Journal of Network Intelligence
Taiwan Ubiquitous Information
 ISSN 2414-8105 (Online)
Volume **8**, Number **1**, February **2023**

Skyline Frequent-utility Patterns Mining: A Survey

Jimmy Ming-Tai Wu*

College of Computer Science and Engineering
Shandong University of Science and Technology
Qindao, China
wmt@wmt35.idv.tw

Huiying Zhou

College of Computer Science and Engineering
Shandong University of Science and Technology
Qindao, China
2254625719@qq.com

Matin Pirouz

Department of Computer Science
California State University Fresno
California, USA
mpirouz@csufresno.edu

Shahab Tayeb

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

*Corresponding author: Jimmy Ming-Tai Wu

Received November 12, 2022, revised December 20, 2022, accepted January 17, 2023.

Abstract. *High Utility Itemset Mining (HUIM) and Frequent Itemset Mining (FIM) are two important branches in the data mining area, where Frequent Itemset Mining considers itemsets that occur in large numbers in the transaction database, while High Utility Itemset Mining considers the number and unit utility value of itemsets. However, in practical applications, both mining approaches have their own limitations, for example, itemsets with a high number of occurrences may contribute less to the ultimate goal of the merchant, and although these itemsets are strongly related to each other, they do not have great reference value for the final decision. The itemsets with high utility values, on the other hand, may have a very low number of occurrences and weak connections between the itemsets. Therefore recommending these itemsets to users may not be very meaningful. In order to better meet the requirements of real life, some scholars took these two mining algorithms into account, hoping to find those high-quality itemsets, that is, those itemsets with more occurrences and high utility values. For this purpose, the skyline frequent-utility pattern mining (SFUPM) algorithm is introduced. This paper describes in detail the classification of skyline frequent-high uyility itemset mining, mining methods and future research directions.*

Keywords: Data mining; Frequent Itemset Mining; High Utility Itemset Mining; Skyline Frequent Utility Itemset Mining

1. **Introduction.** Data mining is the process of extracting (or "mining") useful intelligence through the analysis of large amounts of information and data sets, which can predict trends, mitigate risks, and thereby help decision makers solve problems and find new recommendations or decisions. Data mining is like mining in real life. Workers look for valuable ores such as gold or coal in a large number of ores, and diggers analyze the piles of materials and look for valuable resources and elements. Data mining has become more and more widely used in the commercial field, and it has become a new economic asset, bringing new entrepreneurial directions, business models and investment opportunities to the whole society.

Data mining is not only like a compass on the sea, which can point out the direction, but also like the eyes and brain, which can gain insights through analyzing data. In today's era of big data, organizations or enterprises need to rely more on data analysis rather than experience and intuition to make decisions. They fully tap and use the value of data to bring strong competitiveness to organizations or enterprises. In addition, in accordance with the requirements of the national development strategy, the shortage of post talents and the drive of the market size all reflect the importance of the data analyst profession from different aspects. In recent years, with the continuous advancement of modern information technology, various technological applications based on big data have become hot spots in the market. By applying big data to product marketing, customer experience improvement, risk control, etc., good results have been achieved. Therefore, data mining will be applied to more and more industries in the future.

Frequent itemset mining (FIM) [1, 2] belongs to a subfield of data mining, and its main task is to find out the transactions that often appear together in the databases. FIM was first widely used in market basket analysis, and then widely used in product recommendation, text mining and web clickstream analysis. However, FIM has many disadvantages, for example, in the business field it only considers the frequency of goods sold, and does not consider the price of goods or unit profit and other indicators. For example, the profit of selling a diamond in a store is much higher than the profit of selling bread for a day. Bread belongs to the frequent itemset, but it may not promote the economic benefits of business, instead, the infrequent itemset of diamonds can bring profits to the store. In order to address these FIM deficiencies, proposes high utility itemset mining (HUIM) [3, 4, 5, 6].

A fundamental job of frequent itemset mining is high utility itemset mining. It simultaneously considers the profit per unit of the itemset and the number of itemset in the database [7, 8], and can be used to measure the "usefulness" of an itemset, thus generating real business benefits. However, since FIM and HUIM only consider one measurement factor, in the practical application, two or more factors should be considered to obtain more valuable information and facilitate decision-making. For example, when buying a house, you need to consider such factors as the location of the community, the price of the house, and the convenient transportation nearby. In order to maximize the potential value of rich data [9], Yeh et al. [10, 11] for the first time combined itemset frequent and itemset utility together and proposed a frequent high utility itemset mining algorithm because of Two-phase algorithm. Then, an algorithm for quickly mining frequent high utility itemsets was proposed by Podpecan et al. [12]. These two algorithms are because of a minimum threshold of utility and a minimum threshold of support. And in the process of algorithm mining, all itemsets that are greater than the minimum utility threshold and the minimum support threshold can be mined. However, setting the right threshold in order to obtain the desired set of items is not a simple task. In order to overcome this limitation, Goyal et al. [13] came up with an effective SKYMINE algorithm to receive skyline frequent-utility patterns (SFUPs) [13, 14, 15]. The SKYMINE algorithm saves

