Mining High Utility Itemsets Using Ant Colony Optimization

Wei Song(✉) and Jiakai Nan

School of Information Science and Technology, North China University of Technology, Beijing 100144, China
songwei@ncut.edu.cn

Abstract. Mining high utility itemsets (HUIs) is one of the most important research topics in data mining. Algorithms based on evolutionary computation are attracting increasing attention, as they have the advantage of avoiding the combinatorial explosion of the HUI search space. In this paper, we propose an HUI mining algorithm based on ant colony optimization (HUIM-ACO) that generates new candidate in a constructive way. In the HUIM-ACO algorithm, we use a search path to represent candidate HUIs. Each search path is initialized proportionally, and incremented by one item per step. A pheromone matrix is proposed to store the pheromone values of a pair of items, allowing the local and global values to be updated. Furthermore, an ant-path quad is designed for the efficient enumeration of HUIs. Experimental comparisons with three state-of-the-art HUI mining algorithms show that HUIM-ACO can discover more HUIs in a shorter period of time.

Keywords: Data mining · High utility itemset · Ant colony optimization · Pheromone matrix · Ant-path quad

1 Introduction

High utility itemset mining (HUIM) has received increased interest in recent years [3, 11]. As an extension of frequent itemsets (FIs), high utility itemsets (HUIs) focus on both the quantity and profit associated with sets of items. For HUIM problems, each item has its own profit and can occur multiple times in one transaction. The utility of an itemset is calculated by summing the product of the item's profit and its number of occurrences in each relevant transaction. HUIM aims to discover all itemsets with utilities no lower than a user-specified threshold.

Although several algorithms [7, 11] have been proposed, enumerating all HUIs exactly produces a very large search space when the number of items or size of the database is large. Furthermore, for application fields such as recommender systems, it is not necessary to use all HUIs [12].

To solve the two abovementioned problems, evolutionary computation approaches can be used to discover most HUIs within an acceptable time. Kannimuthu and Premalatha proposed two HUIM algorithms based on genetic algorithms (GAs) with and without setting the minimum utility threshold, respectively [4]. The main problem with these two algorithms is premature convergence, which results in few HUIs and a long execution time. Lin et al. proposed two HUIM algorithms based on particle swarm optimization (PSO) [5, 6]. Of these two PSO-based algorithms, HUIM-BPSO [6] provides better performance by using a novel tree structure. Song and Huang modeled the HUIM problem using an artificial bee colony (ABC) algorithm, and discovered HUIs by sequentially optimizing employed bees, onlooker bees, and scout bees in an iterative process [8].

Different from the above methods, a general framework called Bio-HUIF has been proposed [9]. In Bio-HUIF, all discovered HUIs, rather than those with higher utility, are proportionally maintained in the next generation. Thus, the population is diverse, which is more suitable for the problem of HUIM.

In this paper, we study the problem of HUIM from the perspective of ant colony optimization (ACO) [1], and propose a novel HUIM-ACO algorithm. We describe the modeling of HUIM based on ACO, including the search path and pheromone. Using a bitmap structure, we present the HUIM-ACO algorithm in detail and demonstrate its superiority through comparisons with three related algorithms in terms of efficiency and number of results.

2 Preliminaries

2.1 Problem of HUIM

Let $\boldsymbol{I} = \{i_1, i_2, \ldots, i_m\}$ be a finite set of items and $X \subseteq \boldsymbol{I}$ be an *itemset*; an itemset containing k items is called a k-itemset. Let $\boldsymbol{D} = \{T_1, T_2, \ldots, T_n\}$ be a transaction database. Each transaction $T_i \in \boldsymbol{D}$ is a subset of $\boldsymbol{I}$.

The *internal utility* $q(i_p, T_d)$ represents the quantity of item i_p in transaction T_d. The *external utility* $p(i_p)$ is the unit profit value of item i_p. The *utility* of item i_p in transaction T_d is defined as $u(i_p, T_d) = p(i_p) \times q(i_p, T_d)$. The utility of itemset X in transaction T_d is defined as $u(X, T_d) = \sum_{i_p \in X \wedge X \subseteq T_d} u(i_p, T_d)$. The utility of itemset X in $\boldsymbol{D}$ is defined as $u(X) = \sum_{X \subseteq T_d \wedge T_d \in \boldsymbol{D}} u(X, T_d)$. The transaction utility of transaction T_d is defined as $TU(T_d) = u(T_d, T_d)$. The *minimum utility threshold* δ is given as a percentage of the total transaction utility values of the database, whereas the *minimum utility value* is defined as $min\_util = \delta \times \sum_{T_d \in \boldsymbol{D}} TU(T_d)$. An itemset X is called a *high utility itemset* if $u(X) \geq min\_util$.

