FSKYMINE: A Faster Algorithm For Mining Skyline Frequent Utility Itemsets

Hung Manh Nguyen
Le Quy Don Technical University
Hanoi, Vietnam
manhhungk12@mta.edu.vn

Anh Viet Phan
Le Quy Don Technical University
Hanoi, Vietnam
anhpv@mta.edu.vn

Lai Van Pham
Military Science and Technology Institute
Hanoi, Vietnam
garry@cinnamon.is

***Abstract*—Skyline frequent utility itemsets mining is a challenging task in frequent itemsets mining and plays an important role in many data mining applications. Previous studies presented two algorithms, namely SKYMINE and SKYMINE2, to mine Skyline Frequent Utility Itemsets (SFUIs). In which, the SKYMINE2 based on utility-list data structure is a state-of-the-art algorithm. However, the SKYMINE2 remains computationally expensive because the algorithm generates numerous utility lists, join operations, and potential SFUIs. In this paper, we propose a more effective algorithm to mine the SFUIs based on extent utility list data structure and using a new strategy to prune the potential skyline frequent utility itemsets. Notably, experimental results with four datasets show the proposed algorithm reduces the number of utility lists, join operations, potential SFUIs effectively and outperforms the SKYMINE2 in terms of runtime.**

***Index Terms*—Within-project Prediction, Cross-project Prediction, Convolutional Neural Networks, Transfer Learning.**

I. Introduction

Mining itemsets from massive data of organizations to discover valuable information has received attention from both academia and industry. The task is to extract the frequently appearing items in transactions from a database. Determining such items helps managers prepare strategic plans to serve customer demands and boost profits. To mine itemsets in real applications, we need to consider not only the frequency but also other utility aspects like profit. Thus, among studies in this field, proposed algorithms have paid more attention to detecting the profitable itemsets in a quantitative database [9].

Frequent itemset mining (FIM) is the traditional approach that uses a pre-specified minimum support ($minSup$) to generate frequent itemsets (FIs) [1], [3], [7], [8], [15]. Regarding this, an itemset is defined as frequent if the number of its occurrences in transactions exceeds the $minSup$. This approach has two issues including 1) the difficulty to specify the $minSup$ value and 2) ignoring the item utilities like weight, unit profit, and quantity, meanwhile such aspects are preferable in practical problems.

Unlike FIM, high-utility itemset mining (HUIM) uses both profits and quantities of products in transactions to extract actual utility values of itemsets [2], [10]. Both FIM and HUIM require choosing an appropriate threshold for minimum support and utility when retrieving the frequent itemset candidates. A small value of the threshold leads to a large set of frequent itemsets (FIs) or high-utility itemsets (HUIs). If we increase the threshold, the FIs or HUIs set may be empty. To deal with the scenario of many FIs or HUIs, a wide range of methods have been investigated such closed itemsets [5], [11], maximal itemsets [5], [16], closed high-utility itemsets [4], [13], maximal high utility itemsets [12], [14].

To overcome the limitations of FIM and HUIM, Goyal et al. develop the SKYMINE algorithm for skyline frequent utility itemsets mining (SFUIM) [6]. The algorithm first generates skyline frequent utility itemsets and then mines such candidates to extract the itemsets. In the SFUIM, an itemset X is considered as skyline frequent utility itemset if there is no other itemset in the database that has both utility and frequency higher than X. It should be noted that unlike FIM and HUIM, the SKYMINE does not require a threshold and it considers both frequency and utility. Due to such advantages, recent studies have focused and made many efforts to enhance SFUIM algorithms [6], [9]. However, these algorithms are time-consuming because numerous potential SFUIs will be generated.

This paper proposes a new algorithm, so-called FMSFUI [1] (Faster Algorithm For Mining Skyline Frequent-Utility Itemsets) to efficiently mine the SFUIs. To speed up the mining process, we develop a pruning strategy and an extent utility list structure to reduce the numbers of utility lists, join operations and generated potential SFUIs. The experimental results on various databases reveal that our proposed algorithm shows better performance than those of SKYMINE [6] and SKYMINE2 [9] in mining SFUIs.

