Exploiting High Utility Occupancy Patterns

Wensheng Gan[1], Jerry Chun-Wei Lin[1⋆], Philippe Fournier-Viger[2], and Han-Chieh Chao[1,3]

[1]School of Computer Science and Technology
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
wsgan001@gmail.com, jerrylin@ieee.org
[2]School of Natural Sciences and Humanities
Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
philfv@hitsz.edu.cn
[3]Department of Computer Science and Information Engineering
National Dong Hwa University, Hualien, Taiwan
hcc@ndhu.edu.tw

Abstract. Most studies have considered the frequency as sole interestingness measure for identifying high quality patterns. However, each object is different in nature, in terms of criteria such as the utility, risk, or interest. Besides, another limitation of frequent patterns is that they generally have a low occupancy, and may not be truly representative. Thus, this paper extends the occupancy measure to assess the utility of patterns in transaction databases. The High Utility Occupancy Pattern Mining (HUOPM) algorithm considers user preferences in terms of frequency, utility, and occupancy. Several novel data structures are designed to discover the complete set of high quality patterns without candidate generation. Extensive experiments have been conducted on several datasets to evaluate the effectiveness and efficiency of HUOPM.

Keywords: Frequency · Occupancy · Utility occupancy · Data structure

1 Introduction

Frequent itemset mining (FIM) and association rule mining (ARM) are some of the most important and fundamental KDD techniques [1, 2, 5, 6], which are applied in numerous domains. In recent decades, the task of frequent pattern mining has been extensively studied by mainly considering the frequency measure for selecting patterns. In real-life applications, the importance of objects or patterns is often evaluated in terms of implicit factors such as the utility, interestingness, risk or profit. Hence, the knowledge that actually matters to user may not be found using traditional FIM and ARM algorithms. Thus, a utility-based mining framework called high utility pattern mining (HUPM) was proposed [4, 9], which considers the relative importance of items (item utility). The utility (i.e., importance, interest or risk) of each item can be predefined based on users'

⋆ Corresponding author

background knowledge or preferences, it has become an emerging research topic in recent years [3, 4, 7–9, 13].

Recently, a study [12] has shown that considering the *occupancy* of patterns is critical for many applications. This allows to find patterns that are more representative, and thus of higher quality. However, implicit factors such as the utility, interestingness, risk or profit of objects or patterns are ignored [12], and it ignores the fact that items/objects may appear more than once in transactions. Besides, HUPM does not assess the occupancy of patterns, the discovered HUPs may be irrelevant or even misleading if they are frequent but are not representative of the supporting transactions. Hence, it is desirable to find patterns that are representative of the transactions where they occur. Recently, the OCEAN algorithm was proposed to address the problem of high utility occupancy pattern mining by introducing the utility occupancy measure [11]. However, OCEAN fails to discover the complete set of high utility occupancy patterns and also encounter performance problems.

Therefore, this paper proposes an effective and more efficient algorithm named **H**igh **U**tility **O**ccupancy **P**attern **M**ining in transactional databases (**HUOPM**). The proposed algorithm extracts patterns based on users' interests, pattern frequency and utility occupancy, and considers that each item may have a distinct utility. The major contributions are summarized as follows. (i) A novel and effective HUOPM algorithm is proposed to address the novel research problem of mining high utility occupancy patterns with the utility occupancy measure. To the best of our knowledge, no prior algorithms address this problem successfully and effectively. (ii) Two data structures called utility-occupancy list (UO-list) and frequency-utility table (FU-table), are developed to store the required information about a database, for mining HUOP without repeatedly scanning the database. And the remaining utility occupancy is utilized to calculate an upper bound to reduce the search space. (iii) Extensive experiments have been conducted on real-world datasets to evaluate how effective and efficient the proposed HUOPM algorithm is.

2 Preliminaries and Problem Statement

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a finite set of m distinct items in a transactional database $D = \{T_1, T_2, \ldots, T_n\}$, where each quantitative transaction $T_q \in D$ is a subset of I, and has a unique identifier *tid*. The support count of an itemset X, denoted as $sup(X)$, is the number of *supporting transactions* containing X [2]. Transaction T_q supports X if $X \subseteq T_q$, and the set of transactions supporting X is denoted as Γ_X. Thus, $sup(X) = |\Gamma_X|$. An itemset X is a frequent pattern (FP) in D if $sup(X) \geq \alpha \times |D|$, where α is minimum support threshold.

