-art algorithms of HUIM especially when the minimum utility threshold is set higher. The reason is that the size of a particle is associated with the number of 1-HTWUIs, less computations are required when the minimum utility threshold is set higher. For more condense datasets used in the experiments, the number of 1-HTWUIs is close to the number of discovered HUIs under higher minimum utility threshold; the number of generated HUIs of the designed binary PSO-based algorithm is close to the traditional way for mining the complete HUIs. For example, in Fig. 7(a) with the 25% minimum utility threshold, the number of HUIs of the HUPE$_{umu}$-GRAM algorithm, the HUIM-BPSO$_{sig}$ algorithm, and the traditional two-phase algorithm (Liu et al., 2005) are respectively found as 12, 75, and 98. The proposed binary PSO-based algorithm can generate nearly the same number of HUIs with higher minimum utility threshold and outperforms the state-of-the-art HUPE$_{umu}$-GRAM algorithm. The results of Fig. 7 can be summarized in Table 4 to show the accuracy of the discovered HUIs.

From Table 4, it can be observed that the proposed HUIM-BPSO$_{sig}$ has higher accuracy for generating the same number of the HUIs compared to the state-of-the-art HUPE$_{umu}$-GRAM algorithm. When the minimum utility threshold is set higher, the designed algorithm can complete generate the same number of HUIs as the statistical algorithm (Liu et al., 2005) in chess and connect datasets. For the mushroom and accidents datasets, the designed

Table 2
Parameters of used datasets.

#|D|	Total number of transactions
#|I|	Number of distinct items
AvgLen	Average length of transactions
MaxLen	Maximal length of transactions

Table 3
Characteristics of used datasets.

Dataset	#|D|	#|I|	AvgLen	MaxLen
Chess	3196	76	37	37
Mushroom	8124	120	23	23
Connect	67,557	130	43	43
Accidents_10%	34,018	469	34	46

