Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **8**, Number **4**, November **2023**

A Utility-frequency Skyline Itemsets Pattern Mining Algorithm with Threshold Restrictions

Yadong Liu

College of Computer Science and Engineering
Shandong University of Science and Technology
Qianwangang Road, 579, Qingdao, China
794010914@qq.com

Shahab Tayeb

Department of Electrical and Computer Engineering
California State University
Engineering East 274, Fresno, USA
tayeb@csufresno.edu

Jimmy Ming-Tai Wu*

Department of Information Management
National Kaohsiung University of Science and Technology
No1, University Rd., Yanchao Dist., Kaohsiung City, 824, Taiwan
wmt@wmt35.idv.tw

*Corresponding author: Jimmy Ming-Tai Wu

Received July 12, 2023, revised August 20, 2023, accepted September 10, 2023.

Abstract. *Skyline frequent utility patterns have been extensively studied. However, the existing skyline pattern mining algorithms are inefficiency and time consuming, and can be inadequate in practical applications, as sometimes traditional skyline pattern may obtain points that are relatively extreme in one dimension. These one-sided extreme points are not the points that the user would like to see in many applications, and the user prefer to see the skyline itemsets with relatively balanced two dimensions. Therefore, this paper proposes a new pattern of skyline mining with threshold settings. In the process of new skyline pattern mining, multiple thresholds are used to exclude extreme points that do not satisfy the user's needs in certain dimensions, while also greatly reducing the number of search size and improving the speed of searching the target itemsets. In addition, the intelligent recommendation system of an online shopping website is taken as an example to demonstrate the application potential of the new pattern in a service AI system. In this paper, the UFmax array structure is applied so that it can be used not only to get the itemsets with the highest utility at the same frequency, but also to exclude the set of items whose utility and frequency both do not satisfy the threshold. The SFUTPMiner algorithm is introduced to make skyline pattern mining more accurate and efficient. Experiments were conducted on six databases respectively, and the proposed algorithm was compared with the current popular mining algorithm. The results show that the designed new pattern of skyline mining with threshold settings not only provides a streamlined and optimised set of result items, but also surpasses the previous traditional pattern in the aspect of runtime, search space and memory consumption.*
Keywords: Data mining, Skyline Pattern, Pattern mining, Utility list

1. **Introduction.** Data mining is the process of searching for information concealed in massive data through algorithms and being able to extract new, valid and relevant knowledge from the raw data [1, 2, 3]. Data mining is very extensive used in industries, a very common example is behavioral data analysis of shoppers' shopping habits. By mining the data in the user's historical shopping basket, the system can gain a deep understanding of the customer group and recommend products that are more in line with their habits. Medical information data is mined and analysed to provide decision solutions for disease prediction and diagnosis. Frequent itemsets mining(FIM) has been extensively studied in the field of data mining and has a very solid relevance in practical applications [4, 5, 6, 7]. Extracting these variables from the data can support the required decision-making. FIM can be used in shopping website orders, web traffic, AI-based data analysis and decision making systems and other fields.

Indeed, it has been recognized that considering frequency alone is often insufficient in many applications. While frequency provides valuable insights into the occurrence of itemsets, there are other important factors that need to be taken into account for a comprehensive analysis. For example, in shopping data analysis, the daily frequency of mobile phones sold in a large shopping mall is significantly lower than the frequency of bread sold in the mall, but the unit profit of the former is much higher than that of the latter, so considering frequency alone in data mining is not comprehensive enough. As a result, research on high-utility itemsets mining (HUIM) has steadily gained more attention [8, 9, 10, 11, 12, 13, 14]. The purpose of HUIM is to find the itemsets that exceeds the minimum utility threshold given by the user. Chan *et al.* [9] first investigated the HUIM issue, and later Yao *et al.* [15] discovered HUI by pairing the number of items and unit profits. Then Liu *et al.* [11] printed a transaction-weighted utilization (TWU) model which satisfied the downward closure feature and solved the problem of utility values not satisfying the antimonotone property in the mining process. Later Lin *et al.* [10] printed an high-utility pattern (HUP) tree structure. After that several algorithms were proposed [16, 17, 18, 19, 20], which greatly improved the efficiency of HUI mining.

All of the above algorithms are designed to provide a group of regulations to help users make rational and effective decisions, but sometimes users may prefer to find more concise rules take action more quickly. Top-k association rule mining [21] and HUI mining [22] were created to discover more succinct regulations. FIM can be leveraged to find itemsets whose frequencies satisfy a minimum frequency threshold, and HUI mining can be leveraged to find itemsets whose utilities satisfy a minimum utility threshold. However, in certain application sites it is often not enough to focus on a single aspect. It is not possible to meet the user's usage needs. For example, a shopping website has thousands of commodities in its database, and each commodity has its own price and sales volume. For the same commodity, its price and sales volume may have some relationship. Items with higher prices tend to sell in fewer quantities than items with lower prices. Everyone has different acceptance of price, and different threshold settings can provide users with different consumption levels with high-selling or high-scoring products that meet their expectations. Therefore, a qualified recommendation system should at least be able to find products that meet user needs in terms of price and sales volume from the database, and make targeted recommendations. To address the limitations of previous patterns, Goyal *et al.* [23] first printed a method to get hold of skyline frequent utility (SFU). The algorithm procedure is found on the UP-tree structure [24] and requires the generation of a large amount of candidate objects. Pan *et al.* [25] printed the SFUPMiner to increase the effectiveness of acquiring skyline patterns. Lin *et al.* [25] further printed a very efficient algorithm is to discovering SFUPs, which is implemented by using the utility list structure. However, original algorithms are nevertheless very time-consuming when scanning the

database and can be inadequate in practice. The characteristics of skyline itemsets make it time-consuming and resource-consuming to find them even in databases with small amounts of data. And as the dimensions increase, the difficulty of searching for skyline continues to double. Without the constraint of threshold, the result set obtained by the traditional skyline search algorithm contains some single-dimensional extreme points with very high utility but very low frequency, or very high frequency but very low utility. These one-dimensional extreme points are not what the user wants to see. For example, for most passengers, they will neither choose hotels with very low prices but very poor comfort, nor hotels with very high comfort but also extremely expensive prices. They prefer to get hotels with less cost under certain comfort conditions or hotels with higher comfort under certain cost restrictions. Moreover, obtaining these extreme points that are not practical enough will consume a lot of time and resources, resulting in a waste of performance. In light of the drawbacks mentioned earlier, we propose a novel approach for setting thresholds in the skyline pattern mining process. This new approach effectively addresses these limitations and offers several benefits. By applying the proposed thresholds, we can exclude extreme points that do not align with the user's requirements, resulting in a more refined acquisition of skyline patterns. Moreover, this approach significantly reduces the size of the search space, overcoming the resource consumption issues associated with skyline pattern mining. As a result, our method enhances the search performance specifically targeting the desired itemsets. The main contributions of our work can be summarized as follows:

1. This paper proposed a new skyline frequency-utility with thresholds pattern (SFUTP), which is based on the traditional skyline pattern by adding limits to both the utility and frequency dimensions to exclude certain single-dimensional extremes where the utility or frequency does not meet the user's needs.
2. The UFmax array structure was proposed, through which the threshold value can be used to filter the searched itemsets, greatly improving the reduction of search space and improving the efficiency of algorithm execution.
3. Based on the new pattern of skyline mining with thresholding, the SFUTPMiner algorithm is proposed, which uses UFmax arrays and threshold settings to achieve fast mining of the search space.

2. **Related Work.**

In this section, we will introduce the related research content regarding high-utility itemset mining and skyline itemset mining.