Definition 1. Each item i_m in a database D has a unit profit denoted as $pr(i_m)$, which represents its relative importance to the user. Item unit profits are indicated in a profit-table, denoted as $ptable = \{pr(i_1), pr(i_2), \ldots, pr(i_m)\}$. The utility of an item i_j in T_q is $u(i_j, T_q) = q(i_j, T_q) \times pr(i_j)$, in which $q(i_j)$ is the quantity

of i_j in T_q. The utility of an itemset X in T_q is $u(X, T_q) = \sum_{i_j \in X \wedge X \subseteq T_q} u(i_j, T_q)$. Thus, the utility of X in D is $u(X) = \sum_{X \subseteq T_q \wedge T_q \in D} u(X, T_q)$. The transaction utility of T_q is $tu(T_q) = \sum_{i_j \in T_q} u(i_j, T_q)$ [4, 9].

Definition 2. The utility occupancy of an itemset X in T_q and D are respectively defined as $uo(X, T_q) = \frac{u(X, T_q)}{tu(T_q)}$ and $uo(X) = \frac{\sum_{X \subseteq T_q \wedge T_q \in D} uo(X, T_q)}{|\Gamma_X|}$.

Definition 3. Given a minimum support threshold α ($0 < \alpha \leq 1$) and a minimum utility occupancy threshold β ($0 < \beta \leq 1$), an itemset X in a database D is said to be a high utility occupancy pattern, denoted as $HUOP$, if it satisfies the following two conditions: $sup(X) \geq \alpha \times |D|$ and $uo(X) \geq \beta$.

3 Proposed Algorithm for Mining HUOPs

The search space of HUOPM can be represented as a Set-enumeration tree [10]. The *downward closure* property does not hold for HUOPs, it is a critical issue to design more suitable data structures and efficient pruning strategies to reduce the search space for mining HUOPs.

Definition 4. A frequency-utility tree (FU-tree) is designed as a sorted Set-enumeration tree using the total order $\prec$ on items. Each child node (called extension node) is generated by extending its prefix (parent) node.

Definition 5. The remaining utility occupancy of an itemset X in T_q is denoted as $ruo(X, T_q)$, and defined as the sum of the utility occupancy values of each item appearing after X in T_q: $ruo(X, T_q) = \frac{\sum_{i_j \notin X \wedge X \subseteq T_q \wedge X \prec i_j} u(i_j, T_q)}{tu(T_q)}$.

Definition 6. The utility-occupancy list (UO-list) of an itemset X in a database D is a set of tuples corresponding to transactions where X appears. A tuple contains $<tid, uo, ruo>$ for each T_q containing X, the uo element is the utility occupancy of X in T_q, i.e., $uo(X, T_q)$; the ruo element is defined as the remaining utility occupancy of X in T_q, w.r.t. $ruo(X, T_q)$.

Definition 7. A frequency-utility table (FU-table) of an itemset X contains four informations: the name of the itemset X, the support of X, the sum of the utility occupancies of X in database D (***uo(X)***), and the sum of the remaining utility occupancies of X in D (***ruo(X)***, and $ruo(X) = \frac{\sum_{X \subseteq T_q \wedge T_q \in D} ruo(X, T_q)}{|\Gamma_X|}$).

The construction procedure of the UO-list and FU-table is shown in Algorithm 1, it returns X_{ab} which having the UO-list and FU-table of X_{ab}, denoted as $X_{ab}.UOL$ and $X_{ab}.FUT$.

Lemma 1. Let there be a subtree rooted at X, and Γ_X be the supporting transactions of X. For any possible itemset W in the subtree, we have: $uo(W) \leq \frac{\sum_{W \subseteq T_q \wedge T_q \in D} (uo(X, T_q) + ruo(X, T_q))}{|\Gamma_W|}$.

Lemma 2. Given a minimum support threshold α, a subtree rooted at X. For any pattern W in the subtree, an upper bound on the utility occupancy of W is: $\hat{\phi}(W) = \dfrac{\sum_{top\alpha\times|D|,T_q\in\Gamma_X}(uo(X,T_q)+ruo(X,T_q))}{\alpha\times|D|} \geq uo(W)$. Thus, we can directly calculate an upper bound $\hat{\phi}(W)$ on the utility occupancy of a subtree rooted at a processed node X.