The remainder of the paper is organized as follows. Section II describes the itemset mining problem, and some definitions related to the SKYMINE algorithm. Our proposed pruning strategy and algorithm are presented in Section III. We analyze experiments and results in Section IV, and conclude in Section V.

II. Background

Given a finite set of items (symbols) $I = \{i_1, i_2, , i_m\}$. A transaction database D is a set of transactions $\{T_1, T_2, , T_n\}$, such that for each transaction T_q, $T_q \subseteq I$ and T_q has a unique

[1]The source code is publicly available at https://github.com/pvanh/FSFUI.git

978-1-7281-5163-2/19/$31.00 ©2019 IEEE

identifier q called its Tid. Each item $i \in I$ is associated with a positive number $prof(i)$, called its external unit profit and each item i_p in the transaction T_q is associated with a quantity $q(i_p, T_q)$ called the internal utility of i_q in the transaction T_q.

A itemset X, such that $X \in I$, contains k distinct items called a k-itemset, in which 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 a transaction database D is denoted as f(X) and defined as the number of transactions containing X, i.e., $f(X) = |X \subseteq T_q \wedge T_q \in D|$.

Definition 2. The utility of an item i_j in the transaction T_q is denoted as $u(i_j, , T_q)$ and defined as total profit of item i_q in transaction T_q, i.e., $u(i_j, , T_q) = q(i_j, T_q) \times pr(i_j)$.

Definition 3. The utility of an itemset X in the transaction T_q such that $X \subseteq T_q$ is denoted as $U(X, T_q)$ and defined as $U(X, T_q) = \sum_{i_j \in \wedge X \subseteq T_q} u(i_j, T_q)$.

Definition 4. The utility of an itemset X in a transaction database D is denoted as $u(X)$ and defined as $u(X) = \sum_{(X \subseteq T_q, \wedge T_q, \subseteq D} u(X, T_q)$.

Definition 5. The transaction utility of a transaction T_q in a transaction database D is denoted as $tu(T_q)$ and defined as the sum of the utility of the items in T_q, i.e., $tu(T_q) = \sum_{X \in T_q} u(x, T_q)$.

Definition 6. The transaction-weighted utility of an itemset X in a transaction database D is denoted as $twu(X)$ and defined as the sum of the transaction utility of transactions containing itemset X, i.e., $twu(X) = \sum_{X \subseteq T_q, \wedge T_q \in D} tu(T_q)$.

The algorithms in FIM, HUIM use above definitions to discover the FIs, HUIs. In SFUIM, the following definitions are used to find SFUIs.

Definition 7 [9]. An itemset X is said to dominate another itemset Y in D, denoted as $X \succ Y iff f(X) \geq f(Y)$ and $u(X) \geq u(Y)$.

Definition 8 [9]. The problem of skyline frequent utility itemset mining is to discorver all skyline frequent utility itemsets. An itemset X in a database D is a skyline frequent utility itemset iff it is not dominated by any other itemset in the database.

Definition 9 [9]. The maximal utility of the frequency value r is denoted as $umax(r)$ and defined as the maximal utility of itemsets having the same frequency value r.

Definiton 10 [9]. An itemset X in a transaction database D is considered as a candidate SFUI (CSFUI) if its frequency is equal to r and non-itemset in the database having higher utility than $u(X)$.

III. Proposed Algorithm For Mining SFUIs

In this section, we propose an algorithm for mining the set of skyline frequent-utility itemsets. A strategy and an extent-efficient utility list structure are also developed to reduce the number of utility lists, the number of join operations and the number of generated potential SFUIs.

Definition 11. The itemset P is extended with item x is denoted as Px and defined as $P\_x = P \cup x$. For example, if $P = \{a, c, e\}$ and $x = \{f\}$ then $P\_x = \{a, c, e, f\}$.