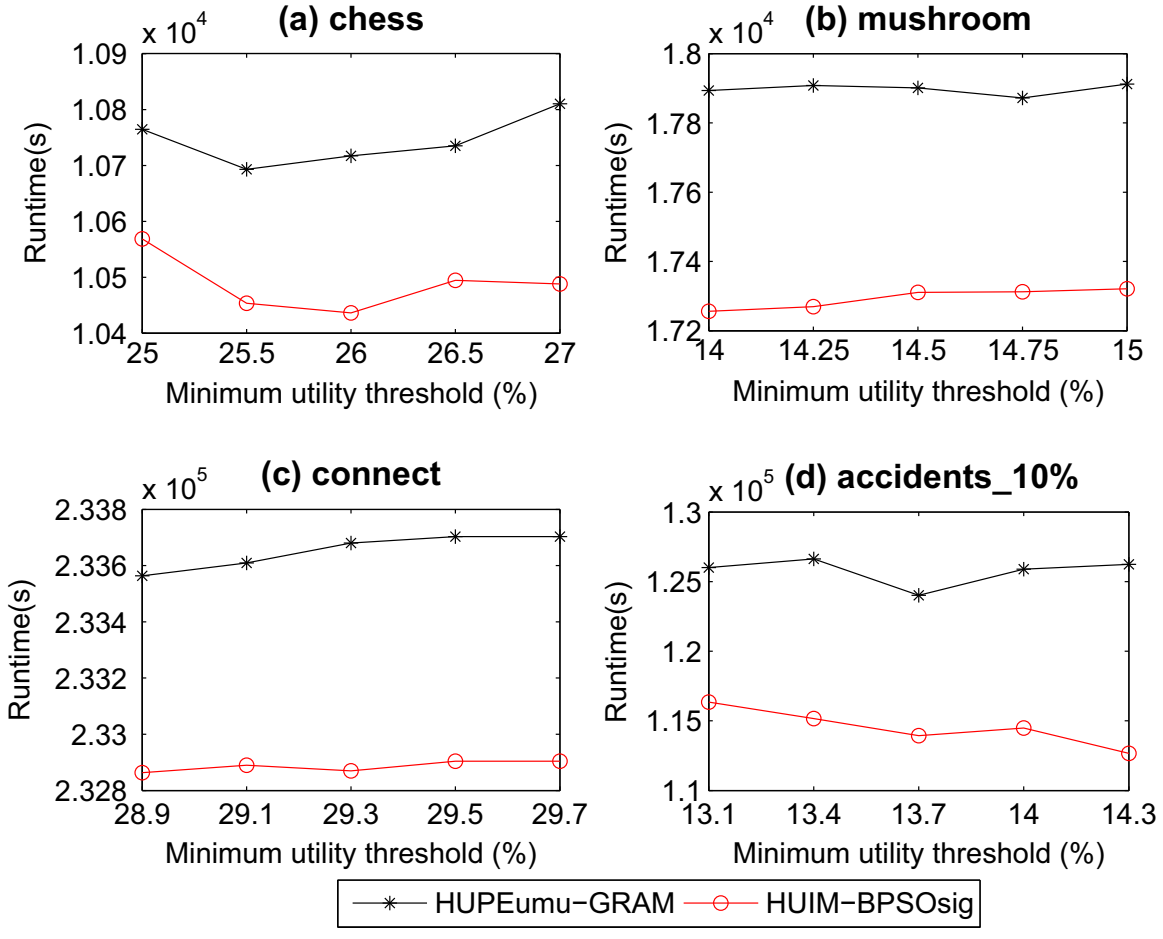

Fig. 6. Runtime w.r.t. variants of minimum utility thresholds.