Algorithm 1: Construction procedure

Input: X, X_a: extension of X by adding a, X_b: extension of X by adding b.
Output: X_{ab}.

```
1  set X_ab.UOL ← ∅, X_ab.FUT ← ∅;
2  for each tuple E_a ∈ X_a.UOL do
3      if ∃E_a ∈ X_b.UOL ∧ E_a.tid == E_b.tid then
4          if X.UOL ≠ ∅ then
5              Search for element E ∈ X.UOL, E.tid = E_a.tid;
6              E_ab ←< E_a.tid, E_a.uo + E_b.uo − E.uo, E_b.ruo >;
7              X_ab.FUT.uo += E_a.uo + E_b.uo − E.uo;
8              X_ab.FUT.ruo += E_b.ruo;
9          else
10             E_ab ←< E_a.tid, E_a.uo + E_b.uo, E_b.ruo >;
11             X_ab.FUT.uo += E_a.uo + E_b.uo;
12             X_ab.FUT.ruo += E_b.ruo;
13         X_ab.UOL ← X_ab.UOL ∪ E_ab;
14         X_ab.FUT.sup ++;
15     else
16         X_a.FUT.sup - -;
17         if X_a.FUT.sup < α × |D| then
18             return null;
19 return X_ab
```

Theorem 1. (*Global downward closure property in the FU-tree*) In the FU-tree, if a tree node is a FP, its parent node is also a FP. Let X^k be a k-itemset (node) and its parent node be denoted as X^{k-1}, which is a (k-1)-itemset. The relationship $sup(X^k) \leq sup(X^{k-1})$ holds.

Theorem 2. (*Partial downward closure property in the FU-tree*) In the FU-tree, the upper bound on utility occupancy of any node in a subtree is no greater than that of its parent node always holds, that is $\hat{\phi}(X^k) \leq \hat{\phi}(X^{k-1})$.

Strategy 1 *If a tree node X has a support count less than ($\alpha \times |D|$), then any nodes containing X (i.e. all supersets of X) can be directly pruned.*

Strategy 2 *In the FU-tree, if the upper bound on the utility occupancy of a tree node X is less than β, then any nodes in the subtree rooted at X w.r.t. all extensions of X can be directly pruned and do not need to be explored.*

Strategy 3 *During the construction of the UO-list, if the remaining support of X_a for constructing X_{ab} is less than $\alpha \times |D|$, the support of X_{ab} will also be*

less than $\alpha \times |D|$, X_{ab} is not a FP, and not a HUOP. Then the construction procedure return null, as shown in Algorithm 1 Lines 16-18.

Based on the above pruning strategies, algorithm 2 shows the pseudocode of the designed HUOPM algorithm. Details of the ***HUOP-Search*** and the ***UpperBound*** procedures are shown in Algorithm 3 and Algorithm 4, respectively.

Algorithm 2: HUOPM algorithm

Input: D, *ptable*, α, β.
Output: The complete set of high utility occupancy patterns.
1 scan D to calculate the $sup(i)$ of each item $i \in I$ and and $tu(T_q)$ of each transaction;
2 find $I^* \leftarrow \{i \in I | sup(i) \geq \alpha \times |D|\}$;
3 sort I^* in the designed total order $\prec$;
4 scan D once to build the UO-list and FU-table for each 1-item $i \in I^*$;
5 **call *HUOP-Search***$(\phi, I^*, \alpha, \beta)$;
6 return $HUOPs$

Algorithm 3: HUOP-Search procedure

Input: X, *extenOfX*, α, β.
Output: The complete set of *HUOPs*.
1 **for** *each itemset* $X_a \in extenOfX$ **do**
2 obtain $sup(X_a)$ and $uo(X_a)$ from the built $X_a.FUT$;
3 **if** $sup(X_a) \geq \alpha \times |D|$ **then**
4 **if** $uo(X_a) \geq \beta$ **then**
5 $HUOPs \leftarrow HUOPs \cup X_a$;
6 $\hat{\phi}(X_a) \leftarrow$ ***UpperBound***$(X_a.UOL, \alpha)$;
7 **if** $\hat{\phi}(X_a) \geq \beta$ **then**
8 $extenOfX_a \leftarrow \emptyset$;
9 **for** *each* $X_b \in extenOfX$ *such that* $X_a \prec X_b$ **do**
10 $X_{ab} \leftarrow X_a \cup X_b$;
11 call ***Construct***(X, X_a, X_b);
12 **call *HUOP-Search*$(X_a, extenOfX_a, \alpha, \beta)$**;
13 **return** $HUOPs$

4 Experimental Results

Only two prior studies, DOFIA [12] and OCEAN [11], are closely related to our work. Major differences are that DOFIA considers both the frequency and

