A More Efficient Algorithm to Mine Skyline Frequent-Utility Patterns

Jerry Chun-Wei Lin[1(✉)], Lu Yang[1], Philippe Fournier-Viger[2], Siddharth Dawar[3], Vikram Goyal[3], Ashish Sureka[4], and Bay Vo[5]

[1] School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
jerrylin@ieee.org, luyang@ikelab.net
[2] School of Natural Sciences and Humanities, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, China
philfv@hitsz.edu.cn
[3] Indraprastha Institute of Information Technology, Delhi, India
{siddharthd,vikram}@iiitd.ac.in
[4] ABB Corporate Research, Bangalore, India
ashish.sureka@in.abb.com
[5] Faculty of Information Technology, Ho Chi Minh City University of Technology, Ho Chi Minh City, Vietnam
bayvodinh@gmail.com

Abstract. In the past, a SKYMINE approach was proposed to both consider the aspects of utility and frequency of the itemsets to mine the skyline frequency-utility skyline patterns (SFUPs). The SKYMINE algorithm requires, however, the amounts of computation to mine the SFUPs based on the utility-pattern (UP)-tree structure performing in a level-wise manner. In this paper, we propose more effective algorithms to mine the SFUPs based on the utility-list structure. Substantial experiments are carried to show that the proposed algorithms outperform the state-of-the-art SKYMINE to mine the SFUPs in terms of runtime and memory usage.

Keywords: Skyline · Utility · Frequent · Umax · Utility-list

1 Introduction

The frequent itemset mining (FIM) is the fundamental task of Knowledge discovery in database, which is used to identify the set of frequent itemsets (FIs) [3,11,12,14,20,22]. In real-life situations, only frequency of the itemsets reveals the insufficient information. To solve the limitation of FIM, high-utility itemset mining (HUIM) [5,7,15,16,25] was proposed to discover the "useful" and "profitable" itemsets from the quantitative database. Lin et al. then presented the high-utility pattern (HUP)-tree algorithm [15] to mine the HUIs. The UP-Growth+ algorithm [24] was further designed to adopt several pruning strategies to speed up mining process based on the developed utility-pattern (UP)-tree.

J. Pan et al. (eds.), *Genetic and Evolutionary Computing*, Advances in Intelligent Systems and Computing 536, DOI 10.1007/978-3-319-48490-7_16

Yeh et al. [26] first proposed the two-phase algorithms for mining high utility and frequency itemsets. Podpecan et al. [21] then proposed a fast algorithm to mine the utility-frequent itemsets. However, it is difficult to define appropriate utility threshold and minimum support threshold for retrieving the required information.

Skyline contains the dominance relationship between tuples based on multi-dimensions. Borzsonyi et al. [6] addressed the first work of skylines in the context of databases and developed several algorithms based on block nested loops, divide-and-conquer, and index scanning mechanism. Chomicki et al. [9] employs a certain ordering of tuples in the window to increase performance of [6]. Tan et al. [23] proposed progressive (or on-line) algorithms that can progressively output skyline points without scanning the entire dataset. Other related works of skyline are still developed in progress [2,8,13,19].

Although FIM and HUIM have widely range in real-world applications, both of them can only focus on one aspect by respectively considering the occurrence frequency or the utility of the itemsets. Goyal et al. [10] first defined the skyline frequent-utility pattern (SFUP) and designed a SKYMINE algorithm to mine the itemsets with high occurrence frequency and high utility. However, the numerous candidates are required to be generated of the SKYMINE, which is a time-consuming task. The problem of memory leakage also happens in the SKYMINE since it is necessary to generate the numerous candidates for mining SFUPs. To speed up the mining process and reduce the memory usage, we design new algorithms to efficient mine the SFUPs. An efficient utility-list structure is adopted in this paper to efficiently mine the SFUPs without candidate generation. Besides, an *umax* array is further developed to keep the maximal utility under the occurrence frequency. A pruning strategy is also developed to reduce the search space for mining SFUPs. Extensive experiments on various databases were conducted and the results showed that the proposed algorithm has better performance than that of the SKYMINE for mining SFUPs.