relevant information by using the UP-tree structure [16], and use a similar method to UP-Growth structure [16] to obtain potential skyline frequent-utility patterns (PSFUPs), and then mines real SFUPs from PSFUPs. The advantage of this approach is that you do not have to set a minimum threshold of utility and a minimum threshold of support, and it can correctly and completely mine all skyline itemsets, namely SFUPs, considering the two factors of the itemset utility and itemset frequent. In addition, skyline frequent-utility itemset have many research directions in the future, for example, combining it with vehicle cloud computing [17], graph clustering [18], privacy-preserving data mining [19], and fuzzy frequent itemset mining [20] to better obtain useful information information.

We detail the key techniques of the SFUPM algorithm and provide a comprehensive review of it, which can serve as the latest advances in the data mining area. The classification of the SFUPM algorithm is shown in Figure 1. The main points of this paper are shown below.

1. A classification of frequent high utility mining techniques is presented, including relevant algorithms for mining SFUPs in stand-alone and big data environments.
2. We subdivide SFUPM algorithms based on stand-alone environment into UP-tree based, utility list based and extended utility list based algorithms. We also subdivide the methods based on big data environment into Hadoop-based environment and Spark-based environment.
3. This paper briefly describes the pruning strategy of the SFUPM algorithm.
4. This paper provides some classification and discussion of existing SFUPM algorithms. Finally, the future research direction of the SFUPM algorithm is briefly imagined.

The remainder of this paper is structured as follows: In section 2, frequent itemset mining and high utility itemset mining are briefly described to illustrate how itemset mining develops from frequent pattern to high utility pattern, and then to frequent high utility pattern mining. In Section 3, we classify and describe the SFUPM algorithms based on a single-computer environment, as well as their advantages and disadvantages. In Section 4, we classify and describe the SFUPM algorithm based on the big data environment. Section 5 summarizes the pruning strategy of the SFUPM algorithm. Section 6 highlights the future research directions of SFUPM algorithms. Finally, Section 7 draws conclusions.

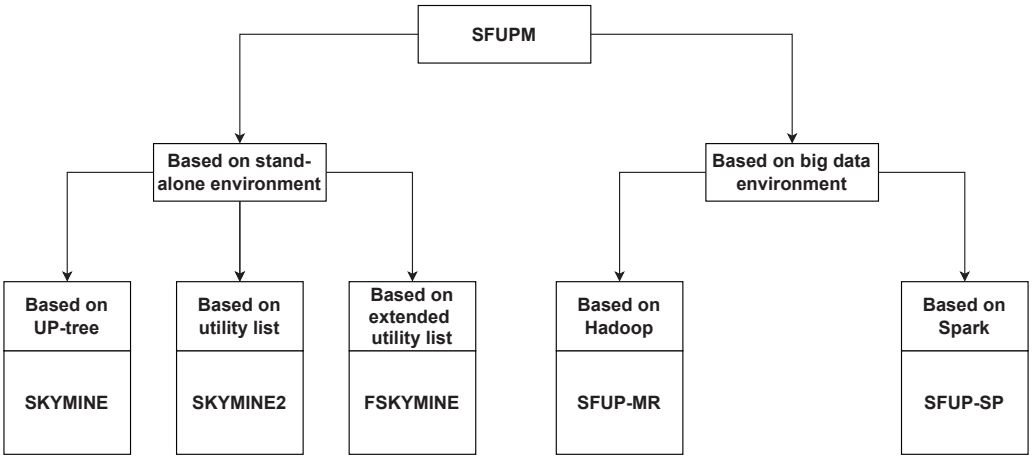

FIGURE 1. Frequent-high utility itemset mining classification methods.

TABLE 1. A transaction database.

TID	Transaction
T_1	(a,3),(d,2),(e,1)
T_2	(a,2),(b,2),(c,3),(d,5),(e,1)
T_3	(a,4),(c,2),(e,3)
T_4	(a,3),(b,3),(e,4)
T_5	(a,2),(b,4),(c,4),(d,1)
T_6	(a,2),(c,1),(d,2),(e,3)
T_7	(a,4),(b,2)

2. **Preliminaries and Definitions.** In this section, we would like to briefly outline the concepts and definitions related to frequent itemset mining and high utility itemset mining.