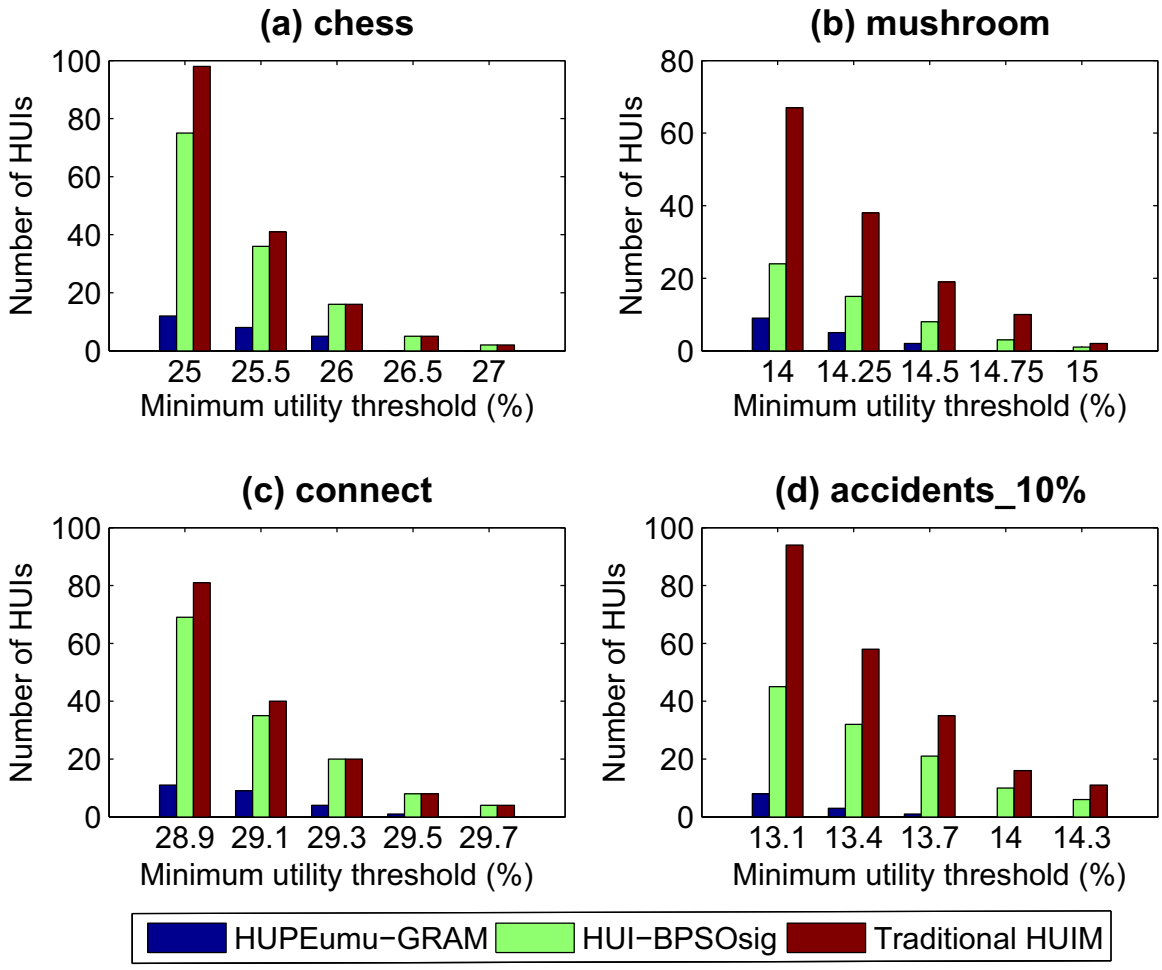

Fig. 7. Number of HUIs w.r.t. variants of minimum utility thresholds.

Table 4
Percentages of discovered HUIs.

Algorithm	Dataset							
	Chess		**Mushroom**		**Connect**		**Accidents**	
HUIM-BPSO$_{sig}$	Thres. (%)	% of HUIs	Thres. (%)	% of HUIs	Thres. (%)	% of HUIs	Thres. (%)	% of HUIs
	25.0	76.53	14.0	35.82	28.9	85.18	13.1	47.87
	25.5	87.80	14.25	39.47	29.1	87.50	13.4	55.17
	26.0	100	14.5	42.10	29.3	100.0	13.7	60.0
	26.5	100	14.75	30.0	29.5	100.0	14.0	62.5
	27.0	100	15.0	50.0	29.7	100.0	14.3	54.54
HUPE$_{umu}$-GRAM	25.0	12.24	14.0	13.43	28.9	13.58	13.1	8.51
	25.5	19.51	14.25	13.15	29.1	22.50	13.4	5.17
	26.0	31.25	14.5	10.52	29.3	20.0	13.7	2.85
	26.5	0.0	14.75	0.0	29.5	12.50	14.0	0.0
	27.0	0.0	15.0	0.0	29.7	0.0	14.3	0.0

HUIM-BPSO$_{sig}$ algorithm can generate more than 50% of HUIs but the HUPE$_{umu}$-GRAM algorithm could not generate any HUIs when the minimum utility threshold is set higher. Since the valid HUIs are retrieved from the 1-HTWUIs, it has higher possibility to generate more HUIs when the minimum utility threshold is set lower for the HUPE$_{umu}$-GRAM algorithm. As the increasing of the minimum utility threshold but the number of 1-HTWUIs remains steady, the state-of-the-art HUPE$_{umu}$-GRAM algorithm cannot produce the valid HUIs with the same number of 1-HTWUIs, which can be observed from Fig. 7 (a)–(d) and will be observed from Fig. 8 in later section. Overall, when the number of iterations is increased, the proposed HUIM-BPSO$_{sig}$ algorithm can achieve higher accuracy for generating the same number of HUIs.