Definition 12. Let $\succ$ be any total order on items from I. A sorted transaction database is denoted as SD and defined as a transaction database, in which the items in transactions are sorted in total order.

Definition 13. The extent utility list of an itemset P_x in a SD is a set of tuples, in which each tuple consists of four fields as $(tid, itemsetutil, itemutil, rutil)$. These fields are defined as follows: the tid is the transaction ID containing the itemset P_x, the $itemsetutil, itemutil$ and $rutil$ respectively are the utility of P in tid, the utility of x and the utility of resting items after x in transaction tid.

Definition 14 (extent utility list). Given a extent utility list of an itemset P_x. The $P\_x.sumitemsetutils$ is defined as the sum of itemsetutil values in extent utility list of P_x, the $P\_x.sumitemutils$ is defined as the sum of itemutil values in extent utility list of P_x, the $P\_x.sumrutils$ is defined as the sum of rutil values in extent utility list of P_x.

Property 1 [9] (sumitemsetutils and sumitemutils). Given an itemset P_x having occurrence frequency is r. If the sum of sumitemsetutils and sumitemutils values of extent utility list of P_x is higher than or equal to $umax(r)$ then P_x is a potential skyline frequent-utility itemset.

Property 2 [9] (sumitemsetutils, itemutils and sumrutils). Given an itemset P_x having occurrence frequency is r. If the sum of sumitemsetutils, sumitemutils and sumrutils values of extent utility list of P_x is less than $umax(r)$ then all extensions of P_x are not SFUIs.

Definition 15. The remaining transaction-weighted utility co-occurrence of pair item x, y such that x and y co-occur in the transaction T in SD, x in front of y in total order is denoted as $rtwuc(x, y, T)$ and defined as the sum of the utility of the resting items in T from item x:

$$rtwuc(x, y, T) = \sum_{i \in T \wedge ifromxintotalorder} u(i, T) \quad (1)$$

Definition 16. The remaining transaction-weighted utility co-occurrence of pair item x, y in a database SD is denoted as $rtwuc(x, y)$ and defined as the sum of the remaining transaction-weighted utility co-occurrence of pair item x, y in all transactions containing both of the item x, y in the database:

$$rtwuc(x, y) == \sum_{(T \in D) \wedge ((x,y) \subseteq T)} rtwuc(x, y, T) \quad (2)$$

A. Pruning Strategy (Estimated Remaining Utility Co-occurrence Pruning - ERUCP)

To mine the SFUIs, the SKYMINE [6] algorithm generates numerous candidates because the algorithm uses the upper bound of the itemsets is overestimated based on the two-phase model, the algorithm SKYMINE2 [9] performs numerous operations of joining two utility lists and generates numerous utility lists and potentials SFUIs. To improve these limitations, we propose a strategy to speed up mining process of the SFUIs as follow:

Lemma 1. Let two extent utility list of two itemsets Px and Py such that Px having occurrence frequency is r, Py having occurrence frequency is r_1. If $min(Px.sumitemsetutils, Py.sumitemsetutils) + rtwuc(x, y)$ is less than $umax(r)$ or $umax(r_1)$ then Pxy and all extensions of Pxy are not SFUIs.

Proof:
For $\forall$ pair x, y:
$u(Pxy) = UTIL(P) + UTIL(xy)$;
Where:
$u(Pxy)$ is the utility of Pxy in the database D,
$UTIL(P)$ is the utility of P in the transactions containing Pxy in the database D,
$UTIL(xy)$ is the utility of $\{xy\}$ in the transactions containing Pxy in the database D.
Therefore, we have:
$u(Pxy) = UTIL(P) + UTIL(xy)$
According to the extent utility list defined in Definition 14, we have: $UTIL(P) \leq min(Px.sumitemsetutils, Py.sumitemsetutils)$;
According to the Definition 16, we have: $UTIL(xy) \leq rtwuc(x, y)$;
Therefore, we have:
$u(Pxy) = UTIL(P) + UTIL(xy) \leq min(Px.sumiutil, Py.sumiutil) + rtwuc(x, y)$.