2.1. **Frequent Itemset Mining (FIM).** The key to the association rule mining [21, 22] problem is the frequent itemset mining algorithm, whose main purpose is to discover associations between itemsets in a transactional database. Suppose that $I = \{i_1, i_2, \ldots, i_m\}$ is a collection of different letters, in which the elements are called items or commodities, and I is a set of all commodities. If there is a set X and $X \subseteq I$, X is called $k-itemset$, where k is the length of the itemset. $D = \{T_1, T_2, \ldots, T_n\}$ is a transaction database containing n transaction data, where transaction T is a collection of items, and $T \subseteq I$. Each transaction is associated with a unique identity, which is recorded as a TID. In the example in Table 1 there are 7 transactions T_q and 5 different items i_j. Let X represent an itemset in I. If $X \subseteq T$, then itemset X appears in transaction T. We call X a frequent itemset if the support of X is greater than or equal to some given minimum support threshold (minsup). The following definition is given for support.

Definition 2.1. *In transaction database D, the support of itemset X is noted as $sup(X)$, and it is described as:*

$$sup(X) = |\{T \in D \mid X \subseteq T\}|/|D| \tag{1}$$

Example 2.1. *When minsup=0.5 in Table 1, Table 2 displays the transaction database's frequent pattern.*

TABLE 2. All frequent itemsets when minsup=0.5.

$1-itemset$	sup	$2-itemset$	sup
a	1	ab	0.57
b	0.57	ac	0.57
c	0.57	ad	0.57
d	0.57	ae	0.71
e	0.71		

During the frequent itemset mining process, a large number of intermediate items are generated, and An Apriori algorithm that employs the qualities of the Apriori method (shown in proposition 1) as a pruning strategy to condense the search space was developed by Agrawal et al [21, 22].

Proposition 2.1. *If the itemset X is a frequent itemset, then all its non-empty subsets are frequent itemsets. This is due to the inverse monotonicity of the support, which is upper bounded in nature. Similarly, if the itemset is an infrequent itemset, then all of its supersets are infrequent itemsets(each itemset's support cannot be higher than any subset's).*

However, the Apriori algorithm requires several scans of the database and may generate a huge candidate set. So as to improve the efficiency of mining, FP-Growth [23], an algorithm for discovering frequent patterns based on Frequent Pattern Tree (FP-Tree), is proposed, which constructs an FP-Tree, maps the data in the dataset to the tree, and finds all frequent itemsets based on this FP-Tree. Mining execution efficiency has greatly improved.

However, frequent itemset mining only reveals the number of occurrences of itemsets, but disregards several other crucial elements, for instance, the interest level, unit utility, relevance and weight of itemsets. So as to address this issue and mine more effective information, high utility itemset mining has been proposed, it combines the number of items in the transaction database with the profit per unit.

2.2. **High Utility Itemset Mining(HUIM).** High utility itemset mining [24, 25, 26, 16, 27] is a research direction of pattern mining in the sub domain of data mining, which includes extracting very important patterns from transaction database. The utility of a model can have different meanings in various problems, for instance, its benefits, customer satisfaction, value of interest or risk, and so [28, 29, 30, 31, 32, 33]. It extends the problem of FIM by taking into account both of the quantity of items and incomes. The revenue table ptable $= \{pr_1, pr_2, \ldots, pr_m\}$ corresponding to the transaction in the transaction database D represents the profit (external utility) generated by each item i_j, and each transaction item is linked to an internal utility (Quantity), denoted as $q\left(i_j, T_q\right)$. The external utility of the item in Table 1 is displayed in Table 3.

Table 3. External utility of items.

Item	*a*	*b*	*c*	*d*	*e*
Utility	3	5	4	1	2

Definition 2.2. *In transaction T_q, the utility of item i_j is referred to as $u\left(i_j, T_q\right)$, and it is described as:*

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

Example 2.2. *In the transaction T_2, the utility of $\{b\}$ is $u(b, T_2) = q(b, T_2) \times pr(b) = 2 \times 5 = 10$.*

Definition 2.3. *In transaction T_q, the utility of the itemset X is referred to as $u\left(X, T_q\right)$, and it is described as:*

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

Example 2.3. *In the transaction T_2, the utility of $\{bcd\}$ is $u(bcd, T_2) = u(b, T_2) + u(c, T_2) + u(d, T_2) = 10 + 12 + 5 = 27$.*

Definition 2.4. *In transaction database D, the utility of itemset X is referred to as u(X), and it is described as:*

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

Example 2.4. *For instance, Table 1 shows that the utility of* $\{a\}$ *in Database D. It is* $u(a) = u(a, T_1) + u(a, T_2) + u(a, T_3) + u(a, T_4) + u(a, T_5) + u(a, T_6) + u(a, T_7) = 9 + 6 + 12 + 9 + 6 + 6 + 12 = 60$. $u(bcd) = u(bcd, T_2) + u(bcd, T_5) = 27 + 37 = 64$.

Definition 2.5. *If* $u(X) \geq minutil$*, the itemset X is referred to as a high-utility itemset, and minutil denotes the minimum utility threshold.*

Example 2.5. *In the example in Table 1, the high-utility itemsets when minutil=50 are displayed in Table 4.*

Table 4. All high-utility itemsets when minutil=50.