6.3. Number of patterns

In this section, the number of HUIs mined by traditional HUIM algorithm (Fournier-Viger et al., 2014; Liu et al., 2005) and the

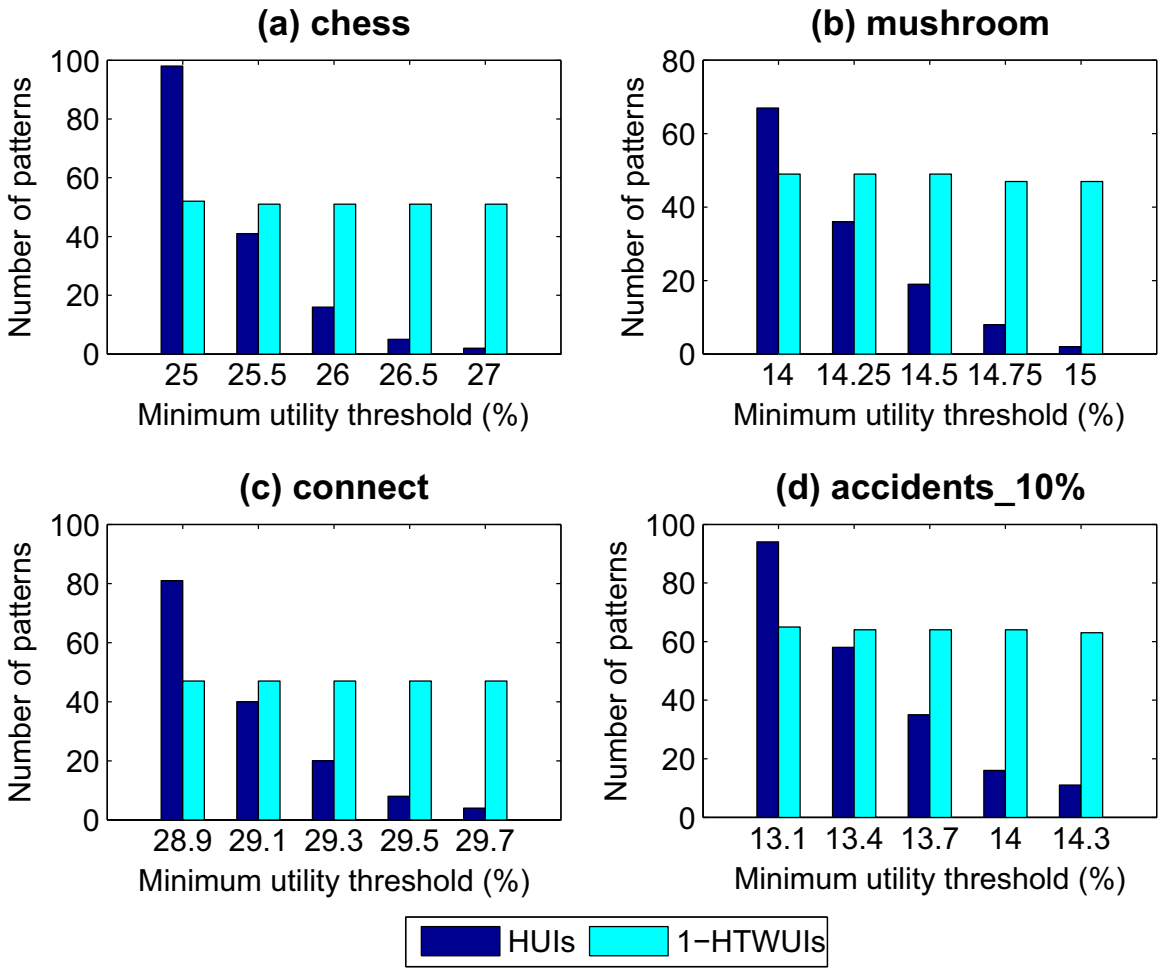

Fig. 8. Number of patterns w.r.t. variants of minimum utility threshold.

number of 1-HTWUIs w.r.t. variants of minimum utility thresholds is evaluated. The difference between the HUIs and 1-HTWUIs is that the number of 1-HTWUIs is used to discover the valid HUIs in the datasets. The valid combination of 1-HTWUIs is then formed as a HUI if its fitness (utility) is no less than the minimum utility threshold. Thus, when the minimum utility threshold is set lower, the number of the HUIs is larger than that of the 1-HTWUIs. However, when the minimum utility threshold is set higher with the same number of 1-HTWUIs, fewer HUIs is thus discovered. Based on this evaluation, it is easy to evaluate the gap between the HUIs and 1-HTWUIs. The results are shown in Fig. 8.

