High Utility Itemset Mining

Atrey Dad, Bharat Didwania, Gaurav Sachdeva, Gaurav Singh,
Dr. D.N Vishwakarma, Dr. Bhaskar Biswas

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 unlike frequent itemset mining (FIM) or association-rule mining (ARM).**

We have used particle swarm optimization algorithm along with top k concept in order to find high utility itemset. The case of negative utility is also considered in optimization.

I. Introduction:

The problem of **high-utility itemset mining** is an extension of the problem of **frequent pattern mining**. Frequent pattern mining is a popular problem in data mining, which consists in finding frequent patterns in transaction databases. The problem of frequent itemset mining is popular. But it has some important limitations when it comes to analyzing customer transactions. An important limitation is that purchase quantities are not taken into account. Thus, an item may only appear once or zero time in a transaction. To address these limitations, the problem of frequent itemset mining has been redefined as the problem of **high-utility itemset mining.**
An itemset is considered as a high utility itemset (HUI) if its utility value is no less than the user-specific minimum utility threshold. In real-world practice, the utility of an itemset can be measured by various factors, such as weight, profit, or cost, which can be defined by users' preferences.

II. Related Work:

- **Particle swarm optimisation**

In the past, many heuristic algorithms have been facilitated to solve the optimization problems for discovering the necessary information. It was proposed in 1995.

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 behaviour using the IR value instead of the specific minimum support and minimum confidence thresholds (Kuo et al. 2011). (Pears and Koh) presented a feasible method to mine the weighted association rules based on PSO (Pears and Koh 2012) problems. (Kennedy and Eberhart) then 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 by the probabilities of sigmoid function.(Sarath and Ravi) then applied the BPSO optimization approach to discover ARs (Sarath and Ravi 2013). Besides of PSO, several evolutionary algorithms such as differential evolution (Gong et al. 2010;Storn and Price 1997) were developed and designed for solving the optimization problem.

III. Definition and notation

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

A profit table

Item	Profit
a	3
b	9
c	1
d	5
e	6
f	1

(External utility) of each item is shown in Table as the profit table. The minimum utility threshold is set as (δ = 30%).

Definition 1: The utility of an item *i* in a transaction *Tq* is denoted as *u(i , Tq)* and is defined as :-

$$u(i, Tq) = q(i, Tq) * pr(i)$$

For example, the utility of items (a), (c) and (e) in transaction T1 are, respectively, calculated as-

$u(a, T1) = q(a, T1) * pr(a) = 1 \times 3 = 3$
$u(c, T1) = q(c, T1) * pr(c) = 18 \times 1 = 18$
$u(e, T1) = q(e, T1) * pr(e) = 1 \times 6 = 3$

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

$$u(X, Tq) = \sum_{i \subseteq X \wedge X \subseteq Tq} u(i, Tq)$$

For example, the utility of the itemsets (ac) and (ace) in transaction T1 are, respectively, calculated as-

u(ac, T1) = u(a, T1) + u(c, T1)
= q(a, T1) × pr(a) + q(c, T1) × pr(c)
=1×3 +18×1 = 21 and,

u(ace, T1) = u(a, T1) + u(c, T1) + u(e, T1)
= q(a, T1) × pr(a) + q(c, T1) × pr(c) + q(e, T1) × pr(e)
= 1 × 3 + 18 × 1 + 1 × 6 = 27

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

$$u(X) = \sum_{X \subseteq Tq \wedge Tq \in D} u(X, Tq$$

For example, the utility of itemsets *(b)* and *(bc)* in *D* are, respectively, calculated as

u(b) = u(b, T2) + u(b, T6) +u (b, T7) + u(b, T9) + u(b, T10)
= 54 + 9 + 90 + 9 + 54 =216 and

u(bc) = u(bc, T10) = 56

Definition 4: The transaction utility of a transaction *Tq* is denoted as *tu(Tq)* and defined as-

$$tu(Tq) = \sum_{X \in Tq} u(X, Tq)$$

For example, *tu(T1) = u(a, T1) + u(c, T1) + u(e, T1)*
=3 + 18 + 6 =27.