Such that if $min(Px.sumiutil, Py.sumiutil) + rtwuc(x, y)$ is less than $umax(r)$ or less than $umax(r1)$ then Pxy and all extensions of Pxy are not SFUIs.

Such that if $min(Px.sumiutil, Py.sumiutil) + rtwuc(x, y)$ is less than $umax(r)$ or less than $umax(r_1)$ then Pxy and all extensions of Pxy are not SFUIs.

B. Proposed Algorithm

In this subsection, we proposed a new effective algorithm to mine skyline frequent utility itemsets. At the beginning, the transaction weighted utility (twu) of all 1-itemsets is calculated. In the next step, 1-itemsets are sorted in twu-ascending order. Then, the remaining transaction-weighted utility of pairs item x, y in a database (rtwu(x,y)) is calculated, and finally, the extent utility lists of 1-itemsets are constructed. Finally, we designed FSFUI-Miner algorithm to discover the potential SFUIs. The pseudocode of FSFUI-Miner is shown in Algorithm 1. At line 13, extent utility lists are pruned (according to Lema 1) to reduce the numbers of join operations and generated potential SFUIs. To determine the actual SFUIs from PSFUIs discovered by FSFUI-Miner algorithm, we use Algorithm 2 to mine.

IV. Experimental Results

In this section, we conducted substantial experiments on four datasets to assess the performance of our proposed algorithm FSKYMINE for mining SFUIs. Table I shows statistical figures on the datasets. To the best of my knowledge, there are only two algorithms, i.e., SKYMINE [6] and SKYKINE2 [9], that were presented to mine the SFUIs by considering both

Algorithm 1: FSFUI-Miner

Input : P is the extent utility list of the itemset P;
$ExtPs$ is the set of extent utility lists of the itemsets extended from P;
$umax$ is an array to keep the maximum utility of the varied frequencies;
$rtwuc$ is array to keep the remaining transaction-weighted utility of pair of any two items x, y in database D.
Output: $PSFUIs$ is the set of Ps potential skyline frequent utility itemsets

1 **foreach** $x \in ExtPs$ **do**
2 **if** $(Px.sumitemsetutil + Px.sumitemutil) = umax[f(Px)])$ **then**
3 $PSFUIs[f(Px)] += Px$;
4 **end**
5 **else if** $sum(Px.itemsetutil + Px.itemutil) > umax[f(Px)])$ **then**
6 $umax(f(Px)) \leftarrow sum(Px.itemsetutil + Px.itemutil)$;
7 Remove Py from $PSFUIs[f(Px)]$;
8 $PSFUIs[f(Px)] \leftarrow Px$;
9 **end**
10 **if** $Px.sumIutils + Px.sumItemutils + Px.sumRutils \geq uEmax[f(Px)]$ **then**
11 $exULs \leftarrow null$;
12 **foreach** *(y after x in* $ExtPs$ **do**
13 **if** $(min(Px.sumitemsetutil, Py.sumitemsetutil) + rtwuc(x, y) \geq umax[f(Px)])$ *and* $(min(Px.sumitemsetutil, Py.sumitemsetutil) + rtwuc(x, y) \geq umax[f(Py)])$ **then**
14 $exULs \leftarrow exULs + Construct(P, Px, Py)$;
15 **end**
16 **end**
17 **FSFUI-Miner** $(Px, exULs, umax, PSFUIs)$;
18 **end**
19 **end**

Algorithm 2: Proposed mining algorithm in [9]

Input : $PSFUIs$ is the set of potential $SFUIs$
Output: $SFUIs$ is the set of skyline frequent utility itemsets

1 **foreach** $X \in PSFUIs$ **do**
2 **foreach** $Y \in PSFUIs$ **do**
3 **if** $(u(X) \geq u(Y)$ *and* $f(X) > f(Y))$ *or* $(u(X) > u(Y)$ *and* $f(X) \geq f(Y))$ **then**
4 $SFUIs \leftarrow X \cup SFUIs$;
5 Remove Y from $PSFUIs$;
6 **end**
7 **end**
8 **end**