The *transaction-weighted utilization* (TWU) of itemset X [7] is the sum of the transaction utilities of all transactions containing X, which is defined as $TWU(X) = \sum_{X \subseteq T_d \wedge T_d \in \boldsymbol{D}} TU(T_d)$. X is a high transaction-weighted utilization itemset (HTWUI) if $TWU(X) \geq min\_util$; otherwise, X is a low transaction-weighted utilization itemset (LTWUI). An HTWUI/LTWUI with k items is called a k-HTWUI/k-LTWUI.

2.2 Ant Colony Optimization

ACO [1] is an evolutionary algorithm inspired by the foraging behavior of ants, and is well-suited to combinatorial problems. As HUIM is, in essence, a combinatorial problem, ACO is used in this study for the efficient discovery of HUIs.

Using ACO, the problem is tackled by simulating a number of artificial ants moving on a graph, in which each vertex represents a place and each edge represents a connection between two places. A variable called a pheromone is associated with each edge and can be read and modified by the ants.

ACO is an iterative algorithm executed by SN ants. At each iteration, each ant A_t ($1 \leq t \leq SN$) builds a solution by walking from the current vertex v_c to vertex v_n on the graph, such that v_n has not been visited by A_t. During the walking process, the next vertex v_n is selected with a probability that is proportional to the pheromone associated with edge (v_c, v_n). The pheromone value of edge (v_c, v_n) is then updated to bias A_t in future iterations to construct solutions similar to the best ones previously constructed. When all ants find the next vertex and update the pheromone values to their current node, the global update step updates the pheromone value of the best tour. Different from the update provided by each ant, the pheromone values in all of the tours passed by the ants within this iteration are modified in the global update step. The process of walking to the next node, local pheromone update, and global pheromone update is repeated until some termination condition is satisfied.

3 Modeling HUIM Using ACO

3.1 Pheromone Matrix

We propose the *pheromone matrix* to record the pheromone values of a pair of items.

Definition 1. Let H be the number of 1-HTWUIs and let the pheromone matrix ***PM*** be an $H \times H$ matrix. Each entry $e_{j,k}$ of ***PM*** corresponds to the pheromone value of the 2-itemset $i_j i_k$. Entries along the principal diagonal, corresponding to pheromone values of an item with itself, are set to zero. For other entries where $i_j \neq i_k$, $e_{j,k}$ is initialized as:

$$e_{j,k} = \begin{cases} u(i_j i_k)/2, \text{ if } & TWU(i_j i_k) \geq min\_util \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

We can see from Eq. 1 that $e_{j,k} = e_{k,j}$, so ***PM*** is a symmetric matrix and the entries of ***PM*** are symmetric with respect to the principal diagonal. Thus, we need only store half of ***PM*** with respect to the principal diagonal.

For the local pheromone update stage, each entry of ***PM*** is modified as:

$$e_{j,k} = e_{j,k} + u(i_j i_k)^{\alpha}/\mathcal{L} \tag{2}$$

where α ($\alpha > 0$) is the *local influence factor* for adjusting the contribution of the utility value in the pheromone and $\mathcal{L}(\mathcal{L} > 1)$ is the *evaporation factor* for diluting the pheromone density.

In the global pheromone update stage, we use a different strategy from the typical ACO. Rather than updating all paths within one iteration, once a new HUI is discovered, our algorithm only updates entries in this path. Specifically, for each entry $e_{j,k}$ in the path generating an HUI, the following modification is executed:

$$e_{j,k} = e_{j,k} + TWU(i_j i_k)^{\beta}/\mathcal{L} \tag{3}$$

where β ($\beta > 0$) is the *global influence factor* for adjusting the contribution of the TWU value in the pheromone.