2.1. **High-utility Itemsets Mining.**

In the research, FIM has always been concerned by the data mining industry [26, 27, 28, 17, 2]. FIM is crucial as the initial stage of association rule mining. Researchers divide the frequent itemsets mining methods into three categories. For the hierarchical growth model, the most common algorithm is Apriori algorithm [5], but the generation of candidate sets in each layer is cumbersome and has drawbacks. To enhance the effectiveness of mining, a method of frequent pattern growth (FP-growth) is also proposed [7], which is to mine FIS from FP-Tree by improving it. In the Eclat model [7, 29], a vertical tid-list database is used, and mining is performed by depth-first search. However, the problem solved by these algorithms is limited to the frequency of the itemsets and does not take into account other factors such as its weight and profit, which is clearly not sufficient in practice. To address the shortcomings of FIM, HUIM has gradually become the focus of researchers' attention as an extension to FIM, itemsets whose utility is greater than the minimum threshold given by the user are called high-utility itemsets. In the field of high utility itemset (HUI) mining, Yao *et al.* [12] proposed an efficient algorithm. However, the itemsets discovered by their algorithm were

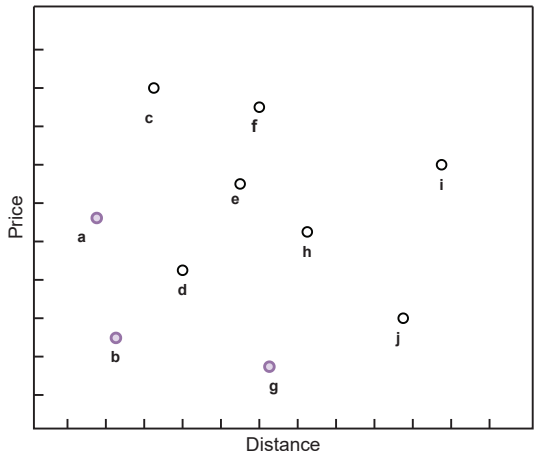

FIGURE 1. Example of skyline points.

found to be imperfect due to the inherent limitations of the downward closure property. To address this issue, Liu *et al.* [11] introduced the concept of transaction-weighted utility (TWU). This approach effectively preserves the transaction-weighted downward closure (TWDC) property, enhancing the accuracy and completeness of HUI mining results. However, the TWU model is not efficient. For this reason Liu and Qu [30] later proposed a new method stemmed from the utility-list structure called HUI-Miner, which can improve the performance of HUI searching. Afterwards, based on the utility-list structure, researchers conducted a variety of optimization [16, 17, 18, 19, 31, 24, 20], which further improved the searching efficiency of HUI.

Frequent itemsets mining can find itemsets with a frequency greater than a minimum frequency threshold, and HUI mining can find itemsets with a utility greater than a minimum utility threshold. However, both can only be searched in a single dimension. To address this drawback by combining the two, Yeh *et al.* [32](2007) first posed a two-stage algorithm to discover itemsets where each itemset's utility and frequency are greater than the user given minimum utility threshold or minimum frequency threshold. Later, Podpean *et al.* [33](2007) developed a fast algorithm to search the high utility and high frequency itemsets, however the algorithm was not efficient and the resulting itemsets were not streamlined enough to help the user make the fastest decision.

2.2. **Skyline Itemsets Mining.** The concept of a skyline can be understood as a unique collection of points that possess a special characteristic. Specifically, each point in the skyline set is not completely dominated by any other point across multiple dimensions. Because it only returns non-dominated points as solutions to decisions, it is important that it behaves in a large database. Let us suppose that there is an itemset containing n items, and that the skyline of these n items are those that are not dominated by any other items. That is, there are no items that exceed their skyline in all dimensions. This means that these items are better than all other items in at least one dimension. In real life applications there is a significant relationship between distance from the city centre and the price of housing, with housing closer to the city centre tending to be more expensive, and housing becoming less expensive as the distance from the city centre increases. Figure 1 provides a good illustration of this.

In Figure 1, the X-axis coordinates indicate the distance of the housing area from the city centre and the Y-axis indicates the price of the housing. It is obvious to observe through the graph that the skyline points in this dataset are {a,b,g}. Considered in two dimensions, only these points are the non-dominant points in this dataset.

Kung *et al.* [34](2005) initial introduced the concept of skyline. Borzsonyi *et al.* [35](2001) introduced the skyline operation for database contexts. Chomicki *et al.* [36](2003) subsequently proposed an improved version of block nesting loops to improve performance by using a specific order of tuples. Tan *et al.* [37](2001) proposed an algorithm for progressively outputting skyline points. Papadias *et al.* [38](2005) proposed a nearest neighbour search based branch-and-bound skyline (BBS) algorithm that implements a single visit to find skyline points. Later Lin et al. [39] introduced two algorithms, namely SKYFUP-B and SKYFUP-D, designed specifically for mining SFUI, these algorithms employ different traversal strategies to explore the search space efficiently, and the SFUPMiner algorithm [40] that employs an efficient utility-list structure to efficiently mine SFUP without generating candidate sets . In addition, the Umax array was further developed to maintain maximum utility at the frequency of occurrence. Then, Song *et al.*[41](2021) proposed a high-performance skyline itemset mining algorithm that can be filtered by utility by improving the array structure of the original skyline storage. By incorporating utility filtering, the SFUI_UF algorithm can efficiently eliminate unpromising itemsets early in the mining process. This filtering technique allows the algorithm to focus on itemsets that have higher utility or potential to be part of the skyline pattern, thereby reducing the search space.

The original traditional skyline pattern has a number of drawbacks due to the lack of threshold constraints. The first is that there may be extreme points in the result set of a search in a single dimension, such as a situation where the value of the frequency dimension is very high but the value of the utility dimension is extremely low, which is equivalent to a situation where the sales of an item in a shopping mall are very hot but the total revenue is extremely low, which is not what the user wants to see. Therefore, the single-dimensional extreme items in the result set may have an adverse effect on the user's decision-making judgment. Secondly, the original skyline pattern mining will search the complete database and construct a large number of utility lists, which will cause the traditional skyline pattern algorithm to consume a lot of time and memory, and the efficiency is low. Therefore, we propose a new skyline mining pattern. in the paper, the pattern takes the minimum utility threshold and the minimum frequency threshold as restrictions, and under the appropriate threshold setting conditions, it can not only exclude the one-dimensional extreme items and obtain the final result set that meets the actual needs of users. It can also greatly reduce the data search space that the algorithm needs to search, reduce the amount of operations and improve the execution efficiency of the algorithm.

3. **Preliminary and Problem Statement.**

3.1. **Preliminaries.**

Suppose we have a finite set of itemsets denoted by I as $\{i_1, i_2, ..., i_m\}$. An itemset X is a subset of I, and if it contains k items, it is referred to as a k-itemset. On the other hand, let D be the transaction database consisting of transactions $\{T_1, T_2, ..., T_n\}$, where each transaction T_q is a subset of I.

To illustrate this, we can consider a database D represented in Table 1. In this table, each transaction is identified by its unique transaction ID (TID), and the quantities column represents the frequency of each item within that transaction. Additionally, Table 2 displays the unit profit associated with each item.

Definition 3.1. *The number of times itemset X appears in database D is defined as f(X):*

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

For example, in the Table 1 database, $f(A)= 4$, $f(AD)= 3$.

Table 1. A transaction database.

TID	Items:quantities
T_1	(A,1) (D,6) (E,1) (C,2)
T_2	(A,5) (D,1) (C,1) (F,1)
T_3	(A,10) (E,3) (C,6)
T_4	(B,2) (A,5) (D,6) (E,1) (C,1) (F,1)
T_5	(B,4) (D,3) (E,1) (C,3)
T_6	(B,2) (E,4) (C,2)
T_7	(D,1) (E,1) (C,1)

Table 2. A profit table.

Item	profit
A	1
B	2
C	1
D	2
E	3
F	5

Definition 3.2. *The $u(i_j, T_q)$ is defined as the utility of item i_j in transaction T_q, and its formula is expressed as follows:*

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

For example, it can be derived from T_1 in Table 1 that the utility of the item $\{C\}$ can be computed as $u(C, T_1) = q(C, T_1) \times pr(C) = 2 \times 1 = 2$.

Definition 3.3. *The utility of an itemset X in a transaction T_q is called as $u(X, T_q)$ and defined as:*

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

For example, in the Table 1 database, $u(AD, T_1) = u(A, T_1) + u(D, T_1) = 1 + 12 = 13$.