Definition 5: The total utility of a database *D* is denoted as *TU* and defined as:

$$TU \quad \sum_{Tq \in D} tu(Tq)$$

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 6: An itemset *X* in a database *D* is a high-utility itemset (HUI) iff its utility is no less than the minimum utility count as-

$$HUI \leftarrow \{X \mid u(X) \geq TU \times \delta\}.$$

For example, the utility of itemsets *(b)* and *(bc)* are, respectively, calculated as-

$$u(b) = 216 \text{ and } u(bc) = 56$$

Thus, the itemset *(b)* is a HUI since $u(b) = 216 > 386 \times 0.3 = 115.8$. The itemset *(bc)* is not a HUI since $u(bc) = 56 < 115.8$.

Definition 7: The transaction weighted utility of an itemset X is denoted as TWU(X), is defined by,

$$TWU(X) = \sum_{X \subseteq Tq \in D} tu(Tq)$$

IV. Algorithm used

In binary PSO (BPSO)-based model for mining HUIs, it consists of pre-processing, particle encoding, fitness evaluation, and the updating processes to mine HUIs. In the first pre-processing process, the high-transaction weighted utilization 1-itemsets (1-HTWUIs) are first discovered based on TWU model. This process can greatly reduce the invalid itemsets based on the transaction-weighted downward closure (TWDC) property. In the second particle encoding process, the items of 1-HTWUIs are sorted in their alphabetic-ascending order corresponding to the *j*-th position of a particle. The particle is thus encoded as the set of binary variables corresponding to the sorted order of 1-HTWUIs. In the fitness evaluation, the particles are then evaluated to find the *pbest* and *gbest* particles in the evolution process. For the last updating process, the particles are correspondingly updated by velocities, *pbest*, *gbest*, and the sigmoid function. If the fitness value of the particle is no less than the minimum utility count, it is concerned as a HUI and put into the set of HUIs. This iteration is repeated until the termination criteria are achieved. After that, the set of HUIs is discovered. Details of the four phases are respectively described below.

4.1 Pre-processing phase

In the designed HUIM-BPSO algorithm, the TWU model 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 HTWUIs, the unpromising items can be efficiently pruned. Thus, the computations for discovering HUIs can be greatly improved. The phase to discover1-HTWUIs is described below. First, the utility of items of each transaction is calculated as the transaction utility (*tu*).The transaction-weighted utility of an item is thus calculated by summing up the transaction utility if an item appearing in the transaction. This process is used to estimate the upper bound value of an item. If the transaction-weighted utility of an item is no less than the minimum utility count, it is considered as a HTWUI. In this example, the minimum utility count is calculated as (386 × 0.3 = 115.8). The discovered1-HTWUIs are shown in Table.

Item	TWU	1-HTWUIs
(*a*)	110	No
(*b*)	264	Yes
(*c*)	183	Yes
(*d*)	233	Yes
(*e*)	361	Yes
(*f*)	163	Yes

4.2 Particle encoding phase

The initial particles are randomly generated on the valid combinations of OR/NOR Tree Structure.

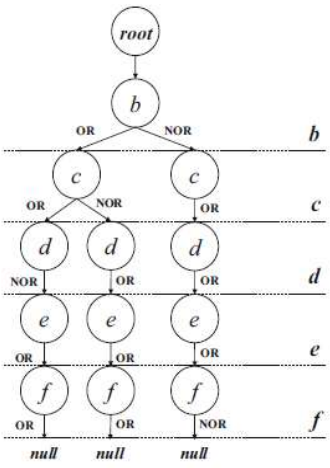
Each particle in the designed approach is to present a set of itemsets, forming the potential HUI. The size of the particle is the number of 1-HTWUIs found in the first pre-processing phase. Each particle is composed by a set of binary variables as 0 or 1, which indicates that a correspondingly item is present or absent in a particle. If the corresponding j-th position of a particle is set as 1, it indicates that an item in j-th position is presented as a potential HUI; otherwise, this item is not included and cannot be a potential HUI. 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. The velocities of particles in a population are randomly generated in the range of (0, 1). In this example, the size of a particle is set as 5 and the particles can be shown as below.