For Eqs. 2 and 3, the utility value is used to update the local pheromone, and the TWU value is used to update the global pheromone. This is mainly because the granularity of the local update is different from that of the global update. As the local update is only executed between two adjacent vertices, the exact utility value is used to update the pheromone. For the global update, all pairs of adjacent vertices within the current path are updated, so we use TWU, the upper approximation of the path, to update the pheromone.

3.2 Initialization of Search Path

When an ant starts its search path, it selects the first item i_j proportionally, determined by:

$$P_j = \frac{TWU\left(i_j\right)^{\gamma}/\mathcal{L}}{\sum_{k=1}^{H} TWU\left(i_k\right)^{\gamma}/\mathcal{L}} \tag{4}$$

where γ ($\gamma > 0$) is the *transaction influence factor* for adjusting the contribution of the TWU value in the first item selection.

For each ant A_t, an *ant-path quad* is represented as $AP_t = < i_c, Pa, Succ, TS >$, where i_c is the current item being visited by A_t, *Pa* is the set of items that have been visited by A_t, *Succ* is the set of items have not been visited by A_t, and *TS* is the set of transactions containing all items in *Pa.*

3.3 State Transition Rule

The state transition rule is used by an ant to probabilistically select its next vertex. Let A_t be an ant. The next vertex representing item i_{next} that A_t will visit is determined by:

$$i_{next} = \begin{cases} arg\,max_{i_j \in AP_t.Succ}\left\{e_{c,j}\right\}, & \text{if } rand \geq \tau \\ i_k \in AP_t.Succ \text{ with a probability } P_{c,k}, & \text{otherwise} \end{cases} \tag{5}$$

where $e_{c,j}$ is the pheromone value between $AP_t.i_c$ and i_j in the pheromone matrix, *r and* is a random number uniformly distributed between 0 and 1, τ ($0 \leq \tau \leq 1$) is the *selection threshold* predefined by users, and the probability from item $AP_t.i_c$ to item i_k is defined as:

$$P_{c,k} = \frac{e_{c,k}}{\sum_{i_l \in AP_t.Succ} e_{c,l}} \tag{6}$$

4 Mining HUIs Using ACO

4.1 Bitmap Item Information Representation

We use a bitmap, an effective representation of item information in FI mining and HUI mining algorithms [10], in HUIM-ACO to identify transactions containing the target itemsets. Specifically, HUIM-ACO uses a *bitmap cover* representation for itemsets. In a bitmap cover, one bit represents each transaction in the database. If item i appears in transaction T_j, then bit j of the bitmap cover for item i is set to one; otherwise, the bit is set to zero. This naturally extends to itemsets. If X be an itemset, $Bit(X)$ corresponds to the bitmap cover that represents the transaction set for the itemset X. Let X and Y be two itemsets. Then, $Bit(X \cup Y)$ can be computed as $Bit(X) \cap Bit(Y)$, i.e., the bitwise-AND of $Bit(X)$ and $Bit(Y)$.

4.2 Proposed Algorithm

Algorithm 1 describes our proposed HUIM-ACO algorithm.

```
Algorithm 1  HUIM-ACO
Input        Transaction database D, population size SN, minimum utility value
             min_util, local influence factor α, global influence factor β, evapora-
             tion factor ∠, maximal number of iterations max_iter
Output       HUIs
1   Initialization( );
2   do
3     $H_{new} = \varnothing$;
4     $iter = 1$;
5     while ($iter \le max\_iter$) do
6       for each ant $A_t$ do
7         if ($iter$==1) then
8           Initialize the first item visited by $A_t$ using Eq.4;
9           Initialize the ant-path quad $AP_t$;
10        end if
11        if ($AP_t.Succ \neq \varnothing$) then
12          Select the next item $i_{t\text{-}next}$ from the current item $AP_t.i_c$ using Eq.5;
13          if ($TWU(AP_t.Pa \cup i_{t\text{-}next}) \ge min\_util$ AND ($AP_t.Pa \cup i_{t\text{-}next} \notin SH$)) then
14            LocalUpdate($AP_t$, $i_{t\text{-}next}$);
15            if ($u(AP_t.Pa)) \ge min\_util$ then
16              $AP_t.Pa \to H_{new}$;
17              GlobalUpdate($AP_t$);
18            end if
19          end if
20        end if
21      end for
22      $iter$ ++;
23    end while
24    $H_{new} \to SH$;
25  while ($H_{new} \neq \varnothing$);
26  Output all HUIs in $SH$.
```