Definition 3.4. *The utility of itemset X in a transaction database D, can be called as $u(X)$ and defined as:*

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

For example, it can be derived from Table 1 that the utility of itemset $\{AD\}$ in database D can be computed as $u(AD) = u(AD, T_1) + u(AD, T_2) + u(AD, T_4) = 7 + 6 + 11 = 24$.

Definition 3.5. *The utility of transaction T_q in transaction database D is called tu(Tq), which represents the sum of utility of all items in transaction T_q and defined as:*

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

For example, it can be derived from Table 1 that $tu(T_1) = u(A, T_1) + u(C, T_1) + u(D, T_1) + u(E, T_1) = 1 + 2 + 12 + 3 = 18$.

Definition 3.6. *The transaction-weighted utility of an itemset X in a transaction database D is called as $twu(X)$ and defined as:*

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

For example, it can be derived from Table 1 that $twu(A) = T_1 + T_2 + T_3 + T_4 = 18 + 13 + 30 + 30 = 91$.

This is followed by a definition of skyline mining.

Definition 3.7. *For itemset X and itemset Y, if $f(X) \geq f(Y)$ and $u(X) > u(Y)$ or $f(X) > f(Y)$ and $u(X) \geq u(Y)$, then the itemset X governs Y and it is represented as $X \succ Y$.*

For the running example, the itemset $\{AC\} \succ \{ACEF\}$ since $u(AC) > u(ACEF)$ and $f(AC) > f(ACEF)$.

Definition 3.8. *Considering the frequency and utility of two-dimensional factors, if an itemset is not dominated by other itemsets in the database, it is called skyline frequency-utility pattern(SFUP).*

The traditional SFUP has the disadvantage that without a threshold as a lower limit, the result set obtained has extreme itemset, i.e. the single dimension of utility or frequency is too low, so we set thresholds on both frequency and utility dimensions to ensure that the result set obtained is more balanced and more in line with user needs.

Definition 3.9. *A skyline itemset whose itemset frequency is not less than the minimum frequency threshold and whose itemset utility is not less than the minimum utility threshold is called a skyline frequent-utility itemset with threshold pattern (SFUTP).*

Definition 3.10. *Under the condition of meeting the minimum threshold, an itemset X is considered as a potential SFUTP (PSFUTP) if its frequency is equal to r and non-itemset having higher utility than $u(X)$.*

3.2. **Problem Statement.** Based on the above definition, we define the SFUTP mining problem as one that considers frequency and utility factors and discovers non-dominated itemsets in a database on the basis of satisfying frequency and utility thresholds.

In the running example, given a minimum utility threshold of 45 and a minimum frequency threshold of 4, $\{EC\}$ and $\{DEC\}$ can be considered as SFUTPs, where $\{DEC\}$ has a utility of 51 and a frequency of 4 and $\{EC\}$ has a utility of 48 and a frequency of 6. Because neither of them is governed by other sets of items in the database, and they meet the given threshold in the two dimensions of utility and frequency.

4. **Skyline Frequency-Utility-Threshold Pattern Mining.** In this section, the first part gives a practical application example of the skyline pattern, showing the actual application scenario in data driven intelligent recommendation system. In the second part of our study, we proposed an efficient algorithm for mining SFUTP. The algorithm is SFUTPMiner, which utilizes a utility-list structure to facilitate fast and straightforward itemset combination operations. The pseudo code for implementing the SFUTPMiner algorithm is provided in the subsequent sections, offering a comprehensive understanding of its implementation details. This algorithm, based on the new pattern, demonstrates its effectiveness in efficiently mining SFUTP patterns.

4.1. **Application of Skyline Mining Algorithm in Shopping Recommendation System.** The AI-based recommendation system is used in all aspects of life. For example, for short video websites, its intelligent algorithm will recommend different types of short videos at different time periods according to the user's age and preference, so as to increase user usage and increase website revenue. Similarly, when shoppers browse online shopping websites, reasonable product recommendations will increase the user's product click rate and purchase rate. AI, on the other hand, can make targeted recommendations based on the user's consumption ability and user preferences. The model and algorithm used in this paper can provide suitable product information for the recommendation system. What customers care about is the rating and price of the product, and what the merchant cares about is the profit and sales volume of the product. For a product of the same type, the AI recommendation system can provide users with products with high ratings and low prices based on the skyline algorithm. From the perspective of merchants, they can choose to recommend popular products with high profits and high sales volume to increase revenue, or recommend products with high profits and low sales volume to increase sales of unpopular products.

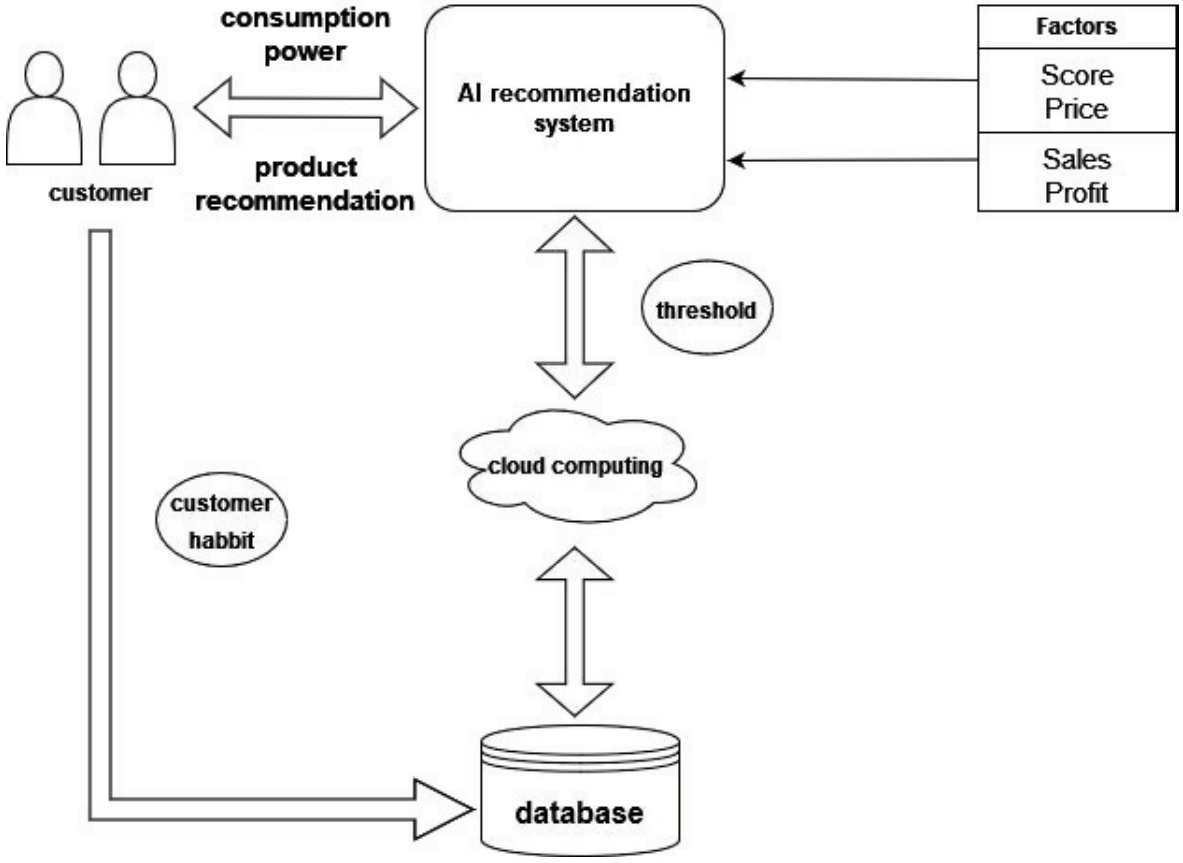

FIGURE 2. Data-driven AI recommendation system.

The AI recommendation system shown in the Figure 2 can infer the user's purchasing ability by learning the user's shopping habits based on multi-dimensional factors, and provide different threshold parameters for the data in the database according to the purchasing ability and purchasing preference, and in many cases, different factors need to be considered to mine the required product information from the database. For example, the matching degree between the user's consumption level and the commodity price level, the favorable rating of the commodity, the sales volume, the commodity profit, the brand

effect, and even the place of shipment of the commodity may affect the recommendation of the system.