	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

4.3 Evaluation phase

The fitness function in the designed algorithm is utilized to evaluate the utility value of each particle, which is the same as the traditional HUIM to discover HUIs as

Fitness*(pi)* = *u(X),*



The designed OR/NOR-tree structure

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 considered as a HUI and will be put in the set of HUIs.

4.4 Updating phase

After the evaluation process of the particles in a population, the velocities of the particles are updated according to the traditional PSO approach, shown in the Eq. (1). The particles are updated based on *sigmoid* function used in BPSOA binary PSO approach to mine high-utility itemsets approach.The obtained equation for updating the particles is shown as follows:

$$\mathrm{xid}(t+1) = \begin{cases} 1 & rand() < sig\big(\mathrm{vid}(t+1)\big) = \frac{1}{1+e^{-\mathrm{vid}(t)}} \\ 0 & otherwise \end{cases}$$

From the above Eq. the *sigmoid* function is used for normalization. The *rand*() function is a uniform distribution in the range of (0, 1). This equation is used to determine the probability of the (*j*-th)

position of a particle. When the generated *rand()* is less than the *sig(vid (t + 1))*, the value of the corresponding *j*-th position of a particle is set as 1;otherwise, it is set as 0.

V. Top – K Concept

Sometimes the value of minimum utility cannot be decided for the large databases.
So instead of guessing minimum threshold value, we found the top K number of itemsets for the given database.
In this project we initially set the minimum utility count to zero and then keep on updating the minimum utility count to find the K number of itemsets have maximum utility.

Algorithm of Top –K Concept in BPSO:

Input: *D*, a quantitative database; *ptable*, a profit table, K is number of top HUIs to mine, *M* is the number of particles of each iteration.

Output: *HUIs*, a set of high-utility itemsets

1 for *each Tq* ∈ *D* **do**
2 **for** *each* $i_j \subseteq Tq$ **do**
3 $tu(Tq) = q(i_j, Tq) \times ptable(i_j)$;

4 calculate $twu(i_j) = \sum_{ij \in Tq} tu(Tq)$
5 $TU = \sum_{Tq \in D} tu(Tq)$
6 find 1-HTWUIs ← $\{i_j \mid twu(i\,j) \geq 0\}$;
7 set I_n = |1-HTWUIs|
8 set particle size construct OR/NOR-tree;
9 Initialize HUI_count = 0
10 Initialize min_util = 0
11 for *i* ← 1 *to M* **do**
12 **for** *j* ← 1 *to* I_n **do**
13 initialize $p_i^j(t)$ = either 0 or 1 (Initial the particles according to OR/NOR-tree)
14 Initialize $v^j_i(t) = rand()$
15 initial the velocity of particles in (0, 1)
16 **if** $fitness(p_i(t)) \geq min\_util$
if *HUI_count <k* **then**
17 *HUI s* ← *Get Item(pi (t))* ∪ *HUIs*
18 HUI_count ←HUI_count + 1
else
19 replace Item(p_i(t)) with Itemset having minimum utility in HUIs list.
20 update min_util with least itemset utility in HUIs list.
21 find *pbest (t)* of each *M* particle
22 update pbest find *gbest (t)* among *M pbest* particles and *gbest (t)*
23 update gbest
24 while *termination criteria is not reached*
25 for *i* ← 1*, M* **do**
26 update the $v_i(t + 1)$ velocities of *M* particles by equation.
for *j* ← 1 *to* I_n **do**
27 check the OR/NOR-tree to return the true value of the *j* -th position of $p^j_i(t + 1)$;
28 update the $p^j_i(t + 1)$ of *M* particles by equation
29 **if** $fitness(p_i(t)) \geq min\_util$
if *HUI_count <k* **then**
30 *HUI s* ← *Get Item(pi (t))* ∪ *HUIs*
31 HUI_count ←HUI_count + 1
else
32 replace Item(p_i(t)) with Itemset having minimum utility in HUIs list.
33 update min_util with least itemset utility in HUIs list.
34 find *pbest (t + 1)* of each *M* particle

35 update pbest find *gbest (t + 1)* among *M pbest* particles and *gbest (t)*
36 update gbest
37 set *t* ← *t* + 1;

38 return HUIs list;

VI. Experimental Result for Top – K Concept

For the given data:

Itemset	**T_q**	**Utility(of each item)**
2 3 4	9	2 2 5
1 2 3 4 5	18	4 2 3 5 4
1 3 4	11	4 2 5
3 4 5	11	2 5 4
1 2 4 5	22	5 4 5 8
1 2 3 4	17	3 8 1 5
4 5	9	5 4

With K equals to 5,

The itemset without using OR/NOR Tree:

1 4	#UTIL: 36
1 2 4	#UTIL: 41
4 5	#UTIL: 40
2 4	#UTIL: 36
1 2 4 5	#UTIL: 37

Total time ~ 46 ms
Memory ~ 1.2801971435546875 MB

The itemsets using OR/NOR tree:

1 2 4 5	#UTIL: 37
1 2 4	#UTIL: 41
1 4	#UTIL: 36
2 4	#UTIL: 36
4 5	#UTIL: 40

Total time ~ 15 ms
Memory ~ 1.2801971435546875 MB

Considering negative utility for item 2, the itemsets are:

1 3 4	#UTIL: 32
4 5	#UTIL: 40
1 4 5	#UTIL: 31
1 2 4 5	#UTIL: 25
3 4 5	#UTIL: 23

VII. Mining Closed Itemsets

The concept of closed itemsets extends from the concept of closed patterns from Frequent Itemsets Mining (FIM). A Closed High Utility Itemset (CHUI) is a HUI having no proper supersets that are HUIs and appear in same number of transactions in the database i.e. having same support.
It removes the redundant HUIs in the output. For Examples:
If itemset [1 2 3] is a HUI and itemset [1 2 3 4] is also HUI and itemset [1 2 3] always appear with item [4] in the database. So itemset [1 2 3] is a HUI due to itemset[1 2 3 4].

A utility-bin array can be used to efficiently calculate the support in O(n) time (n is the number of transactions). A utility-bin array U is initialized. Then, for each transaction T of the database, the utility-bin U[z] for each item $z \in T \cap E(\alpha)$ is updated as $U[z] = U[z] + 1$. Thereafter, we have $U[k] = \sup(\alpha \cup \{k\})\ \forall\ k \in E(\alpha)$.

Algorithm for Closed Itemset Mining using BPSO:

Input: *D*, a quantitative database; *ptable*, a profit table, K is number of top HUIs to mine , *M* is the number of particles of each iteration, min_util is the minUtility count.

Output: *HUIs*, a set of high-utility itemsets
1 for *each Tq* ∈ *D* **do**
2 **for** *each* $i_j \subseteq Tq$ **do**
3 $tu(Tq) = q(i_j, Tq) \times ptable(i_j)$;

4 calculate $twu(i_j) = \sum_{ij \in Tq} tu(Tq)$
5 $TU = \sum_{Tq \in D} tu(Tq)$
6 find 1-HTWUIs ← $\{i_j \mid twu(i\ j) \geq 0\}$;
7 set I_n = |1-HTWUIs|
8 Intialize utilityListOfItem
9 set particle size construct OR/NOR-tree;
10 Initialize HUI_count = 0

11 for $i \leftarrow 1$ *to* M **do**
12 **for** $j \leftarrow 1$ *to* I_n **do**
13 initialize $p_i^j(t)$ = either 0 or 1 (Initial the particles according to OR/NOR-tree)
14 Initialize $v^j_i(t) = rand()$
15 initial the velocity of particles in (0, 1)
16 **if** *fitness*($p_i(t)$) ≥ min_util
17 Calculate support for p_i using utilityListOfItem
18 **for** itemsets in HUIs list
if particle is same as itemsets
19 **break**
else if itemset is subset of HUIs list itemset and support for both are equal
20 **break**
else if HUIs itemset is subset of the itemset and support of both are same
21 **remove** HUI itemset
22 find *pbest (t)* of each *M* particle
23 update pbest find *gbest (t)* among *M pbest* particles and *gbest (t)*
24 update gbest
25 while *termination criteria is not reached*
26 for $i \leftarrow 1$, *M* **do**
27 update the $v_i(t+1)$ velocities of *M* particles by equation.
for $j \leftarrow 1$ *to* I_n **do**
28 check the OR/NOR-tree to return the true value of the *j* -th position of $p^j_i(t+1)$;
29 update the $p^j_i(t+1)$ of *M* particles by equation
30 **if** *fitness*($p_i(t)$) ≥ *min_util*
31 Calculate support for p_i using utilityListOfItem
32 **for** itemsets in HUIs list
if particle is same as itemsets
33 **break else if** itemset is subset of HUIs list itemset and support for both are equal
34 **break**
else if HUIs itemset is subset of the itemset and support of both are same
35 **remove** HUI itemset