In Algorithm 1, the procedure Initialization(), described in Algorithm 2, is called in Step 1. The main loop from Steps 2 to 25 mines the HUIs iteratively until no new HUIs are discovered by any ants. Step 3 initializes H_{new}, the set of newly discovered HUIs within one iteration, as the empty set. Step 4 then initializes the iteration number to 1. The loop Steps 5–23 processes each individual ant with the condition that the search time has not exceeded the maximal threshold. The loop from Step 6 to Step 21 describes the actions of an ant updating the pheromone values and discovering HUIs in one iteration. For the first iteration of one ant, the first selected item and the ant-path quad are initialized in Steps 7 to 10. If there are items to explore, the next item to be visited is determined in Step 12. If the TWU value of the current path adding the newly determined next item is no lower than the minimum utility threshold, and the merging result is not discovered as an HUI before, Step 14 calls the procedure LocalUpdate() described in Algorithm 3. Similarly, when the newly formed path is an HUI, the procedure GlobalUpdate(), described in Algorithm 4, is called. The transaction set *TS* in the ant-path quad ensures that both the TWU value and utility value can be efficiently determined by resorting to transactions in *TS*, rather than the entire database. The search time is then incremented by one. In Step 24, the newly discovered HUIs within that iteration are added to *SH*, which is the set of all discovered HUIs. Finally, Step 26 outputs all mining results.

```
Algorithm 2   Initialization( )
1             Scan database **D** once, and delete 1-LTWUIs;
2             Represent the reorganized database as a bitmap;
3             Initialize each element of pheromone matrix **PM** using Eq. 1;
4             $SH=\emptyset$.
```

In Algorithm 2, the 1-LTWUIs are first deleted in Step 1. Step 2 then constructs the bitmap representation of the database. The pheromone value of each pair of 1-HTWUIs is initialized in Step 3. Finally, the set of all HUIs *SH* is initialized to the empty set in Step 4.

```
Algorithm 3   LocalUpdate($AP_t$, $i_{t\text{-}next}$)
1             Update local pheromone value on $e_{c,t\text{-}next}$ using Eq. 2;
2             $i_{t\text{-}next} \rightarrow AP_t.Pa$;
3             $AP_t.Succ = AP_t.Succ \setminus i_{t\text{-}next}$;
4             $AP_t.i_c = i_{t\text{-}next}$;
5             Update $AP_t.TS$.
```

In Algorithm 3, the pheromone value corresponding to $AP_t.i_c$ and $i_{t\text{-}next}$ are updated in Step 1. Step 2 adds $i_{t\text{-}next}$ to the path of the current ant. Then, $i_{t\text{-}next}$ is deleted from the set of items that have not been visited in Step 3. Step 4 uses $i_{t\text{-}next}$ as the new current item. Finally, the set of transactions containing the current path is updated in Step 5.

```
Algorithm 4   GlobalUpdate($AP_t$)
1             for $l$=1 to $|AP_t.Pa| - 1$ do
2                 Update pheromone value $e_{l,\,l+1}$ using Eq.3;
3             end for
```

In Algorithm 4, the pheromone values of each pair of items in the current path are updated sequentially from the first pair to the ($|AP_t.Pa|$ - 1)-*th* pair, where $|AP_t.Pa|$ is the number of items in the current path $AP_t.Pa$.

5 Performance Evaluation

In this section, we evaluate the performance of our HUIM-ACO algorithm and compare it with that of HUIM-BPSO [6], Bio-HUIF-GA [9], and Bio-HUIF-PSO [9]. The source code of the three comparison algorithms was downloaded from the SPMF data mining library [2].

5.1 Test Environment and Datasets

We conducted the experiments on a computer with a quad-core 2.50 GHz CPU and 8 GB memory running 64-bit Microsoft Windows 10. We wrote our programs in Java. We used four real datasets, downloaded from the SPMF data mining library [2], to evaluate the performance of the algorithms. Table 1 summarizes the characteristics of the datasets used in the experiments.

Table 1. Characteristics of datasets used for experimental evaluations

Datasets	Avg. trans. length	No. of items	No. of trans
Chess	37	76	3,196
Mushroom	23	119	8,124
Connect	43	130	67,557
Accident	33.8	468	340,183

In our algorithm, the local influence factor α, global influence factor β, transaction influence factor γ, evaporation factor $\mathcal{L}$, selection threshold τ, population size *SN*, and maximal number of iterations *max_iter* must be set to appropriate values. We first outline the approximate range of these parameters, and then determine their optimal values by progressive refinement. The final values of these parameters used in the experiments are listed in Table 2.