Algorithm 4: UpperBound procedure

Input: $X_q.UOL$, α.
Output: The upper bound $\hat{\phi}(X_a)$.
1 $sumTopK \leftarrow 0, \hat{\phi}(X_a) \leftarrow 0, V_{occu} \leftarrow \emptyset$;
2 calculate $(uo(X, T_q) + ruo(X, T_q))$ of each tuple from the built $X_a.UOL$ and put them into the set of V_{occu};
3 sort V_{occu} by descending order as $V^{\downarrow}_{occu}$;
4 **for** $k \leftarrow 1$ *to* $\alpha \times |D|$ *in* $V^{\downarrow}_{occu}$ **do**
5 $\quad sumTopK \leftarrow sumTopK + V^{\downarrow}_{occu}[k]$;
6 $\hat{\phi}(X_a) = \dfrac{sumTopK}{\alpha \times |D|}$;
7 **return** $\hat{\phi}(X_a)$

occupancy for mining high qualified patterns, but the occupancy is based on the number of items in itemsets/transactions rather than the concept of utility [12]. Thus, OCEAN [11] is implemented to generate high utility occupancy patterns (denoted as HUOPs*), and HUOPs are generated by HUOPM. All algorithms in the experiments are implemented using the Java language and executed on a PC with an Intel Core i5-3460 3.2 GHz processor and 4 GB of memory, running on the 32 bit Microsoft Windows 7 platform. Two real-world datasets, BMSPOS [1] and chess [1], are used in the experiments with a simulation method [13]. Note that there are three versions, HUOPM$_{P12}$ (with strategies 1 and 2), HUOPM$_{P13}$ (with strategies 1 and 3), and HUOPM$_{P123}$ (with strategies 1, 2 and 3), are compared to evaluate the efficiency of HUOPM.

Table 1. Number of patterns under various parameters

BMSPOS	α(%)	0.010	0.015	0.020	0.025	0.030	0.035
(β: 0.3)	HUOPs*	29389	17005	11876	8961	7065	5884
	HUOPs	29444	17048	11899	8987	7094	5905
chess	α(%)	60	61	62	63	64	65
(β: 0.3)	HUOPs*	2423	1776	1374	1108	903	695
	HUOPs	11469	8949	7075	5510	4273	3351
BMSPOS	β(%)	0.24	0.26	0.28	0.30	0.32	0.34
(α: 0.02%)	HUOPs*	287850	106239	34195	11876	5089	2696
	HUOPs	287902	106285	34233	11899	5115	2721
chess	β(%)	0.26	0.28	0.30	0.32	0.34	0.36
(α: 62%)	HUOPs*	9820	4837	1374	514	125	14
	HUOPs	24905	14050	7075	3153	1170	357

Pattern Analysis. To analyze the usefulness of the HUOPM framework, the derived pattern HUOPs* and HUOPs are evaluated. Results for various parameter values are shown in Table 1. It can be clearly observed that the sets of derived HUOPs* is always smaller than that of HUOPs, which means that numerous interesting patterns are effectively discovered by HUOPM, while most

of them are missed by OCEAN. Thus, OCEAN fails to discover the completed set of HUOPs. Both HUOPs* and HUOPs decrease when α is increased, less HUOPs* and HUOPs are obtained when β is set higher. Besides, the number of missing patterns (i.e. HUOPs - HUOPs*) sometimes increases when varying β, while it sometimes decreases. It can be concluded that the proposed HUOPM algorithm is acceptable and solve a serious shortcoming of OCEAN.

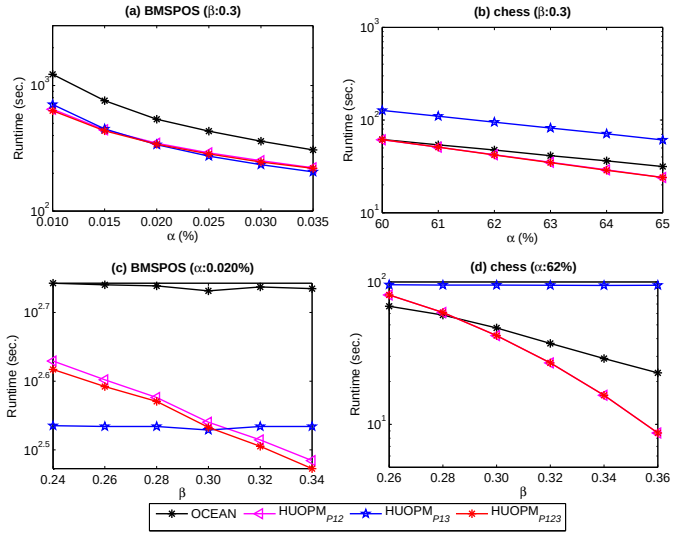

Fig. 1. Runtime under various parameters.