$1-itemset$	$utility$	$2-itemset$	$utility$	$3-itemset$	$utility$	$4-itemset$	$utility$
a	60	ab	88	abc	70	abcd	76
b	55	ac	70	abe	50		
		ae	66	ace	62		
		bc	58	bcd	64		

Definition 2.6. *The transaction utility in the transaction database D is referred to as* $tu(T_q)$*, and it is described as:*

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

Example 2.6. *In Table 1,* $tu(T_1) = u(a, T_1) + u(d, T_1) + u(e, T_1) = 9 + 2 + 2 = 13$. *Calculating* T_2 *to* T_7 *in the same way results in* $tu(T_2) = 35, tu(T_3) = 26, tu(T_4) = 32, tu(T_5) = 43, tu(T_6) = 18, tu(T_7) = 22$.

Definition 2.7. *In transaction database D, the transaction weighted utility of itemset X is referred to as twu (X), and it is described as:*

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

Example 2.7. *For instance, each item's twu in Table 1 is* $twu(a) = T_1 + T_2 + T_3 + T_4 + T_5 + T_6 + T_7 = 189, twu(b) = 132, twu(c) = 122, twu(d) = 109, twu(e) = 124$.

Definition 2.8. *If* $twu(X) \geq minutil$*, minutil is the minimal standard of utility, the itemset X is known as the High Transaction Weighted Utility Itemset (HTWUI). For instance, the itemset (ab) is HTWUI for minutil = 88.*

Proposition 2.2. *Assume that Y is a (k-1)-itemset and that X is a k-itemset, where* $Y \subset X$*. If X is HTWUI, then Y is as well. All of an itemset's supersets will thus be LTWUI if it is a low transaction weighted utility itemset (LTWUI). As a result, during high utility itemset mining, this feature may be utilized to restrict the search area by eliminating LTWUI and its supersets from the search space.*

However, both the frequent itemset mining algorithm and the high utility itemset mining algorithm only consider one factor, but in practical applications, two or more factors need to be considered at the same time to help decision-making. As a consequence, a mining algorithm for high utility-frequent itemset is presented.

2.3. **Frequent Utility Itemset Mining(FUIM).** The frequent utility itemset mining algorithm is a comprehensive consideration of the frequent itemset mining algorithm and the high utility itemset mining algorithm. It mines SFUPs by taking both frequent and utility into account, and finally mines skyline itemset. Skyline itemset belongs to the classical algorithm of multi-factor optimization issues. Many classical multi-factor issues transform multi-factor optimization issues into single factor optimization issues by weight vectors and other factors, but the solution of this algorithm depends on weight vectors. Skyline considers multiple targets at the same time, and finally returns a series of parallel, uncontrollable solutions. "Control" means that an itemset is no worse than another itemset in each dimension, and the itemset is better than another itemset in at least one dimension.

Definition 2.9. *For itemset X and Y, if* $f(X) > f(Y)$ *and* $u(X) \geq u(Y)$ *or* $f(X) \geq f(Y)$ *and* $u(X) > u(Y)$*, then think that itemset X dominates itemset Y, and recorded as* $X \succ Y$*.*

Example 2.8. *In the example of this paper, the set of terms* $\{a\} \succ \{de\}$*, because* $f(a) > f(de)$ *and* $u(a) > u(de)$*.*

Definition 2.10. *When frequent and utility are considered together, if an itemset X is not dominated by any other itemsets in database D, it is said to be a skyline frequent utility pattern (SFUP).*

3. **SFUPM Algorithm Based on Single-Computer Environment.** In this section, we summarize the SFUPM algorithm in a stand-alone environment, which is based on UP-tree, utility list and extended utility list respectively.

3.1. **Mining Algorithm Based on UP-tree.** The UP-tree based mining algorithm is known as the SKYLINE algorithm [13], Filtering and refining are the two steps of the algorithm. A group of potential itemsets are mined in the filtering step, and then their final validation is performed in the refinement phase. The following subsections provide a brief description of the UP-tree.

3.1.1. *UP-tree.* A compact tree structure made up of header tables is known as a UP-tree. A list of items for each node in the UP-tree is shown in the title table. The title table includes an item list of all nodes in the UP-tree, it contains the itemset, the estimated utility value of the itemset, and links to nodes in the UP-tree of the itemset. The link node is the starting node containing the same itemset node link, which is used to traverse the nodes of the same itemset in the UP-tree. Each node n in the UP-tree (except the special node root node that does not contain any itemset) has items (n.name), a parent node (n.parent), node utility (n.utility), support count (n.count), the same item node link (n.hlink) information. The n.utility represents the estimated utility of the itemset on the path from this node to the root node. Likewise, for the group of items on the route from this node to the root node, the value of n.count shows how many support count there are. The UP-tree in the example in this paper is shown in Figure 2.