4.2. **Search Space and Utility-list Structure.** First we show the search space required for SFUTP mining, as shown in Figure 3, using the example in the form of a set enumeration tree, and in order to sort the itemset mentioned on the utility-list (UL) in ascending order by TWU, we use the common "depth-first search" technique to search the given search space.

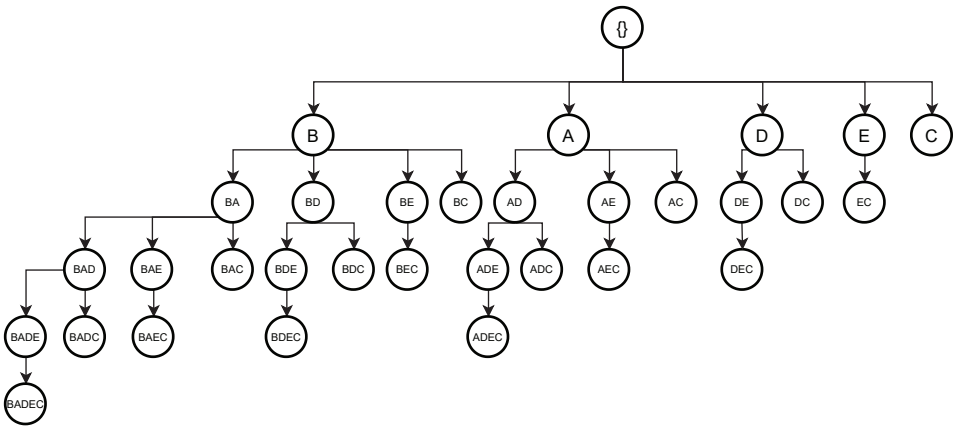

FIGURE 3. Example of a search graph with five items.

B		
tid	iutil	rutil
4	4	21
5	8	12
6	4	14

A		
tid	iutil	rutil
1	1	17
2	5	3
3	10	15
4	5	16

D		
tid	iutil	rutil
1	12	5
2	2	1
4	12	4
5	6	6
7	2	4

E		
tid	iutil	rutil
1	3	2
3	9	6
4	3	1
5	3	3
6	12	2
7	3	1

C		
tid	iutil	rutil
1	2	0
2	1	0
3	6	0
4	1	0
5	3	0
6	2	0
7	1	0

FIGURE 4. The constructed utility-lists.

Let the items in a database be sorted in ascending TWU order. The utility-list [30] of an itemset X is a set of tuples, in which each tuple consists of three parts as (*tid*, *iutil*, *rutil*), where *tid* is the transaction ID containing itemset X, *iutil* is the actual utility of the set X in the transaction, and *rutil* is the sum of the utilities of all items after X in *tid*. For instance, in Table 3, the utility-list structures for (D) and (BCD), their utility lists are $UL.D = (T1, 12, 5), (T2, 2, 1), (T4, 12, 4), (T5, 6, 6), (T7, 2, 4)$ and $UL.DE = (T1, 15, 2), (T4, 15, 1), (T7, 5, 1)$. The structure of the utility-list for 1-itemsets according to the given examples in Figure 3 is shown in Figure 4.

4.3. **UFmax Array Structure.** In this section, we apply an array structure called the Utility-Max array. With this array structure, the itemsets with maximum utility at each frequency value can be obtained by scanning through the database. The subsequent search for SFUTP is facilitated.

We have optimised the original Umax array structure. UFmax is an array of f_{max} elements. where f_{max} is the maximum frequency of the 1-itemset contained in the database D. The UFmax array with frequency i is defined as

$$UFmax[i] = \{u(X) \mid f(X) \geq i\}. \tag{7}$$

To facilitate the filtering of itemsets that satisfy the user-defined minimum utility and frequency thresholds, the UFmax array is initialized with appropriate values. The initial value of *UFmax[1]-UFmax[minfre]* is set to the maximum value. Similarly, the initial value of *UFmax[i]-UFmax[fmax]* is set to the user-defined minimum utility threshold, where fmax corresponds to the maximum frequency of the 1-itemset in the database.

By setting these initial values, the UFmax array becomes a useful tool for efficiently filtering out itemsets that meet the specified minimum utility and frequency thresholds. This initialization process allows for quick identification of itemsets that do not satisfy the user's criteria, thereby reducing the search space and improving the efficiency of the mining process.

For example, if we set the minimum utility threshold to 40 and the minimum frequency threshold to 4, then the UFmax array is initially set as follows: *UFmax[1]=UFmax[2] =UFmax[3]=inf*, *UFmax[4]=UFmax[5]=UFmax[6] =UFmax[7]*=40. for itemset $\{DC\}$, $u(DC) = 42$, $f(DC) = 5$. So $u(DC)$ is compared with *UFmax[5]*, *u(DC)*=42¿*UFmax[5]* =40, so *UFmax[5]* is updated, *UFmax[5]*=42, and $\{DC\}$ is temporarily stored in the list of candidate sets. For itemset $\{ADC\}$, *u(ADC)*=41, *f(ADC)*=3. So $u(ADC)$ is compared with *UFmax[3]*, *u(ADC)*=41¡*UFmax[3]=Inf*, so itemset $\{ADC\}$ does not meet user requirements and is not updated to the candidate set list. That is, the filtering of our desired itemset can be done by the UFmax array with only one comparison.

4.4. **Pruning Strategies.** Indeed, the generation of numerous candidate itemsets in the algorithm, can be a potential issue that affects the efficiency of the algorithm. However, several pruning strategies can be employed to mitigate this problem and reduce the generation of excessive candidate sets during the algorithm's execution.

When a threshold is added to the pattern mining process, the TWU value can be used in the first stage to filter low-utility itemsets. The search space can be further pruned based on the utility-list structure and the UFmax array.

Definition 4.1. *In the transaction database D. the minimum utility of items (MUS) is the maximum utility of 1-items with the highest frequency and defined it as:*

$$MUS = \{u(i) \mid f(i) = f_{max}\}. \tag{8}$$

where i is an item in D and f_{max} is the maximal frequency of all 1-itemsets in D.

For the database shown in Table 1, 1-item {C} have the highest frequency : $f(C) = 7$. Since $u(C) = 40$, $MUS = 40$.

1. During the initial stage, if the TWU of a 1-itemset falls below a specified minutility, it indicates that the 1-itemset and all its extension sets cannot be the SFUTP.

2. Consider an itemset, X. In its utility-list structure, if the sum of iutil is found to be less than $UFmax[f(x)]$, it implies that X is not a SFUTP itemset. Thus, we can safely eliminate X from the search space.

3. If the combined *iutil* and *rutil* values of itemset X in its utility list are lower than *UFmax[f(x)]*, it implies that all supersets of X are not SFUTP. Consequently, these supersets can be pruned and eliminated from the search space.

4.5. **Proposed SFUTP Algorithm.** This paper designs the following algorithm for mining the SFUTP demanded by the user. First, the SFUTPMiner algorithm is used to search the database to get the final list of candidates, then the judge algorithm is used to judge, and finally the real SFUTP is obtained.

The work of Algorithm 1 is to calculate the relevant information of each item before searching, and initialize the *UFmax* array. First calculate the maximum frequency of itemsets in the database and the *twu* value of each 1-item. The value of *fmax* can be used as the upper bound of the capacity of the *UFmax* array initialization and each item is sorted in ascending order of the *twu* value. Then initialize the *UFmax* array, and initialize the *UFmax* array with different index values to different values. For the *UFmax* array whose index value is less than the minimum frequency threshold, the initial value is set to infinity to filter out itemsets that do not meet the frequency threshold. For the *UFmax* array whose index value is greater than the minimum frequency threshold, the initial value is set to the minimum utility threshold to filter out itemsets that do not meet the utility threshold. Next, the PatternSearch algorithm is invoked to carefully mine the search space for potential SFUTPs.