2 Preliminaries and Problem Statement

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 $T_q \in D$ $(1 \leq q \leq n)$ 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 its purchase quantity (internal utility) and denoted as $q(i_j, T_q)$. A profit table ptable $= \{pr(i_1), pr(i_2), \ldots, pr(i_m)\}$ indicates the profit value 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 $X \subseteq T_q$.

Definition 1. The occurrence frequency of an itemset X in D is denoted as $f(X)$, where X is a set of items and $f(X)$ is defined as the number of transactions T_q in D containing X as:

$$f(X) = |\{X \subseteq T_q \wedge T_q \in D\}|. \tag{1}$$

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

$$u(i_j, T_q) = q(i_j, T_q) \times pr(i_j). \tag{2}$$

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

$$u(X, T_q) = \sum_{i_j \subseteq X \wedge X \subseteq T_q} u(i_j, T_q). \tag{3}$$

Definition 4. The utility of an itemset X in a database D is denoted as $u(X)$, and defined as:

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

Definition 5. The transaction utility of a transaction T_q is denoted as $tu(T_q)$ and defined as:

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

Definition 6. The transaction-weighted utility of an itemset X in D is denoted as $twu(X)$ and defined as:

$$twu(X) = \sum_{X \subseteq T_q \wedge T_q \in D} tu(T_q). \tag{6}$$

The above definitions are used to find whether the FIs or HUIs. The state-of-the-art algorithms whether in FIM or HUIM cannot consider both frequency and utility together. To obtain the skyline frequent-utility patterns (SFUPs), the definitions are given below.

Definition 7. An itemset X dominates another itemset Y in D, denoted as $X \succ Y$ iff $f(X) \geq f(Y)$ and $u(X) \geq u(Y)$.

Definition 8. An itemset X in a database D is a skyline frequent-utility pattern (SFUP) iff it is not dominated by any other itemset in the database by considering both the frequency and utility factors.

Problem Statement: Based on the above definitions, we define the problem of skyline frequent-utility pattern mining (SFUPM) as discovering the set of non-dominated itemsets in the database by considering both the frequency and utility factors.

3 Proposed Algorithms for Mining SFUPs

In this section, two efficient algorithms are designed for mining the set of skyline frequent-utility patterns (SFUPs). It is based on the well-known utility-list structure to fast combine the itemsets by simple join operation. Details are given as follows.

3.1 Utility-List Structure

Let $\rhd$ be an ascending order on the items in the databases. The utility-list [17] of an itemset X in a database D is a set of tuples, in which each tuple consists of three fields as $(tid, iutil, rutil)$. The tid is the transaction ID containing the itemset X. The $iutil$ and $rutil$ elements of a tuple respectively are the utility of X in tid; i.e., $u(X, T_q)$ and the resting utility of the items except the itemset X in tid, which can be defined as: $\sum_{i_j \in T_q \wedge i_j \notin X} u(i_j, T_q)$. Here is an example to illustrate the construed utility-list shown in Fig. 1.

E		
tid	*iutil*	*rutil*
2	4	14
4	4	2
5	8	14

C		
tid	*iutil*	*rutil*
1	2	7
2	3	11
3	2	9
4	2	0
6	1	11

B		
tid	*iutil*	*rutil*
1	2	5
2	2	9
5	4	10
6	8	3

A		
tid	*iutil*	*rutil*
2	4	5
3	4	5
5	5	5
6	3	0

D		
Tid	*lutil*	*Rutil*
1	*5*	*0*
2	*5*	*0*
3	*5*	*0*
5	*5*	*0*
7	*5*	*0*

Fig. 1. The constructed utility-lists.