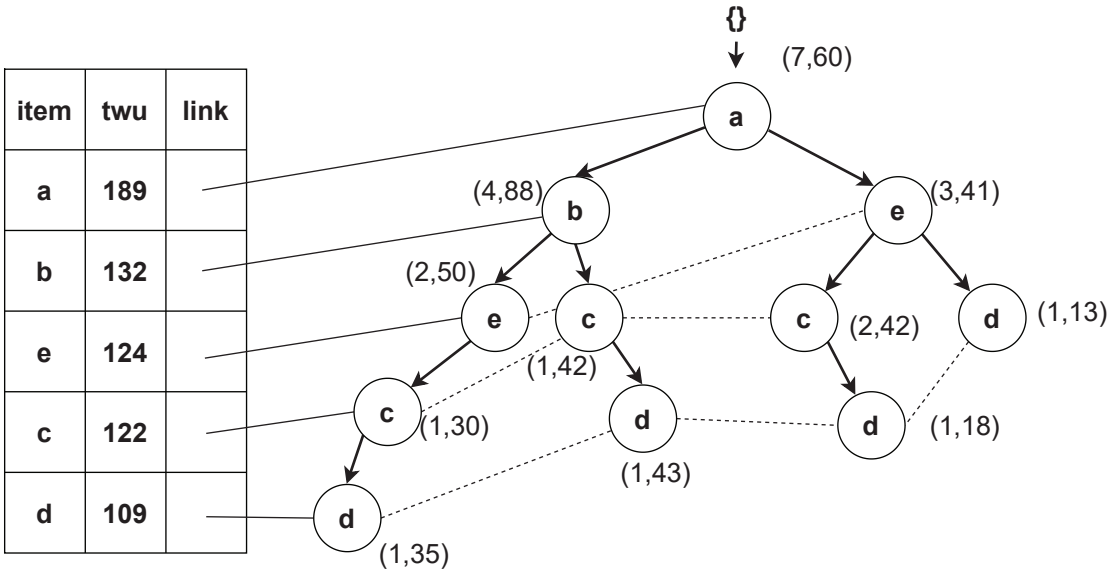

FIGURE 2. UP-tree represented by the example.

3.1.2. *SKYMINE Algorithm Process.* The first stage of SKYMINE algorithm follows procedures for mining PFU itemset in general. First, the PFU itemsets are sorted by support count and stored in a list L, and set the utility threshold at all indices m in the list to 0, that is $umin(m) = 0 \quad (0 < m < |D|)$. Use a good search strategy to search for PFU patterns from database D. When a specific counting supported PFU mode (such as n) is found during the search, it will be inserted at index n to the list L. We are linked the utility values of the lower limit (lb) and upper limit (ub) with the PFU itemset. After add to the PFU itemset of index n, the update method of list L are as follows:

(1) Check the $umin(m)$ value at index n and the $umin(m)$ value at index $m \quad (0 \leq m \leq n < |D|)$. If $umin(m)$ is less than the lb utility value of the set of PFU items being added, update it.

(2) If the PFU mode with index n or less is to be dominated by the added PFU itemset, delete it. If another PFU itemset with higher lb utility value exists and higher support value in the list, the PFU item group is said to be dominant. The algorithm continues to search for more PFU patterns until the search strategy cannot find any patterns.

However, the SKYMINE algorithm will need to generate a significant number of candidate itemsets, so the entire SFUP mining requires a long execution time. Therefore, an algorithm based on the utility list structure was proposed by Jeng Shyamg Pana et al. [34].

3.2. **Mining Algorithm Based on Utility List.** The SKYMINE2 algorithm makes use of the utility list structure [34, 15, 35] for data mining. In contrast to the SKYMINE algorithm, the utility list structure in the algorithm can be used to efficiently calculate the actual utility of itemsets without generating candidates. In addition, a maximum utility (umax) array for storing the utility-related information of each item set with the largest frequency is also proposed, which can significantly reduce the amount of itemsets considered when mining SFUP directly and can be based on both frequency and utility. Efficiently finds non-dominated itemsets.

3.2.1. *Utility List Structure.* We sorted the items in the database in ascending order by *twu* [4]. In database D, the utility list structure of itemset X is represented as a group

of tuples, which contains three fields: $tid, iutil$ and $rutil$. These three fields respectively represent the transaction ID that contains the itemset X, X's actual utility in tid, for instance $u(X, T_q)$, the total of utility of all itemsets after X in tid, for instance $\sum_{i_j \in T_q/X} u(i_j, T_q)$. In the example of this article, the order of all $1-itemsets$ sorted in ascending order by twu is d, c, e, b, a. Their utility list structures are shown in Figure 3.

d		
tid	iutil	rutil
1	2	11
2	5	30
5	1	42
6	2	16

c		
tid	iutil	rutil
2	12	18
3	8	18
5	16	26
6	4	12

e		
tid	iutil	rutil
1	2	9
2	2	16
3	6	12
4	8	24
6	6	6

b		
tid	iutil	rutil
2	10	6
4	15	9
5	20	6
7	10	12

a		
tid	iutil	rutil
1	9	0
2	6	0
3	12	0
4	9	0
5	6	0
6	6	0
7	12	0

FIGURE 3. The constructed utility list.