Algorithm 2 is a search procedure for potential SFUTPs, which uses depth-first search. Construct its utility list for itemset M, and scan the information in the utility list to make a judgment. If the sum of its utility is greater than the value of the *UFmax* array whose index is *f(M)*, then put M into the potential list. And update the *UFmax* value. If the sum of its utility and residual utility is greater than the value of *UFmax* for it, it can be considered to expand its utility list and continue to search for its superset. Then continue to call the PatternSearch procedure recursively to determine the new PSFUIs. Final return to PSFUTP list after update.

Algorithm 3 is the final judgment stage, which compares each itemset in the potential SFUTP list to finally obtain the real SFUTP set.

5. **Illustrative Examples.** In this section, we give an illustrative case to explicate the process of our proposed SFUTPMiner algorithm, here using the Table 1 database as an example and Table 2 showing the profit for each item. Assume that the user is given a minimum utility threshold of 45 and a minimum frequency threshold of 4. The first scan of the database gives the maximum frequency of the 1-itemset as f(C) = 7, so MUS = U(C) = 16, and the TWU of all 1-itemsets was calculated by using a database scan with the result $\{TWU : (A) : 86, (B) : 68, (C) : 130, (D) : 87, (E) : 117(F) : 43\}$. Afterwards, the items in the database with TWU less than the given minimum utility threshold or MUS were removed and the set of all remaining items was sorted in ascending order of TWU to complete the reorganization of transaction database. The reorganised database

Algorithm 1 SFUTPMiner

Require:

D, a transaction database ; UFmax, an array structure for maintaining maximum utility of itemsets and for utility and frequency filtering. $minutil$,user-given minimum utility threshold; $minfre$,user-given minimum frequency threshold.

Ensure:

$PSFUTIs$:the potential skyline itemsets with thresholds restrictions.

```
for each transaction T_q ∈ D do
   for each item x ∈ T_q do
      Compute f_max and twu;
   end for
end for
Sort x in twu-ascending order;
for index=1 to minfre-1 do
   Set UFmax[index] to Inf;
end for
for index= minfre-1 to f_max do
   Set UFmax[index] to minutil;
end for
PatternSearch()
Return PSFUTPs;
```

Algorithm 2 PatternSearch

Require:

$M.UL$, the utility-list of the itemset M; exM,the extensions of M; $UFmax$, an array structure.

Ensure:

PSFUTIs: the potential skyline itemsets with thresholds restrictions.

```
for each item i in M'UL do
   if the sum of M.iutil > UFmax(f(M)) then
      Set UFmax(f(M)) = U(M);
      PSFUTIs ← M;
      Dispose itemset N from PSFUTIs if (f(N)==f(M));
   end if
   if the sum of (M.iutil+ M.rutil) ≥ UFmax(f(M)) then
      exM.ulists := M.ulists + construct(ex.ulist);
      PatternSearch(exM.ulists,UFmax,PSFUTPs);
   end if
end for
```

is shown in Table 3. Next, a list of utilities is constructed for each 1-itemset in the database and the results are shown in Figure 4. Next, the SFUTPMiner set the initial value of *UFmax[1]* to *UFmax[3]* to Inf and *UFmax[4]* to *UFmax[7]* to 45. Next, we then use depth-first search to continue mining the filtered search nodes to find other SFUTP itemsets.

Firstly, we start exploring from $\{B\}$, by calculating that its utility value is 16 and its frequency is 3. $U(B) < UFmax[3]$, so $\{B\}$ is not eligible and the PSFUTP table is not updated. Consider again the superset of $\{B\}$, the sum of iutil + rutil of $\{B\}$ can be calculated as 63, which is less than $UFmax[3]$, and it can be known that all of the

Algorithm 3 Judge algorithm

Require:

PSFUTPs,the potential skyline pattern with thresholds restrictions.

Ensure:

List of SFUTPs, the list of final skyline pattern.

```
for each itemset A ∈ PSFUTPs do
  for each itemsetB ∈ PSFUTPs do
    if u(A) ≥ u(B) ⋀ f(A) > f(B) || u(A) > (B) ⋀ f(A) ≥ f(B) then
      SFUTPs ← A ∪ SFUTPs;
      Dispose of B from PSFUTPs;
    end if
  end for
end for
Return List of SFUTPs;
```

Table 3. A transaction database.

TID	Items:quantities
T_1	A:1 D:12 E:3 C:2
T_2	A:5 D:2 C:1
T_3	A:10 E:9 C:6
T_4	B:4 A:5 D:12 E:3 C:1
T_5	B:8 D:6 E:3 C:3
T_6	B:4 E:12 C:2
T_7	D:2 E:3 C:1

supersets of $\{B\}$ have no hope of becoming candidate itemsets, so no extension is needed. Next consider $\{A\}$, which is finally judged to be incompatible with both $\{A\}$ and its superset. The utility of $\{D\}$ is 34 and the frequency is 5, so $U(D) < UFmax[4]$. Since the sum of iutil + rutil of $\{D\}$ can be calculated as 54, which is significantly higher than $UFmax[4]$, $\{D\}$ can be extended, and the supersets of $\{D\}$ are $\{DE\}$ and $\{DC\}$. The utility of $\{DE\}$ is calculated to be 44 and the frequency is 4. $U(DE) < UFmax[4]$, $\{DE\}$ is not qualified. Consider again the superset of $\{DE\}$, which can be extended because the sum of its iutil + rutil is 51. The superset of $\{DE\}$ has $\{DEC\}$, the utility of $\{DEC\}$ is calculated to be 51 and the frequency is 4,$U(DEC) > UFmax[4]$. So $\{DEC\}$ is stored in the set of potential SFUTPs and the value of $UFmax[4]$ updates to 51.

where $\{DEC\}$ has a utility of 51 and a frequency of 4. $\{EC\}$ has a utility of 48 and a frequency of 6. It is judged that neither $\{DEC\}$ nor $\{EC\}$ can completely dominate the other, so both $\{DEC\}$ and $\{EC\}$ are SFUTPs, the skyline frequency utility pattern with threshold we are looking for.

6. **Experimental Evaluation.** Several datasets were used to conduct extensive experiments in this section. Throughout the experiments, different thresholds were set, and rigorous tests were performed. The objective was to assess the performance and efficiency of the new skyline pattern in comparison to existing methods. The algorithm SFUI_UF [41] has been compared with SFUPMiner [40], SKYFUP_B [39] and SKYFUP_D [39] algorithms in the paper. It shows the excellent performance of SFUI_UF, so the SFUI_UF algorithm is the best traditional skyline pattern algorithm at present. Therefore, this experiment chose to compare the SFUTPMiner algorithm in the new pattern with the

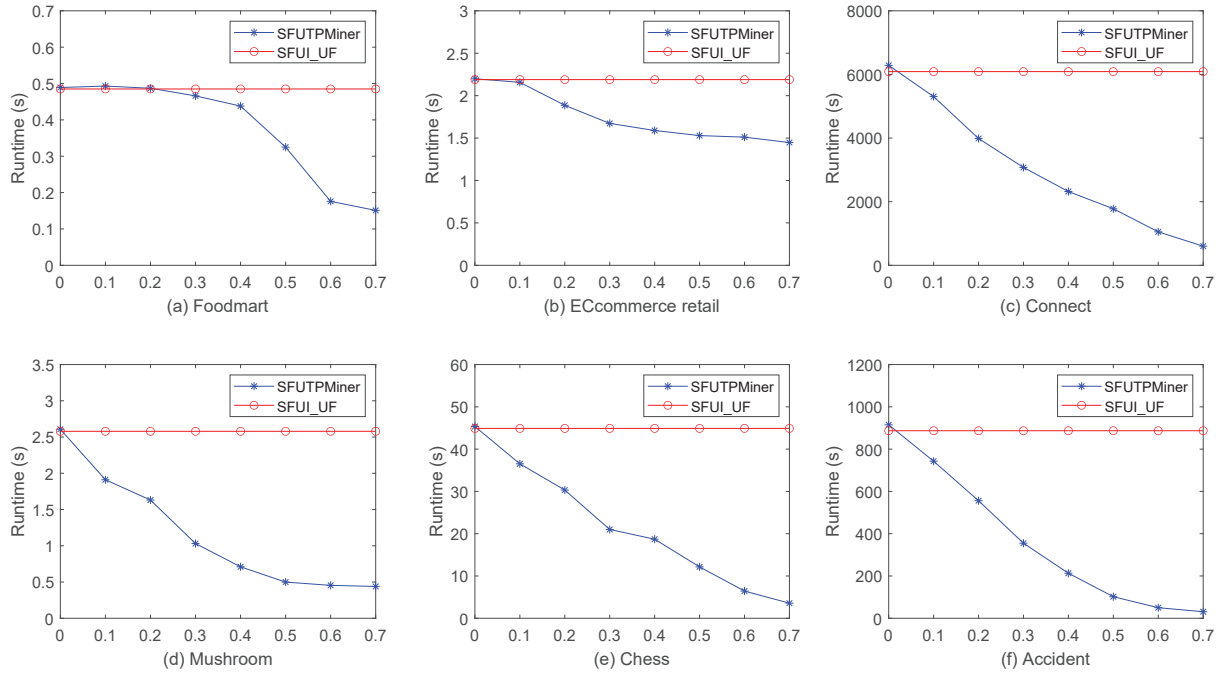

FIGURE 5. Running time comparison.