From Fig. 8, it can be seen that the difference between the number of HUIs and the number of 1-HTWUIs is very large in all datasets. For example, the number of HUIs and the number of 1-HTWUIs in the chess dataset with minimum support threshold 26% are respectively 16 and 51. It indicates that the size of chromosome is set as 51 in the designed algorithm. In the designed HUIM-BPSO$_{sig}$ algorithm, the valid HUIs are retrieved from the permutations of 1-HTWUIs. When the minimum utility threshold is not varying changed, the size of the discovered 1-HTWUIs remains steady. Besides, the value of the minimum utility threshold greatly affects the size of the HUIs. The reason is that when the size of 1-HTWUIs is nearly the same w.r.t. variants of minimum utility threshold, lower minimum utility threshold has higher possibility to generate the valid combinations from the 1-HTWUIs. When the minimum utility threshold is slightly increased, the number of 1-HTWUIs steadily remains but the valid combinations from 1-HTWUIs cannot be easily discovered. From the observed results in Fig. 8, we can conclude that the evolution process of the designed algorithm can efficiently find the valid HUIs from a very large combinations.

6.4. Convergence

In this section, the convergence of the proposed HUIM-BPSO$_{sig}$ algorithm is evaluated compared to the improved HUPE$_{umu}$-GRAM algorithm and the number of discovered HUIs of HUIM in different datasets. The results are shown in Fig. 9.

From Fig. 9, it can be observed that the speed of convergence of the improved HUPE$_{umu}$-GRAM is slower than that of the designed HUIM-BPSO$_{sig}$ algorithm. For example in Fig. 9(a), the number of discovered HUIs of three algorithms are respectively found as 0, 19 and 98 while the number of iteration is achieved at 1000. The HUPE$_{umu}$-GRAM algorithm suffers the serious combinational problem in the evolution process. Overall, the proposed HUIM-BPSO$_{sig}$ algorithm can be faster converged than the HUPE$_{umu}$-GRAM algorithm.

7. Conclusions

High utility itemset mining (HUIM) has emerging as an important topic in recent years since it can reveal the highly profitable products instead of the frequent itemset mining (FIM). Several algorithms have been proposed to efficiently mine the high-utility itemsets (HUIs) from the quantitative databases and most of them