In the algorithm, in addition to maintaining the maximum utility corresponding to the frequent value, a maximum utility (umax) array is also set to maintain the maximum utility of the frequent value.

Definition 3.1. *The array holds the maximum utility for frequency value i and it is described as umax(i).*

Definition 3.2. *If the itemset X's frequency is i and the utility of the non-itemset is more than u(X), it is termed a potential SFUP (PSFUP).*

However, because the algorithm generates a large number of utility lists, connection operations and potential SFUI, the SKYMINE2 algorithm still has a high computational cost. Therefore, an algorithm built the extended utility list data structure was proposed by Hung Manh Nguyen et al.[36].

3.3. **Mining Algorithm Based on Extended Utility List.** The FSKYMINE algorithm based on the extended utility list can effectively reduce the number of utility lists, join operations and potential SFUIs in the process of mining skyline patterns.

Definition 3.3. *The itemset i extension itemset S is expressed as Si and* $S\_i = S \cup i$, *for instance, if* $S = \{b, d, g\}$ *and* $i = \{e\}$, *then* $S\_i = \{b, d, e, g\}$.

Definition 3.4. *After the transaction database is sorted, the extended utility list of the itemset S_i in it is represented by a group of tuples, and the tuples are composed of four fields: tid, itemsetutil, itemutil and rutil. Where tid represents the ID of the transaction containing itemset S_i, itemsetutil represents the utility of S in tid, itemsutil represents the utility of i, and rutil represents the total of the utilities of itemsets remaining after i in transaction tid.*

Definition 3.5. *In the extended utility list of itemset S_i, S_i.sumtemutils represents the total of itemsettil in the utility list of itemset S_i, S_i.sumitemsetutils represents the total of itemsetutil in the scope utility list of itemset S_i, S_i.sumrutils represents the itemset S_i the total of the rutil extended utility list.*

Proposition 3.1. *When the itemset S_i's frequency is i, if the sum of sumitemutils value and sumitemsetutils value in the extended utility list of itemset S_x is not less than umax(i), then S_i is considered to be a PSFUI.*

Proposition 3.2. *When the itemset S_i's frequency is i, if the total of sumrutils value and sumitemsetutils value and sumitemutils value in the extended utility list of itemset S_i is smaller than umax(i), so all extensions of S_i are not considered to be SFUI.*

Definition 3.6. *When itemset m and itemset n co-occur in transaction T in the sorted transaction database, and m is ordered before n, denote the total of the utility of item m's remaining items in T as rtwuc(m, n, T):*

$$\mathrm{rtwuc}(m, n, T) = \sum_{i \in T \wedge i \text{ from m in total order}} u(i, T) \tag{7}$$

Definition 3.7. *When the item pair m, n appear in the database at the same time, the total of the remaining transaction-weighted utilities of the item pair m, n in all transactions where the item pair m, n appears is defined as:*

$$\mathrm{rtwuc}(m, n) == \sum_{(T \in D) \wedge ((m,n) \subseteq T)} \mathrm{rtwuc}(m, n, T) \tag{8}$$

First, calculate the *twu* for all $1 - itemsets$. Second, Sort $1 - itemsets$ in *twu* ascending order. Then, calculate the remaining transaction-weighted utility ($rtwuc(m, n)$) of itemsets in the database for x, y, and formation the extended utility list of $1 - itemsets$.

4. **SFUPM Algorithm Based on Big Data Environment.** In this section, we summarize the SFUPM algorithm in the big data environment, which is based on Hadoop platform and Spark platform respectively. In this section, we summarize the SFUPM algorithm in the big data environment, which is based on Hadoop platform and Spark platform respectively.

4.1. **Mining Algorithm Based on Hadoop Platform.** SFUP-MR algorithm is a new parallel algorithm based on hadoop platform, namely, mapreduce based three-stage iterative algorithm. MapReduce [37] is used to divide the mining task of an entire large dataset into multiple independent subtasks to find frequent-high utility patterns in parallel. Hadoop is a distributed architecture, which makes full use of the performance of clusters for computation and hadoop is a distributed architecture that leverages the performance of clusters for computation and storage. The key components are mapReduce and hadoop distributed file system (hdfs). Thanks to hdfs, data localization operations on storage nodes can be accomplished by storing data on the nodes when computing on hadoop. However, in the previous high-performance computing cluster, computing nodes need to access the shared file system composed of storage area networks (SANs) to obtain data. This left some compute nodes in the cluster to wait for data when there was not enough data, wasting resources.