SFUI_UF algorithm, and set utility threshold and frequency threshold for the SFUTP-Miner algorithm at the same time. Our experiments were performed on a computer with a 6-Core 3.00 GHz CPU and 8 GB memory running 64-bit Microsoft Windows 10. The algorithms in this paper were programmed using the Java language.The six different datasets downloaded from the SPMF [42] were used for the experiments conducted. The six datasets are: foodmart, ecommerce retail, connect, mushroom, chess and accident. The characteristics of each dataset are provided in the Table 4.

TABLE 4. Characteristics of the six datasets.

Dataset	**#Trans**	**#Items count**	**#Density**
Foodmart	4,141	1,559	0.28%
ECommerce retail	14,975	3,468	0.34%
Connect	67,557	129	33.33%
Mushroom	8,416	119	19.33%
Chess	3,196	75	49.33%
Accident	340,183	468	7.22%

6.1. **Runtime.** By conducting experiments using six datasets, we first compare the SFUTPMiner algorithm in our new pattern with the previous most advanced SFUI_UF algorithm in the field of running time by setting different thresholds, and the effects of this comparison are revealed in Figure 5.

In the experiment, SFUI_UF algorithm has no threshold limit. For SFUTPMiner algorithm, To determine the thresholds for the new skyline pattern, we utilized a range of values from 10% to 70% of the maximum frequency and maximum utility observed in the dataset. These thresholds were chosen to ensure comprehensive testing of the pattern's

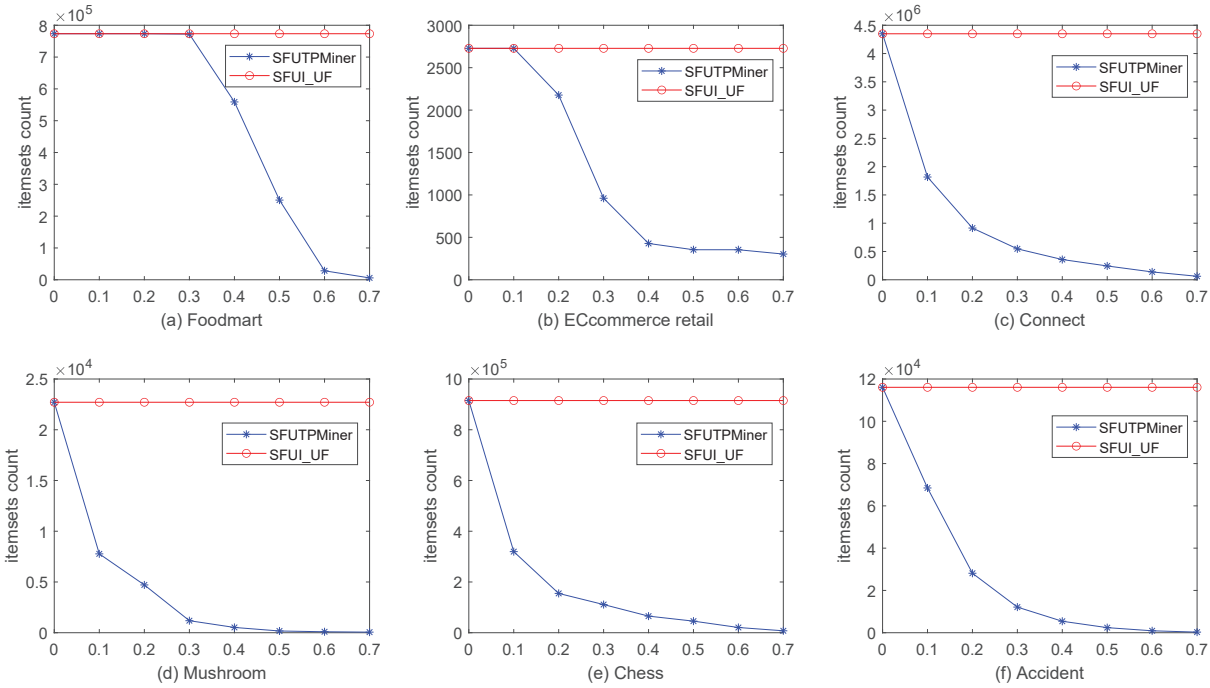

FIGURE 6. Search space size comparison.

effectiveness. By considering different thresholds within this range, we were able to assess the pattern's performance across various levels of frequency and utility in the dataset. The experimental results show that when the threshold is small, the algorithm execution time and SFUI_ UF is almost the same, which may be caused by the low threshold and the failure to effectively prune the search space. When the threshold value surpasses 0.2, the SFUTPMiner algorithm demonstrates significantly shorter execution times compared to the SFUI_UF algorithm. Moreover, as the threshold value increases, the execution time of the SFUTPMiner algorithm decreases considerably. Under the appropriate threshold constraints, The larger the data set, the more obvious the time advantage of the SFUTP new pattern. It can be seen that adding thresholds setting to the new pattern can solve the disadvantages of low efficiency and slow speed in the mining process of the original skyline pattern.

The SFUTPMiner algorithm itself does not have the advantage of faster speed, but because we have added two dimensional threshold limits to the original skyline pattern and proposed a new skyline pattern, namely SFUTP, in the same database search process, the SFUTPMiner algorithm of the new pattern will scan less space than the traditional skyline pattern mining algorithm, and the utility list to be constructed is also less, and its calculation amount will be greatly reduced, Therefore, its consumption time is shorter.

6.2. **Search Space Size.** Figure 6 illustrates the effects of comparing the search space in the execution of the two algorithm. In this comparison, the search space refers to the number of items traversed during the execution of the algorithm.

The experimental results show that in foodmart and ecommerce retail datasets, when the threshold value is small, the search space is almost the same as SFUI_UF, which may be due to the lower threshold value, which does not achieve effective pruning of the search space, but when the threshold value exceeds 0.2, the SFUTPMiner algorithm exhibits a significantly smaller number of scanned itemsets compared to the SFUI_UF algorithm. As the threshold value increases in the SFUTPMiner algorithm, the search

space required gradually decreases. This means that fewer candidate itemsets need to be considered during the mining process. Specifically, when the threshold value reaches 0.7, the search space required by SFUTPMiner becomes significantly smaller compared to the traditional skyline pattern. This reduction in search space is a desirable characteristic of the SFUTPMiner algorithm. It implies that SFUTPMiner can efficiently identify frequent itemsets that satisfy the user-defined threshold, leading to improved mining performance and potentially faster execution times. This observation indicates that SFUTPMiner is more efficient in terms of the number of itemsets it needs to traverse, particularly as the threshold value increases. One of the main reasons for this result is that previous skyline pattern mining algorithms, including SFUI_UF, do not utilize thresholds. These traditional algorithms typically scan and evaluate all itemsets in the dataset without considering any specific threshold values. they prune the data in the whole database and finally obtain the result itemsets. For the SFUTP new pattern, it filters the items in the database that do not meet the threshold in advance by setting the threshold from the beginning. During the mining process of the skyline pattern, SFUTP faces a search space that is essentially the filtered database space based on the threshold. By applying the proposed pruning strategy to this reduced database, the search scope further narrows down. Table 5 shows that although the dataset is very large, the quantity of skyline itemsets is pretty few compared to the large dataset, i.e. most of the itemsets in the dataset are hopeless itemsets, so a good pruning strategy is very important. The SFUTP pattern used in this paper, by setting thresholds on both the frequency and utility dimensions, can prune the searched itemsets very significantly, greatly simplifying the search space.