Table 2. Parameter settings for the four datasets

Datasets	α	β	γ	$\mathcal{L}$	τ	*SN*	*max_iter*
Chess	0.10	3.00	5.00	1000	0.80	2000	25
Mushroom	0.60	1.90	3.60	10000	0.90	2000	25
Connect	1.00	2.10	4.00	10000	0.95	5000	30
Accident	0.90	2.10	4.00	10000	0.93	2000	25

5.2 Runtime

First, we demonstrate the efficiency of the proposed algorithm. A comparison of the runtimes of each algorithm with each of the four datasets is shown in Fig. 1. When measuring the runtime, we varied the minimum utility threshold for each dataset.

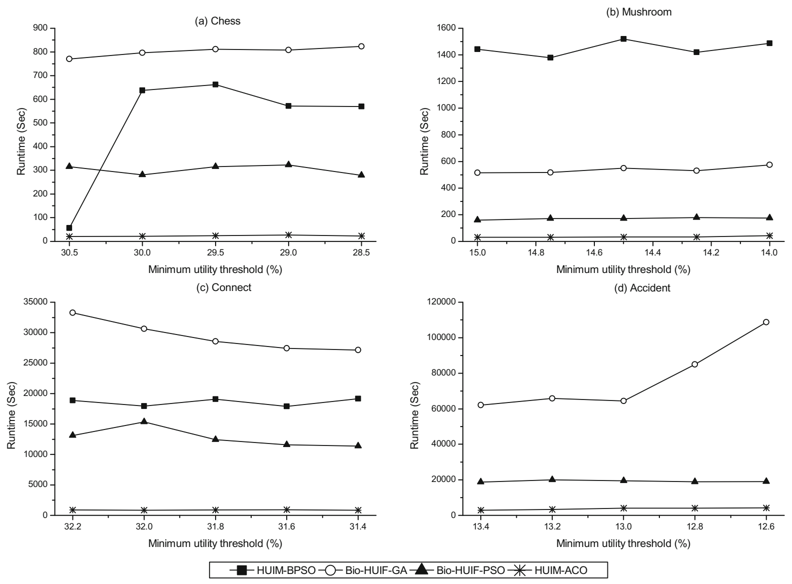

Fig. 1. Execution times for the four datasets

For the Accident dataset, HUIM-BPSO did not return any results, even after four days; thus, we have not plotted the results for this algorithm in Fig. 1(d). We can see from Fig. 1 that the proposed HUIM-ACO is always one order of magnitude faster than HUIM-BPSO and Bio-HUIF-GA. For the Chess and Connect datasets, HUIM-ACO is one order of magnitude faster than Bio-HUIF-PSO. It is interesting to note that the execution time of HUIM-ACO is steady, and is almost unaffected by the change in the minimum utility thresholds.

5.3 Number of Discovered HUIs

Because bio-inspired HUIM algorithms cannot ensure the discovery of all itemsets within a certain number of cycles, we compared the number of discovered HUIs among the various bio-inspired algorithms. For the same reasons stated in Sect. 5.2, we do not plot the results of HUIM-BPSO for the Accident dataset.

As we can see from Fig. 2, the proposed HUIM-ACO discovers more HUIs than the other three algorithms, on average, for the Chess and Mushroom datasets. For the Connect dataset, Bio-HUIF-GA and Bio-HUIF-PSO can discover slightly more HUIs than HUIM-ACO. For the Accident dataset, although the number of HUIs discovered by HUIF-GA and Bio-HUIF-PSO is higher than for HUIM-ACO, the proposed method can find more HUIs than the other two algorithms when the minimum utility threshold is less than 12.8%.