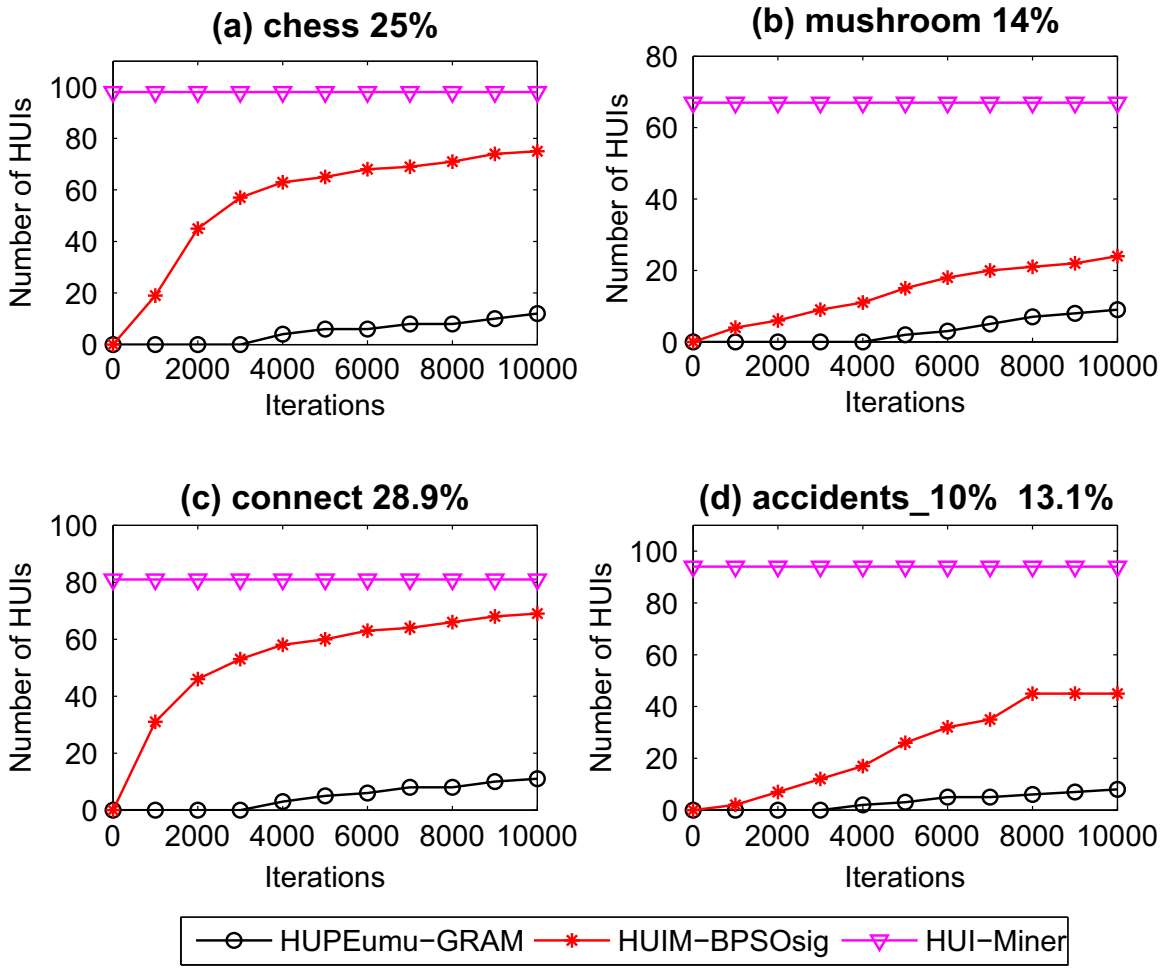

Fig. 9. Convergence w.r.t. variants of iterations.

applied the statistical analysis. This process may take very large computations when the number of distinct items or the size of the database is very large. In the past, the GA-based approach namely HUPE$_{umu}$-GRAM was proposed to mine HUIs based on genetic algorithm. It suffers the combinational problem of the generated itemsets and requires several parameters in the evolution process. In this paper, a particle swarm optimization (PSO)-based algorithm namely HUIM-BPSO$_{sig}$ is proposed to efficiently mine HUIs. The proposed algorithm first adopts the TWU model to find the number of high-transaction-weighted utilization 1-itemsets (1-HTWUIs) as the particle size, which can greatly reduce the combinational problem in the evolution process. The sigmoid function of the binary PSO (BPSO) is adopted in the developed HUIM-BPSO$_{sig}$ to mine HUIs. From the conducted experiments, the proposed approach has better results than the GA-based algorithm in terms of execution time, the ability for discovering HUIs, and the convergence.

Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (NSFC) under Grant no. 61503092.

References

Agrawal, R., Srikant, R., 1994. Fast algorithms for mining association rules in large databases. In: Proceedings of the International Conference on Very Large Data Bases, pp. 487–499.

Cattral, R., Oppacher, F., Graham, K.J.L., 2009. Techniques for evolutionary rule discovery in data mining. IEEE Congr. Evolut. Comput., 1737–1744.

Chan, R., Yang, Q., Shen, Y.D., 2003. Minging high utility itemsets. In: Proceedings of the IEEE International Conference on Data Mining, pp. 19–26.

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.

Fournier-Viger, P., Wu, C.W., Tseng, V.S., 2014. Novel concise representations of high utility itemsets using generator patterns. Adv. Data Min. Appl. 8933, 30–43.