MapReduce is a sufficiently abstract and general programming model. Its main principle is divide and conquer. More importantly, it can recursively merge and sort data. In addition, it simplifies parallel programming tasks by hiding the underlying details of parallelization from programmers. The processing of any mapReduce algorithm consists of map and reduce two stages. Mapper and reducer are the two objects that handle the map phase and reduce phase, respectively. The input data is separated into key-value

pairs and sent to several mappers. The key and value pairs generated by the mapper are shuffled and sorted in the middle and transferred to the reducer. The reducer merges keys and values with the same as key to get a merged key-value pair. Then output middle or the final results to hdfs.

4.2. **Mining Algorithm Based on Spark Platform.** The SFUP-SP algorithm is a parallel-style algorithm built on the Apache Spark platform [38]. The purpose of this algorithm is to achieve the mining of skyline patterns through the use of two major classes of operators, transformations and actions [39]. Meanwhile, the application of resilient distributed dataset (RDD) [40, 39] in cloud computing provides a good environment for data analysis of big data. Spark framework is divided into a driver side and an executor side [41, 39, 42], where the driver side inputs data, generates DAG [39, 43], manages resources and is responsible for scheduling; the executor side executes transfomation and action arithmetic to perform multiple tasks concurrently. Spark is based on the idea of mapreduce is extended, it not only retains the advantages of mapreduce distributed parallel computing and also fill the obvious deficiencies of mapreduce, and mapreduce will be saved in the intermediate results of the nature of the disk is different to Spark will be the Job intermediate output and results stored in memory, so it does not need to Therefore, it does not need to read and write hdfs, so Spark can be better suited to data mining and machine learning and other algorithms that require iterative mapreduce.

Spark is a memory-based computing framework, which allows intermediate data to be stored in memory to improve the running speed, and provides rich APIs for operating data to improve the development speed. Spark constructs each task processed as a DAG (Directed Acyclic Graph) for execution. The implementation principle is to iteratively calculate data in memory based on RDD (Resilient Distributed Dataset, Elastic Distributed Dataset). , to achieve high-performance fast computing processing of batch and streaming data. Spark can be easily integrated with other open source products. For example: spark can use hadoop's yarn and Apache Mesos as its resource management and scheduler; it can process all data supported by hadoop, including hdfs, hbase and cassandra, etc.

5. **Pruning Strategy.** In this section, we briefly introduce several pruning strategies [44, 45, 15] commonly used in the skyline frequent-utility patterns mining algorithm. They are applicable to stand-alone environment and big data environment respectively.

5.1. **Iutil and Rutil Pruning.** In the process of mining SFUPs, the SKYMINE algorithm generates a large set of candidate items and the algorithm overestimates the upper limit of the itemset. To address this issue, a strategy was defined in order to accelerate the mining of SFUPs.

Definition 5.1. *If the frequency of the pattern is not less than index i, where (*$1 \leq i \leq |D|$*, then the maximum utility of the pattern can be maintained in the utility-maximum (utilmax) structure, defined as follows:*

$$utilmax(i) = max\{u(X) \mid f(X) \geq i\} \tag{9}$$

Allocate a fixed address space to the utilmax array, whose size is the maximum frequency of the itemsets in database D add 1. Based on this structure, in the whole mining process of the algorithm, the maximum utility of the itemset under the constraint of the frequency condition can be maintained, and the search space in the mining process of SFUP can be reduced.

Lemma 5.1. *Let A be the set of itemset, and the total of its iutil in its UL structure is smaller than utilmax(f(A)), A cannot be SFUPs.*

Proof: When the itemset B has $\sum_{A\subseteq T_q \wedge T_q \in D} \text{util}\,(A, T_q) < \text{utilmax}(f(A))$ conditions, it can be concluded that $u(B) < u(A)$, $f(B) \geq f(A)$. So A is controlled by B, A cannot be SFUP.

Lemma 5.2. *Let A be the set of itemset, then add itemset B to itemset A, expand it and define it as (A∪B), denoted by A▷B. If the total of the iutil and rutil values corresponding to A is less than utilmax(f(A)), then all extensions of A cannot be SFUP.*

Proof: Extend the set of item A to $A^{'}$. $\forall A' \subseteq \text{t} \Rightarrow (A' - A) = (A'/A)$ because $A \subset A' \subseteq t \Rightarrow (A'/A) \subseteq (t/A)$

$$\begin{aligned} therefore \quad f\,(A') \leq f(A),\ \text{and}\ u\,(A', t) &= u(A, t) + u\,(A' - A, t) \\ &= u(A, t) + u\,(A'/A, t) \\ &= u(A, t) + \sum_{i_j \in A'/A} u\,(i_j, t) \\ &\leq u(A, t) + \sum_{i_j \in (t/A)} u\,(i_j, t) \\ &= u(A, t) + ru(A, t) \end{aligned} \tag{10}$$