Efficiency Analysis. A performance comparison of the different strategies used in HUOPM is presented in Fig. 1. It can be clearly observed that the runtime of OCEAN is the worst, compared to the other algorithms in most cases. We also draw the following conclusions. (i) The difference between HUOPM$_{P12}$ and HUOPM$_{P123}$ indicates that pruning strategy 3 always reduces the search space by pruning subtrees. (ii) By comparing HUOPM$_{P13}$ and HUOPM$_{P123}$, it can be concluded that pruning strategy 2, using the upper bound of utility occupancy, provides a trade-off between efficiency and effectiveness. Because it needs to spend additional time to calculate the upper bounds, but sometimes unpromising itemsets can be directly pruned by other pruning strategies. (iii) HUOPM applies pruning strategies to prune unpromising itemsets early, which greatly speed up the mining efficiency, compared to the OCEAN algorithm.

5 Conclusions

In this paper, we proposed an effective HUOPM algorithm to address a new research problem of mining high utility occupancy patterns with utility occupancy.

The utility occupancy can lead to useful itemsets that contribute a large portion of total utility for each individual transaction representing user interests or user habit. Based on the two *downward closure* properties and pruning strategies, HUOPM can directly mine HUOPs without candidate generation. Extensive experiments show that HUOPM can efficiently find the complete set of HUOPs and significantly outperforms the state-of-the-art OCEAN algorithm.

6 Acknowledgments

This research was partially supported by the National Natural Science Foundation of China (NSFC) under grant No. 61503092 and by the Tencent Project under grant CCF-Tencent IAGR20160115.

References

1. Frequent itemset mining dataset repository. http://fimi.ua.ac.be/data/. 2012.
2. Agrawal, R., Srikant, R. Fast algorithms for mining association rules in large databases. *Intern. Conf. on Very Large Data Bases*, pp. 487–499, 1994.
3. Ahmed, C.F., Tanbeer, S.K., Jeong, B.S., Le, Y.K. Efficient tree structures for high utility pattern mining in incremental databases. *IEEE Transactions on Knowledge and Data Engineering*, 21(12):1708–1721, 2009.
4. Chan, R., Yang, Q., Shen, Y.D.: Mining high utility itemsets. *Intern. Conf. on Data Mining*, pp. 19–26, 2003.
5. Fournier-Viger, P., Lin, J.C.W., Kiran, R.U., Koh, Y.S., Thomas, R.: A survey of sequential pattern mining. *Data Science and Pattern Recognition*, vol. 1(1), pp. 54-77, 2017.
6. Han, J. Pei, J., Yin, Y. Mao, R. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Mining and Knowledge Discovery*, 8(1):53–87, 2004.
7. Liu, M., Qu, J. Mining high utility itemsets without candidate generation. *ACM Intern. Conf. on Information and Knowledge Management*, pp. 55–64, 2012.
8. Lin J.C.W., Gan, W., Fournier-Viger, P., Hong, T.P., Tseng, V.S.: Efficient algorithms for mining high-utility itemsets in uncertain databases. *Knowledge Based Systems*, 96:171–187, 2016.
9. Yao, H., Hamilton, J., Butz, C.J. A foundational approach to mining itemset utilities from databases. *SIAM Intern. Conf. on Data Mining*, pp. 211–225, 2004.
10. Rymon, R. Search through systematic set enumeration. *Technical Reports (CIS)*, 297, 1992.
11. Shen, B., Wen, Z., Zhao, Y., Zhou, D., Zheng, W. OCEAN: fast discovery of high utility occupancy itemsets. *Pacific-Asia Conf. on Knowledge Discovery and Data Mining*, pp. 354–365, 2016.
12. Tang, L., Zhang, L., Luo, P., Wang, M. Incorporating occupancy into frequent pattern mining for high quality pattern recommendation. *21st ACM Intern. Conf. on Information and knowledge management*, pp. 75–84, 2012.
13. Tseng, V.S., Shie, B.E., Wu, C.W., Yu, P.S. Efficient algorithms for mining high utility itemsets from transactional databases. *IEEE Transactions on Knowledge and Data Engineering*, 25(8):1772–1786, 2013.