36 find *pbest* $(t+1)$ of each *M* particle
37 update pbest find *gbest* $(t+1)$ among *M pbest* particles and *gbest (t)*
38 update gbest
39 set $t \leftarrow t + 1$;
40 return HUIs list;

VIII. Experiment result using Closed Concept

For the given data:

Itemset	**T_q**	**Utility(of each item)**
2 3 4	9	2 2 5
1 2 3 4 5	18	4 2 3 5 4
1 3 4	11	4 2 5
3 4 5	11	2 5 4
1 2 4 5	22	5 4 5 8
1 2 3 4	17	3 8 1 5
4 5	9	5 4

With min_util equals to 22,

The itemset without closed concept:

1 2 5	#UTIL: 27	#Support: 2
1 4 5	#UTIL: 31	#Support: 2
4 5	#UTIL: 40	#Support: 4
1 2 4 5	#UTIL: 37	#Support: 2
1 4	#UTIL: 36	#Support: 4
1 2 3 4	#UTIL: 31	#Support: 2
2 4 5	#UTIL: 28	#Support: 2
3 4	#UTIL: 35	#Support: 5
1 3 4	#UTIL: 32	#Support: 3
1 2	#UTIL: 26	#Support: 3
3 4 5	#UTIL: 23	#Support: 2
4	#UTIL: 35	#Support: 7

Total time ~ 16 ms
Memory ~ 1.2882232666015625 MB
High-utility itemsets count : 12

The itemsets using closed concept:

2 3 4	#UTIL: 33	#Support: 3
1 2 4	#UTIL: 41	#Support: 3
1 4	#UTIL: 36	#Support: 4
1 2 3 4	#UTIL: 31	#Support: 2
2 4	#UTIL: 36	#Support: 4
1 2 4 5	#UTIL: 37	#Support: 2
4 5	#UTIL: 40	#Support: 4
3 4 5	#UTIL: 23	#Support: 2

Total time ~ 32 ms
Memory ~ 1.6083221435546875 MB
High-utility itemsets count : 8

IX. Conclusion and Future Work

In this project, we have presented a novel algorithm named Binary Swarm Particle Optimization (using Top-K concept) for mining high utility itemsets in databases where item unit profits may be positive or negative. We have used Educational Feedback real time and standard datasets. When measuring the running time of BPSO algorithm, the min_util values are varied for the negative databases. For negative databases, lower the min_util is, larger the number of high utility itemsets are, and thus the more the running time is. We also have applied an approach called Closed Itemset Mining and fused it with Binary Particle Swarm Optimization Algorithm so as to increase the overall accuracy of the mined High Utility Itemsets by reducing the redundancy. When Closed Mining was applied it was seen that the running time increased because of the support calculation of HUI's and use of Hash- Maps however the results were more useful as redundancy was checked.

In future we can explore this search for finding the sequential patterns from the databases containing negative utility values of the items using Generic Algorithm and further pruning can be applied using Fuzzy Logic technique.

X. References

1. **A binary PSO approach to mine high-utility itemsets**
   Jerry Chun-Wei Lin, Lu Yang, Philippe Fournier-Viger, Tzung-Pei Hong, Miroslav Voznak.

2. **Mining high-utility itemsets based on particle swarm optimization**
   Jerry Chun-Wei Lin, Lu Yang, Philippe Fournier-Viger, Jimmy Ming-Thai Wu, Tzung-Pei Hong, Leon Shyue-Liang Wang, Justin Zhan.

3. **Fast and Memory Efficient Discovery of Closed High-Utility Itemsets**
   Philippe Fournier-Viger, Souleymane Zida, Jerry Chun-Wei Lin, Cheng-Wei Wu, Vincent S. Tseng