The Table 5 also reveals the new pattern used in the paper obtains a different amount of resultant sets on certain datasets than those obtained by the traditional pattern algorithm. This is because the pattern we proposed is different from the original pattern. Due to the increased threshold limit, the search space changes when the algorithm performs the search task. In theory, the result set obtained in the new pattern should be less than or equal to the result set of the original pattern. The result set obtained in the SFUTP may be less, but it is more in line with the significance of practical application, Because the SFUTP will exclude some points that are extremely unbalanced in the single dimension, leaving the middle section of the skyline that is more helpful to users. However, we also noticed that in the mushroom dataset, when the threshold setting reaches 0.7, the result set is empty. This is caused by too much pruning of search space due to too large threshold setting, so appropriate threshold setting is crucial for the new pattern.

Table 5. Skyline itemsets count.

thresholds	**0**	**0.1**	**0.2**	**0.3**	**0.4**	**0.5**	**0.6**	**0.7**
Foodmart	1	1	1	1	1	1	1	1
ECommerce retail	2	2	2	2	2	2	2	1
Connect	46	45	42	40	37	34	30	26
Mushroom	17	17	16	10	5	5	4	0
Chess	35	34	32	31	30	28	26	21
Accident	18	18	17	15	14	14	11	8

6.3. **Memory Consumption.** The experiment compared the SFUTPMiner algorithm in new pattern with the SFUI_UF algorithm for different threshold cases and measured the memory usage using the JAVA API, as shown in Figure 7.

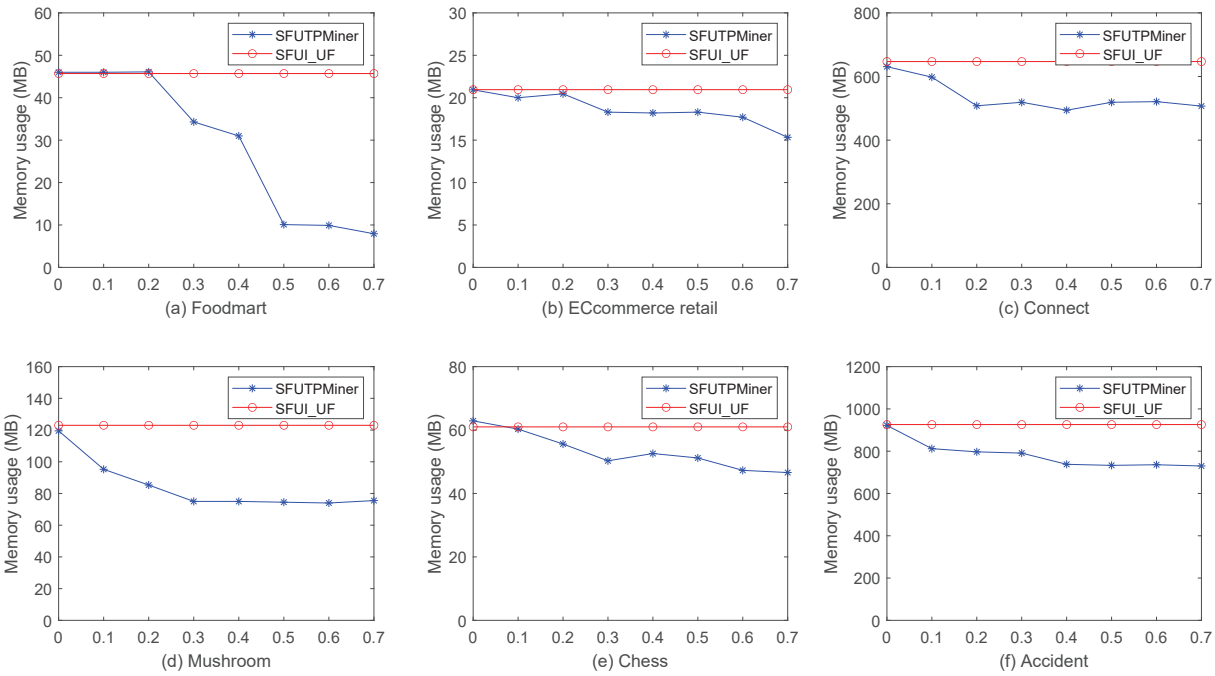

FIGURE 7. Memory consumption comparison.

Looking at the experimental result presented in Figure 7, since one can see that, although the memory consumption does not show a significant decline trend with the increase of the threshold, the memory consumption of the SFUTPMiner algorithm in the new pattern with threshold setting is smaller than that of the traditional skyline pattern in most conditions.

7. **Conclusion.** This paper introduces a novel pattern and a new pattern mining algorithm SFUTPMiner , which incorporates threshold settings. The proposed approach offers several advantages, including significant reduction in search size, improved speed of skyline itemsets search, and enhanced accuracy of search results. We also propose a new application scenario in the paper, combining pattern mining with AI for shopping recommendation systems. The new pattern proposed in this paper can serve the recommendation system very well, infer the user's purchasing ability by learning the user's shopping habits, and provide different threshold parameters for the data in the database according to the purchasing ability and purchasing preference, while considering multi-dimensional Factors are used to mine the required product information from the database and serve AI to make personalized recommendations to customers. While skyline mining is the search for non-dominated itemsets in two dimensions, with the setting of thresholds, certain points that are extreme in a single dimension can be filtered to find quality items that satisfy the user's minimal needs in multiple dimensions. Thanks to the setting of thresholds, a variety of pruning strategies can be applied, thus greatly simplifying the search space. The SFUTPMiner algorithm used in this paper is stemmed from the utility-list structure and applies an optimised UFmax array structure so that it can not only find the itemsets with the highest utility at the same frequency, but also exclude itemsets whose utility and frequency do not meet the threshold, greatly improving the search efficiency. The final experiments, which compared with an excellent algorithm from the past, were performed on various actual datasets. The experiment findings suggest that the new pattern and SFUTPMiner algorithm offer notable advantages in terms

of both time and space efficiency, particularly when dealing with large datasets. It not only solves the drawbacks of slow speed and poor performance of traditional skyline pattern in large datasets, but also greatly optimizes the result itemsets, and can accurately provide users with good quality skyline result itemsets. The new pattern also has defects and limitations. For example, proper threshold selection is a difficult problem to solve. In large databases, because building a utility list requires a lot of time and memory, the algorithm in large databases still consumes a lot of time and needs more novel ways and frameworks to solve it, which is also the direction of our future research.

Acknowledgment. This research is supported by Shandong Provincial Natural Science Foundation (ZR201911150391).

REFERENCES

[1] W. Gan, L. Chen, S. Wan, J. Chen, and C.-M. Chen, "Anomaly rule detection in sequence data," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[2] T.-Y. Wu, J. C.-W. Lin, U. Yun, C.-H. Chen, G. Srivastava, and X. Lv, "An efficient algorithm for fuzzy frequent itemset mining," *Journal of Intelligent & Fuzzy Systems*, vol. 38, no. 5, pp. 5787–5797, 2020.

[3] T.-Y. Wu, J. C.-W. Lin, Y. Zhang, and C.-H. Chen, "A grid-based swarm intelligence algorithm for privacy-preserving data mining," *Applied Sciences*, vol. 9, no. 4, p. 774, 2019.

[4] R. Agrawal, T. Imieliński, and A. Swami, "Mining association rules between sets of items in large databases," in *Proceedings of the 1993 ACM SIGMOD international conference on Management of data*, 1993, pp. 207–216.

[5] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215. Citeseer, 1994, pp. 487–499.