In addition to keep the maximal utility of the frequency value, an utility-max ($umax$) array is then set in the beginning of the developed algorithm, which is used to keep the maximal utility of the frequency value.

Definition 9. An $umax$ array keeps the maximal utility of the frequency value r, which is defined as $umax(r)$.

Definition 10. An itemset X is considered as a potential SFUP (PSFUP) if its frequency is equal to r and non-itemset having higher utility than $u(X)$.

3.2 Pruning Strategy

To mine the SFUPs, the SKYMINE algorithm generates numerous candidates from the UP-tree structure, and the upper bound of the itemsets is overestimated based on the two-phase model. To solve above problems, we define a strategy to speed up mining process of the SFUPs.

Pruning Strategy: Let X be an itemset, and let the extensions of X by appending an item Y to X as $(X \cup Y)$ such that $X \rhd Y$. If the sum of $iutil$ and $rutil$ values in the utility-list of X is less than $umax(r)$, $r = f(X)$, then all the extensions of X are not SFUPs.

For each itemset X, it can be known that the frequency of its extensions is higher than or equals to itself. Based on above pruning strategy, it can be found that there must be an itemset Y dominates X' having the same frequency with X and same utility with $umax(r)$, $r = f(X)$. It indicates that X' is not a SFUP.

3.3 Proposed Algorithms

In the designed algorithms, the utility-list structure is first constructed. The transaction-weighted utility (twu) of all 1-itemsets is discovered and the 1-itemsets in the database are also sorted in twu-ascending order. The sorted database is then used to construct the initial utility-list of 1-itemsets. After that, the P-Miner algorithm is used to find the potential SFUPs. The pseudo-code of the **P-Miner** algorithm is described in Algorithm 1, which is used to find the potential SFUPs.

Algorithm 1. P-Miner

Input: $P.UL$, the utility-list of the itemset P; $P'ULs$, the set of utility-lists of $P's$ extensions; $umax$, an array to keep the maximum utility of the varied frequencies.
Output: $PSFUIs$; the set of $P's$ potential skyline frequent-utility itemsets.

```
1  for each X in P'ULs do
2      if sum(X.iutil) ≥ umax(f(X)) then
3          umax(f(X)) ← sum(X.iutil);
4          PSFUIs ← X;
5          remove Y from PSFUIs if (f(Y) == f(X));
6      if sum(X.iutil) + sum(X.rutil) ≥ umax(f(X)) then
7          exULs := null;
8          for each utilit-list Y after X in P'ULs do
9              exULs := exULs + construct(P.UL.X, Y);
10         P-Miner(X, exULs, umax, PSFUPs);
```

After all PSFUPs are discovered, the proposed mining algorithm is then executed to find the actual SFUPs from PSFUPs. The proposed mining algorithm is shown in Algorithm 2.

Algorithm 2. Proposed mining algorithm

Input: $PSFUPs$, the set of potential SFUPs.
Output: $SFUPs$, the set of skyline frequent-utility itemsets.

```
1  for each X ∈ PSFUPs do
2      for each Y ∈ PSFUPs do
3          if u(X) ≥ u(Y) ∧ f(X) > f(Y)||u(X) > u(Y) ∧ f(X) ≥ f(Y) then
4              SFUPs ← X ∪ SFUPs;
5              remove Y from PSFUPs;
6  return SFUPs;
```

4 Experimental Results

In this section, substantial experiments were conducted to evaluate the proposed algorithm for mining SFUPs on several datasets. Note that only one existing algorithm called SKYMINE [10] was proposed to mine the SFUPs by considering both the frequency and utility of the itemsets. Four real-world datasets called chess [1], mushroom [1], foodmart [18] and retail [1] and one synthetic T10I4N4KDXK dataset [4] were used in the experiments to evaluate the performance of the proposed algorithm. In the experiments, the program is terminated if the runtime exceeds 2×10^2 s or the memory leakage occurred.