Fournier-Viger, P., Wu, C.W., Zida, S., Tseng, V.S., 2014. FHM: faster high-utility itemset mining using estimated utility co-occurrence pruning: faster high-utility itemset mining using estimated utility co-occurrence pruning. Found. Intell. Syst. 8502, 83–92.

Fournier-Viger, P., Zida, S., 2015. FOSHU: faster on-shelf high utility itemsets mining with or without negative unit profit. In: Proceedings of the ACM Symposium on Applied Computing, pp. 857–864.

Holland, J., 1975. Adaptation in natural and artificial systems. MIT Press, Cambridge, MA.

Kannimuthu, S., Premalatha, K., 2014. Discovery of high utility itemsets using genetic algorithm with ranked mutation. Appl. Artif. Intell. 28 (4), 337–359.

Kennedy, J., Eberhart, R., 1995. Particle swarm optimization. In: Proceedings of the IEEE International Conference on Neural Networks, vol. 4, pp. 1942–1948.

Kennedy, J., Eberhart, R., 1997. A discrete binary version of particle swarm algorithm. In: Proceedings of the 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.

Lan, G.C., Hong, T.P., Tseng, V.S., 2013. An efficient projection-based indexing approach for mining high utility itemsets. Knowl. Inf. Syst. 38 (1), 85–107.

Lin, C.W., Hong, T.P., Lu, W.H., 2011. An effective tree structure for mining high utility itemsets. Expert Syst. Appl. 38 (6), 7419–7424.

Liu, Y., Liao, W.K., Choudhary, A., 2005. A two-phase algorithm for fast discovery of high utility itemsets. Lect. Notes Comput. Sci., 689–695.

Liu, M., Qu, J., 2012. Mining high utility itemsets without candidate generation. In: Proceedings of the ACM International Conference on Information and Knowledge Management, pp. 55–64.

Martnez-Ballesteros, M., Martnez-lvarez, F., Riquelme, J.C., 2010. Mining quantitative association rules based on evolutionary computation and its application to

atmospheric pollution. Integr. Comput.-Aided Eng. 17 (3), 227–242.
Menhas, M.I., Fei, M., Wang, L., Fu, X., 2011. A novel hybrid binary pso algorithm. Lecture Notes in Computer Science, vol. 6728, pp. 93–100.
Nouaouria, N., Boukadouma, M., Proulx, R., 2013. Particle swarm classi cation: a survey and positioning. Pattern Recognit. 46 (7), 2028–20440.
Pears. R., Koh. Y.S., 2012. Weighted association rule mining using particle swarm pptimization. Lecture Notes in Computer Science, vol. 7104, pp. 327–338.
Salleb-Aouissi, A., Vrain, C., Nortet, C., 2007. QuantMiner: a genetic algorithm for mining quantitative association rules. In: Proceedings of the International Joint Conference on Artifical Intelligence, pp. 1035–1040.
Sarath, K.N.V.D., Ravi, V., 2013. Association rule mining using binary particle swarm optimization. Eng. Appl. Artif. Intell. 26, 1832–1840.
SPMF, 2015. An Open-Source Data Mining Library. ⟨http://www.philippe-fournier-viger.com/spmf/index.php⟩.
Tseng, V.S., Wu, C.W., Shie, B.E., Yu, P.S., 2010. UP-growth: An efficient algorithm for high utility itemset mining. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 253–262.
Yao, H., Hamilton, H.J., 2006. Mining itemset utilities from transaction databases. Data Knowl. Eng. 59 (3), 603–626.
Yao, H., Hamilton, H.J., Butz, C.J., 2004. A foundational approach to mining itemset utilities from databases. In: Proceedings of the SIAM International Conference on Data Mining, pp. 211–225.
Zida, S., Fournier-Viger, P., Lin, C.W., Wu, C.W., Tseng, V.S., 2015. EFIM: a highly efficient algorithm for high-utility itemset mining. In: Proceedings of the Mexican International Conference on Artificial Intelligence.
Zihayat, M., An, A., 2014. Mining top-k high utility patterns over data streams. Inf. Sci. 285, 138–161.