Algorithm 3: The Construct function

Input : P is the extent utility list of the itemset P;
Px is the extent utility list of the itemset Px;
Py is the extent utility list of the itemset Py.
Output: the extent utility list of Pxy

```
foreach tuple elex ∈ Px do
    if ∃eley ∈ Py and elex.tid = eley.tid then
        elexy = (elex.tid, elex.itemsetutil +
          elex.itemutil, eley.itemUtil, eley.rutil);
        ExtentUListofPxy ←
          X ∪ ExtentUListofPxy;
    end
end
return ExtentUListofPxy;
```

TABLE I
THE DATASETS

Dataset	#transactions	#items
Chess	3,196	75
Mushroom	8,124	119
Foodmart	4,141	1,559
Retail	88,162	16,470

the frequency and utility of the itemsets in the transaction database. According to technicals point of view, SKYMINE2 is the more state-of-the-art algorithm. A computer with core i3 processor of 2.3 GHz, with 4GB RAM running Windows 7 is used in our experiments.

Tables II−V compare FSKYMINE and SKYMINE2 in terms of running time to produce the same high-utility item sets. The "Ratio" columns show the ratios of SKYMINE2's values to those of FSKYMINE. As can be seen in Table II, FSKYMINE runs much faster than SKYMINE2 on all databases. Specially, one Retail, the running time of FSKYMINE is 6 times less than that of SKYMINE2. Applying the pruning strategy (Lemma 1) results in a significant reduction of join operations, utility lists and potential SFUIs in comparison with SKYMINE2 (III−V). This helps to reduce the running time as seen as in Table II.

TABLE II
THE RUNNING TIME OF THE ALGORITHMS (MS)

Dataset	FSKYMINE	SKYMINE2	Ratio
Chess	**279,887**	447,163	1.6
Mushroom	**10,627**	18,051	1.7
Foodmart	**374**	840	2.2
Retail	**37,521**	234,159	**6.2**

TABLE III
THE NUMBER OF JOIN OPERATIONS

Dataset	FSKYMINE	SKYMINE2	Ratio
Chess	14,196,256	15,433,526	1.09
Mushroom	528,200	810,321	1.53
Foodmart	791	1,306,272	1651.45
Retail	4,671,121	506,089,424	108.34

TABLE IV
THE NUMBER OF UTILITY LIST OR EXTENT UTILITY LIST

Dataset	FSKYMINE	SKYMINE2	Ratio
Chess	14,196,331	15,433,601	1.09
Mushroom	528,319	810,440	1.53
Foodmart	2,350	1,307,831	556.52
Retail	4,687,591	506,105,894	107

TABLE V
THE NUMBER OF POTENTIAL SFUIS

Dataset	FSKYMINE	SKYMINE2	Ratio
Chess	3,074	3,077	1
Mushroom	1,385	1,385	1
Foodmart	24	24	1
Retail	803	826	1.03

Our proposed algorithm is specially efficient for scenarios of numerous items that normally appear in real-world applications. On *Foodmart*, FSKYMINE produces the number of join operations and utility list less than those of SKYMINE2 1,651 and 556 times. Similarly, such ratios are 108 and 107 on *Retail*. As a result, FSKYMINE significantly outperforms SKYMINE2 on these datasets.

V. CONCLUSION

In this paper, we proposed a very fast algorithm namely FSKYMINE to mine skyline frequent utility itemsets. FSKYMINE is specially efficient for datasets with many items in comparison with SKYMINE2, the state-of-the-art algorithm. The proposed algorithm relays on a strategy of ERUCP (Estimated Remaining Utility Co-Occurrence Pruning) to reduce the number of join operations in mining process skyline frequent utility itemsets using extent utility list data structure, and the number of generated potential SFUIs. Our experiments on four datasets have confirmed the assumption.