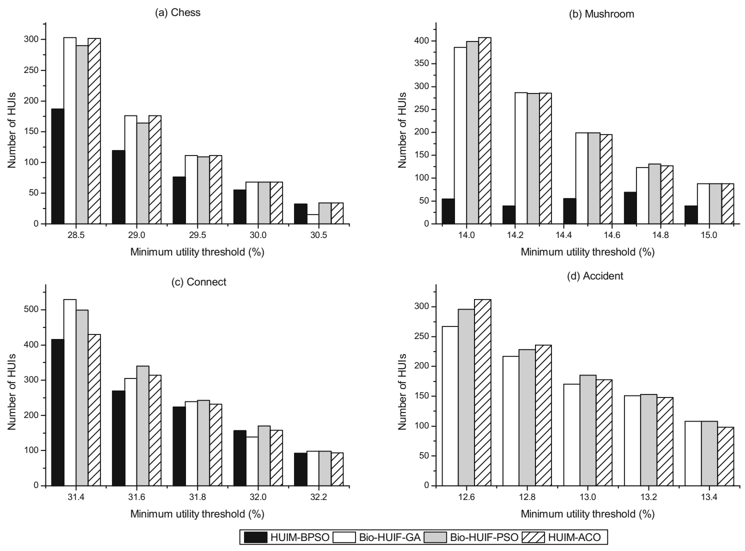

Fig. 2. Number of discovered HUIs

6 Conclusions

This paper describes a heuristic approach to tackling HUIM problems by proposing a novel ACO-based algorithm called HUIM-ACO. In the HUIM-ACO algorithm, a pheromone matrix is proposed for storing the pheromone values of pairs of items, and the ant-path quad is used to accelerate HUI discovery and pheromone value updates. Furthermore, the initialization and transition of the search path are defined for the problem of HUIM. Experimental results show that HUIM-ACO can discover more HUIs than other bio-inspired algorithms with high efficiency.

Acknowledgments. This work was partially supported by the National Natural Science Foundation of China (61977001), the Great Wall Scholar Program (CIT&TCD20190305), and Beijing Urban Governance Research Center.

References

1. Dorigo, M., Birattari, M., Stützle, T.: Ant colony optimization: artificial ants as a computational intelligence technique. IEEE Comput. Intell. M. **1**(4), 28–39 (2006)

2. Fournier-Viger, P., et al.: The SPMF open-source data mining library version 2. In: Berendt, B., et al. (eds.) ECML PKDD 2016. LNCS, vol. 9853, pp. 36–40. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-46131-1_8
3. Fournier-Viger, P., Lin, J.C.W., Nkambou, R., Vo, B., Tseng, V.S.: High-Utility Pattern Mining. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-04921-8
4. Kannimuthu, S., Premalatha, K.: Discovery of high utility itemsets using genetic algorithm with ranked mutation. Appl. Artif. Intell. **28**(4), 337–359 (2014)
5. Lin, J.C.-W., et al.: Mining high-utility itemsets based on particle swarm optimization. Eng. Appl. Artif. Intel. **55**, 320–330 (2016)
6. Lin, J.C.-W., Yang, L., Fournier-Viger, P., Hong, T.-P., Voznak, M.: A binary PSO ap-proach to mine high-utility itemsets. Soft Comput. **21**(17), 5103–5121 (2017)
7. Liu, Y., Liao, W.-K., Choudhary, A.N.: A two-phase algorithm for fast discovery of high utility itemsets. In: Ho, T.B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS, vol. 3518, pp. 689–695. Springer, Heidelberg (2005). https://doi.org/10.1007/11430919_79
8. Song, W., Huang, C.: Discovering high utility itemsets based on the artificial bee colony algorithm. In: Phung, D., Tseng, V., Webb, G., Ho, B., Ganji, M., Rashidi, L. (eds.) PAKDD 2018. LNCS, vol. 10939, pp. 3–14. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93040-4_1
9. Song, W., Huang, C.: Mining high utility itemsets using bio-inspired algorithms: a diverse optimal value framework. IEEE Access **6**, 19568–19582 (2018)
10. Song, W., Liu, Y., Li, J.: BAHUI: fast and memory efficient mining of high utility itemsets based on bitmap. Int. J. Data Warehous. **10**(1), 1–15 (2014)
11. Song, W., Zhang, Z., Li, J.: A high utility itemset mining algorithm based on subsume index. Knowl. Inf. Syst. **49**(1), 315–340 (2016)
12. Yang, R., Xu, M., Jones, P., Samatova, N.: Real time utility-based recommendation for revenue optimization via an adaptive online top-k high utility itemsets mining model. In: Proceedings of the 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery, pp. 1859–1866 (2017)