4.1 Runtime

The proposed algorithm was compared with the state-of-the-art SKYMINE algorithm [10] on five datasets and the results are shown in Table 1.

Table 1. Runtime of the compared algorithms.

	Proposed mining algorithm	SKYMINE
Chess	202.69 s	-
Mushroom	10.57 s	202.82 s
Foodmart	2.64 s	98.59 s
Retail	117.57 s	-
T10I4N4KD100K	57.32 s	346.45 s

From Table 1, it can be observed that there are no results of the SKYMINE algorithm on chess and retail datasets. The reason is that the memory leakage occurred for those two datasets and the algorithm is terminated. We also can observe that the proposed algorithm outperforms the SKYMINE algorithm and generally up to almost one or two orders of magnitude faster than the SKYMINE algorithm. The proposed algorithm always has better results than that of SKYMINE algorithm since the proposed algorithm can directly exact the actual utility of the itemsets with only two scans of dataset. The SKYMINE algorithm generates, however, many redundant candidates with the overestimated value of the itemsets. Thus, the SKYMINE algorithm requires more time for generating the candidates and determining the actual SFUPs.

4.2 Memory Usage

In this section, the memory usage of the proposed algorithm and the SKYMINE algorithm were also compared. The results on all datasets are shown in Table 2.

From Table 2, there are no results for the memory usage of the SKYMINE algorithm on chess and retail datasets since the memory leakage occurred.

Table 2. Memory usage of the compared algorithms.

	Proposed mining algorithm	SKYMINE
Chess	137.00 M	-
Mushroom	188.87 M	423.80 M
Foodmart	32.13 M	667.46 M
Retail	143.68 M	-
T10I4N4KD100K	234.30 M	446.69 M

It can be clearly seen that the proposed algorithm requires less memory compared to the SKYMINE algorithm on all datasets. The reason is that the SKYMINE algorithm mines, however, the SFUPs based on UP-Growth algorithm, which requires to generate the numerous candidates and it is not an efficient way to mine the SFPUs.

5 Conclusion

In this paper, more efficient algorithms are proposed to mine a set of skyline frequent-utility itemsets without candidate generation by considering the frequency and utility factors. The designed algorithms rely on the utility-list structure and the *umax* array for mining SFUPs. A pruning strategy are used to early prune the unpromising candidates for deriving the SFUPs. Based on the designed algorithms, it is unnecessary to pre-defined the minimum support or utility thresholds but the set of useful and meaning information can be returned and discovered. Substantial experiments were conducted on both real-life and synthetic datasets to asses the performance of the proposed algorithm in terms of runtime and memory usages.

Acknowledgment. This research was partially supported by the National Natural Science Foundation of China (NSFC) under grant No. 6150309.

References

1. Frequent itemset mining dataset repository (2012). http://fimi.ua.ac.be/data/
2. Afrati, F.N., Koutris, P., Suciu, D., Ullman, J.D.: Parallel skyline queries. Theory Comput. Syst. **57**(4), 1008–1037 (2015)
3. Agrawal, R., Srikant, R.: Fast algorithm for mining association rules. In: International Conference on Very Large Data Bases, pp. 487–499 (1994)
4. Agrawal, R., Srikant, R.: Quest synthetic data generator (1994). http://www.Almaden.ibm.com/cs/quest/syndata.html
5. Ahmed, C.F., Tanbeer, S.K., Jeong, B.S., Le, Y.K.: Efficient tree structures for high utility pattern mining in incremental databases. IEEE Trans. Knowl. Data Eng. **21**(12), 1708–1721 (2009)