[6] G. Grahne and J. Zhu, "Efficiently using prefix-trees in mining frequent itemsets." in *FIMI*, vol. 90, 2003, p. 65.

[7] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," *ACM sigmod record*, vol. 29, no. 2, pp. 1–12, 2000.

[8] C. F. Ahmed, S. K. Tanbeer, B.-S. Jeong, and Y.-K. Lee, "Efficient tree structures for high utility pattern mining in incremental databases," *IEEE Transactions on Knowledge and Data Engineering*, vol. 21, no. 12, pp. 1708–1721, 2009.

[9] R. Chan, Q. Yang, and Y.-D. Shen, "Mining high utility itemsets," in *Third IEEE international conference on data mining.* IEEE Computer Society, 2003, pp. 19–19.

[10] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "An effective tree structure for mining high utility itemsets," *Expert Systems with Applications*, vol. 38, no. 6, pp. 7419–7424, 2011.

[11] Y. Liu, W.-k. Liao, and A. Choudhary, "A two-phase algorithm for fast discovery of high utility itemsets," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining.* Springer, 2005, pp. 689–695.

[12] H. Yao, H. J. Hamilton, and L. Geng, "A unified framework for utility-based measures for mining itemsets," in *Proc. of ACM SIGKDD 2nd Workshop on Utility-Based Data Mining.* Citeseer, 2006, pp. 28–37.

[13] L. Chen, W. Gan, Q. Lin, S. Huang, and C.-M. Chen, "Ohuqi: Mining on-shelf high-utility quantitative itemsets," *The Journal of Supercomputing*, pp. 1–25, 2022.

[14] C.-M. Chen, L. Chen, W. Gan, L. Qiu, and W. Ding, "Discovering high utility-occupancy patterns from uncertain data," *Information Sciences*, vol. 546, pp. 1208–1229, 2021.

[15] H. Yao, H. J. Hamilton, and C. J. Butz, "A foundational approach to mining itemset utilities from databases," in *Proceedings of the 2004 SIAM International Conference on Data Mining.* SIAM, 2004, pp. 482–486.

[16] U. Ahmed, J. C.-W. Lin, G. Srivastava, R. Yasin, and Y. Djenouri, "An evolutionary model to mine high expected utility patterns from uncertain databases," *IEEE transactions on emerging topics in computational intelligence*, vol. 5, no. 1, pp. 19–28, 2020.

[17] W. Gan, J. C.-W. Lin, P. Fournier-Viger, H.-C. Chao, V. S. Tseng, and S. Y. Philip, "A survey of utility-oriented pattern mining," *IEEE Transactions on Knowledge and Data Engineering*, vol. 33, no. 4, pp. 1306–1327, 2019.

[18] W. Gan, J. C.-W. Lin, J. Zhang, and P. S. Yu, "Utility mining across multi-sequences with individualized thresholds," *ACM Transactions on Data Science*, vol. 1, no. 2, pp. 1–29, 2020.
[19] G. Srivastava, J. C.-W. Lin, M. Pirouz, Y. Li, and U. Yun, "A pre-large weighted-fusion system of sensed high-utility patterns," *IEEE Sensors Journal*, vol. 21, no. 14, pp. 15 626–15 634, 2020.
[20] J. M.-T. Wu, J. C.-W. Lin, and A. Tamrakar, "High-utility itemset mining with effective pruning strategies," *ACM Transactions on Knowledge Discovery from Data (TKDD)*, vol. 13, no. 6, pp. 1–22, 2019.
[21] P. Fournier-Viger, C.-W. Wu, and V. S. Tseng, "Mining top-k association rules," in *Canadian Conference on Artificial Intelligence*. Springer, 2012, pp. 61–73.
[22] V. S. Tseng, C.-W. Wu, P. Fournier-Viger, and S. Y. Philip, "Efficient algorithms for mining top-k high utility itemsets," *IEEE Transactions on Knowledge and data engineering*, vol. 28, no. 1, pp. 54–67, 2015.
[23] V. Goyal, A. Sureka, and D. Patel, "Efficient skyline itemsets mining," in *Proceedings of the Eighth International C* Conference on Computer Science & Software Engineering*, 2015, pp. 119–124.
[24] V. S. Tseng, C.-W. Wu, B.-E. Shie, and P. S. Yu, "Up-growth: an efficient algorithm for high utility itemset mining," in *Proceedings of the 16th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2010, pp. 253–262.
[25] J.-S. Pan, J. C.-W. Lin, L. Yang, P. Fournier-Viger, and T.-P. Hong, "Efficiently mining of skyline frequent-utility patterns," *Intelligent Data Analysis*, vol. 21, no. 6, pp. 1407–1423, 2017.
[26] P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, and R. Thomas, "A survey of sequential pattern mining," *Data Science and Pattern Recognition*, vol. 1, no. 1, pp. 54–77, 2017.
[27] P. Fournier-Viger, J. C.-W. Lin, B. Vo, T. T. Chi, J. Zhang, and H. B. Le, "A survey of itemset mining," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 4, p. e1207, 2017.
[28] W. Gan, J. C.-W. Lin, H.-C. Chao, and J. Zhan, "Data mining in distributed environment: a survey," *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, vol. 7, no. 6, p. e1216, 2017.
[29] J. Liu, Y. Pan, K. Wang, and J. Han, "Mining frequent item sets by opportunistic projection," in *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002, pp. 229–238.
[30] M. Liu and J. Qu, "Mining high utility itemsets without candidate generation," in *Proceedings of the 21st ACM international conference on Information and knowledge management*, 2012, pp. 55–64.
[31] V. S. Tseng, B.-E. Shie, C.-W. Wu, and S. Y. Philip, "Efficient algorithms for mining high utility itemsets from transactional databases," *IEEE transactions on knowledge and data engineering*, vol. 25, no. 8, pp. 1772–1786, 2012.
[32] J.-S. Yeh, Y.-C. Li, and C.-C. Chang, "Two-phase algorithms for a novel utility-frequent mining model," in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2007, pp. 433–444.
[33] V. Podpecan, N. Lavrac, and I. Kononenko, "A fast algorithm for mining utility-frequent itemsets," *Constraint-Based Mining and Learning*, p. 9, 2007.
[34] H.-T. Kung, F. Luccio, and F. P. Preparata, "On finding the maxima of a set of vectors," *Journal of the ACM (JACM)*, vol. 22, no. 4, pp. 469–476, 1975.
[35] S. Borzsony, D. Kossmann, and K. Stocker, "The skyline operator," in *Proceedings 17th international conference on data engineering*. IEEE, 2001, pp. 421–430.
[36] J. Chomicki, P. Godfrey, J. Gryz, and D. Liang, "Skyline with presorting," in *ICDE*, vol. 3, 2003, pp. 717–719.
[37] K.-L. Tan, P.-K. Eng, B. C. Ooi *et al.*, "Efficient progressive skyline computation," in *VLDB*, vol. 1, 2001, pp. 301–310.
[38] D. Papadias, Y. Tao, G. Fu, and B. Seeger, "Progressive skyline computation in database systems," *ACM Transactions on Database Systems (TODS)*, vol. 30, no. 1, pp. 41–82, 2005.
[39] J. C.-W. Lin, L. Yang, P. Fournier-Viger, and T.-P. Hong, "Mining of skyline patterns by considering both frequent and utility constraints," *Engineering Applications of Artificial Intelligence*, vol. 77, pp. 229–238, 2019.
[40] J. C.-W. Lin, L. Yang, P. Fournier-Viger, S. Dawar, V. Goyal, A. Sureka, and B. Vo, "A more efficient algorithm to mine skyline frequent-utility patterns," in *International conference on genetic and evolutionary computing*. Springer, 2016, pp. 127–135.

[41] W. Song, C. Zheng, and P. Fournier-Viger, "Mining skyline frequent-utility itemsets with utility filtering," in *Pacific Rim International Conference on Artificial Intelligence*. Springer, 2021, pp. 411–424.

[42] Fournier-Viger, "Spmf: A java open-source data mining library," ,https://www.philippe-fournier-viger.com/spmf, 2016.