ACKNOWLEDGEMENTS

This research is funded by the Vietnam National Foundation for Science and Technology Development (NAFOSTED) under grant number 102.05-2018.306.

REFERENCES

[1] Rakesh Agrawal, Ramakrishnan Srikant, et al. Fast algorithms for mining association rules. In *Proc. 20th int. conf. very large data bases, VLDB*, volume 1215, pages 487–499, 1994.

[2] Chowdhury Farhan Ahmed, Syed Khairuzzaman Tanbeer, Byeong-Soo Jeong, and Young-Koo Lee. Efficient tree structures for high utility pattern mining in incremental databases. *IEEE Transactions on Knowledge and Data Engineering*, 21(12):1708–1721, 2009.

[3] Philippe Fournier-Viger, Espérance Mwamikazi, Ted Gueniche, and Usef Faghihi. Meit: Memory efficient itemset tree for targeted association rule mining. In *International Conference on Advanced Data Mining and Applications*, pages 95–106. Springer, 2013.

[4] Philippe Fournier-Viger, Souleymane Zida, Jerry Chun-Wei Lin, Cheng-Wei Wu, and Vincent S Tseng. Efim-closed: fast and memory efficient discovery of closed high-utility itemsets. In *International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 199–213. Springer, 2016.

[5] Tarek F Gharib. An efficient algorithm for mining frequent maximal and closed itemsets. *International Journal of Hybrid Intelligent Systems*, 6(3):147–153, 2009.

[6] Vikram Goyal, Ashish Sureka, and Dhaval Patel. Efficient skyline itemsets mining. In *Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering*, pages 119–124. ACM, 2015.

[7] Gösta Grahne and Jianfei Zhu. Fast algorithms for frequent itemset mining using fp-trees. *IEEE transactions on knowledge and data engineering*, 17(10):1347–1362, 2005.

[8] Jiawei Han, Jian Pei, and Yiwen Yin. Mining frequent patterns without candidate generation. In *ACM sigmod record*, volume 29, pages 1–12. ACM, 2000.

[9] Jerry Chun-Wei Lin, Lu Yang, Philippe Fournier-Viger, Siddharth Dawar, Vikram Goyal, Ashish Sureka, and Bay Vo. A more efficient algorithm to mine skyline frequent-utility patterns. In *International Conference on Genetic and Evolutionary Computing*, pages 127–135. Springer, 2016.

[10] Mengchi Liu and Junfeng Qu. Mining high utility itemsets without candidate generation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, pages 55–64. ACM, 2012.

[11] Lisheng Ma and Yi Qi. An efficient algorithm for frequent closed itemsets mining. In *2008 International Conference on Computer Science and Software Engineering*, volume 4, pages 259–262. IEEE, 2008.

[12] Bai-En Shie, S Yu Philip, and Vincent S Tseng. Efficient algorithms for mining maximal high utility itemsets from data streams with different models. *Expert Systems with Applications*, 39(17):12947–12960, 2012.

[13] Vincent S Tseng, Cheng-Wei Wu, Philippe Fournier-Viger, and S Yu Philip. Efficient algorithms for mining the concise and lossless representation of high utility itemsets. *IEEE transactions on knowledge and data engineering*, 27(3):726–739, 2014.

[14] Cheng-Wei Wu, Philippe Fournier-Viger, Jia-Yuan Gu, and Vincent S Tseng. Mining compact high utility itemsets without candidate generation. In *High-Utility Pattern Mining*, pages 279–302. Springer, 2019.

[15] Mohammed Javeed Zaki. Scalable algorithms for association mining. *IEEE transactions on knowledge and data engineering*, 12(3):372–390, 2000.

[16] Xin Zhang, Kunlun Li, and Pin Liao. A depth-first search algorithm of mining maximal frequent itemsets. In *2015 Seventh International Conference on Advanced Computational Intelligence (ICACI)*, pages 170–173. IEEE, 2015.