6. Borzsonyi, S., Kossmann, D., Stocker, K.: The skyline operator. In: International Conference on Data Engineering, pp. 421–430 (2001)
7. Chan, R., Yang, Q., Shen, Y.D.: Mining high utility itemsets. In: IEEE International Conference on Data Mining, pp. 19–26 (2003)
8. Chan, C.Y., Jagadish, H.V., Tan, K.L., Tung, A.K.H., Zhang, Z.: Finding k-dominant skylines in high dimensional space. In: ACM SIGMOD International Conference on Management of Data, pp. 503–514 (2006)
9. Chomicki, J., Godfrey, P., Gryz, J., Liang, D.: Skyline with presorting. In: International Conference on Data Engineering, pp. 717–720 (2003)
10. Goyal, V., Sureka, A., Patel, D.: Efficient skyline itemsets mining. In: The International C* Conference on Computer Science & Software Engineering, pp. 119–124 (2015)
11. Grahne, G., Zhu, J.: Efficiently using prefix-trees in mining frequent itemsets. In: IEEE ICDM Workshop on Frequent Itemset Mining Implementations (2003)
12. Han, J., Pei, J., Yin, Y.: Mining frequent patterns without candidate generation. In: ACM SIGKDD International Conference on Management of Data, pp. 1–12 (2000)
13. Kossmann, D., Ramsak, F., Rost, S.: Shooting stars in the sky: an online algorithm for skyline queries. In: International Conference on Very Large Data Bases, pp. 275–286 (2002)
14. Lin, C.W., Hong, T.P., Lu, W.H.: The pre-FUFP algorithm for incremental mining. Expert Syst. Appl. **36**(5), 9498–9505 (2009)
15. Lin, C.W., Hong, T.P., Lu, W.H.: An effective tree structure for mining high utility itemsets. Expert Syst. Appl. **38**(6), 7419–7424 (2011)
16. Liu, Y., Liao, W., Choudhary, A.: A two-phase algorithm for fast discovery of high utility itemsets. In: Ho, T.B., Cheung, D., Liu, H. (eds.) PAKDD 2005. LNCS (LNAI), vol. 3518, pp. 689–695. Springer, Heidelberg (2005). doi:10.1007/11430919_79
17. Liu, M., Qu, J.: Mining high utility itemsets without candidate generation. In: ACM International Conference on Information and Knowledge Management, pp. 55–64 (2012)
18. Microsoft, Example database foodmart of Microsoft analysis services. http://msdn.microsoft.com/en-us/library/aa217032(SQL.80).aspx
19. Papadias, D., Tao, Y., Seeger, B.: Progressive skyline computation in database systems. ACM Trans. Database Syst. **30**(1), 41–82 (2005)
20. Park, J.S., Chen, M.S., Yu, P.S.: An effective hash based algorithm for mining association rules. In: ACM SIGMOD International Conference on Management of Data, pp. 175–186 (1995)
21. Podpecan, V., Lavrac, N., Kononenko, I.: A fast algorithm for mining utility-frequent itemsets. In: International workshop on Constraint-based Mining and Learning, pp. 9–20 (2007)
22. Savasere, A., Omiecinski, E., Navathe, S.: An efficient algorithm for mining association rules in large databases. In: International Conference on Very Large Databases, pp. 432–444 (1995)
23. Tan, K.L., Eng, P.K., Ooi, B.C.: Efficient progressive skyline computation. In: International Conference on Very Large Data Bases, pp. 301–310 (2001)
24. Tseng, V.S., Shie, B.E., Wu, C.W., Yu, P.S.: Efficient algorithms for mining high utility itemsets from transactional databases. IEEE Trans. Knowl. Data Eng. **25**(8), 1772–1786 (2012)

25. Yao, H., Hamilton, H.J., Geng, L.: A unified framework for utility-based measures for mining itemsets. In: ACM SIGKDD International Conference on Utility-Based Data Mining, pp. 28–37 (2006)
26. Yeh, J.-S., Li, Y.-C., Chang, C.-C.: Two-phase algorithms for a novel utility-frequent mining model. In: Washio, T., et al. (eds.) PAKDD 2007. LNCS (LNAI), vol. 4819, pp. 433–444. Springer, Heidelberg (2007). doi:10.1007/978-3-540-77018-3_43