Let $tid(t)$ represents the ID of transaction t in database D, the tid tuple of itemset A is represented by $A.tids$, and the tid tuple of itemset $A^{'}$ is represented by $A^{'}.tids$, so:

$$\begin{aligned} &because \quad A \subset A' \Rightarrow A'.tids \subseteq A.tids. \\ &therefore \quad u\,(A') = \sum_{id(t) \in A'.tids} u\,(A', t) \leq \sum_{id(t) \in A'.tids} u(A, t) + \text{rutil}(A, t) \\ &\qquad \leq \sum_{id(t) \in A.tids} u(A, t) + \text{rutil}(A, t) \\ &\qquad < utilmax(f(A)) \end{aligned} \tag{11}$$

Because $f\,(A') \leq f(A)$, therefore $\exists\ u(B) = utilmax(f(A)) \geq u\,(A')$ and $f(B) \geq f(A) \geq f\,(A') \Rightarrow A'$ cannot be SFUPs.

Therefore, the pruning strategy that follows may be derived from the aforementioned two lemmas:

All extensions of itemset A cannot be SFUPs if the total of *iutil* and *rutil* in the UL structure of itemset A is smaller than $utilmax(f(A))$.

5.2. **Sub-tree Utility and Local Utility Pruning.** In this section, we briefly introduce how to use the utilmax array, the upper bound value of subtree utility and the upper bound value of local utility to form a new pruning strategy suitable for big data, so as to reduce the search range in the algorithm mining process the goal of.

Definition 5.2. *Suppose the items is $I = \{i_1, i_2, \ldots, i_k\}$, after sorting I by twu in descending order, the last item of the set X of items is i_n. The items $I' = \{i_{n+1}, i_{n+2}, \ldots, i_k\}$ is a sub-list of I, and an item $y = i_m \in I'$, $g(X \cup \{y\})$ is a group of transactions, where $X \cup \{y\}$ is an item in the transaction. The sub-tree utility of y with respect to X is called as $su(X, y)$, and it is described as follows:*

$$su(X,y) = \sum_{T \in g(X \cup \{y\})} \left[\sum_{i \in T \cap \{i_{n+1}, i_{n+2}, \ldots, i_k\}} u(i,T) + u(X,T) + u(y,T) \right] \tag{12}$$

Definition 5.3. *Suppose the items is $I = \{i_1, i_2, \ldots, i_k\}$, after sorting I by twu in descending order, the last item of the set X of items is i_n. The items $I' = \{i_{n+1}, i_{n+2}, \ldots, i_k\}$ is a sub-list of I, and an item $y = i_m \in I'$, $g(X \cup \{y\})$ is a group of transactions, where $X \cup \{y\}$ is an item in the transaction. The local utility of y with respect to X is called as $lu(X,y)$, and it is described as follows:*

$$lu(X,y) = \sum_{T \in g(X \cup \{y\})} \left[\sum_{i \in T \cap I'} u(i,T) + u(X,T) \right] \tag{13}$$

6. **Future Direction of Frequent Utility Itemset Mining Algorithm.** In recent years, many research progresses have been made in mining frequent utility itemsets. However, there are more emerging issues to consider. Therefore, we will briefly discuss the future directions of frequent utility itemset mining algorithms in this section.

6.1. **Mining Algorithm for Frequent Utility Itemset with Negative Utility.** All algorithms in the previous studies considered only positive utilities. However, in real life, transactional datasets are of negative utility [46]. In business competition, many companies use marketing schemes to sell a product to consumers at zero or negative profit, and then increase their business revenue by guiding consumers to buy other goods with high profit through certain conditions. For example, in real life, we often encounter supermarkets engaging in promotional activities in order to increase sales, such as buying instant noodles and giving away bowls, and the bowls are sent at a negative profit loss. It is much more than the loss caused by the bowl, which still makes the seller profitable. If a mining algorithm that does not consider negative utility itemsets is used, the high utility itemsets will be estimated as low utility itemsets when mining is performed, resulting in a significant number of high utility itemsets not being mined, which leads to mining of frequent-high utility itemset is incomplete. Therefore, mining frequent utility itemsets with negative utility values would be a good research direction.

6.2. **Incremental Mining Algorithm for Frequent Utility Itemset.** The databases in previous studies are static databases, which are usually analyzed statically and then the mining process is executed in a batch manner. However, in practical applications, the transactions in the database are usually in a state of change. Since customer purchases are always dynamic, the original database will receive some new transaction records. Then the previous frequent-high utility itemset may become no longer valid, or there may be some new hidden information in the updated database. In order to better deal with this problem, we can propose an efficient incremental mining algorithm [47] to mine frequent-high utility itemsets existing in the dynamic database. In some cases, the algorithm can avert or decrease the need to scan the original database again, thereby saving the computation time of incremental mining. In order to mine the frequent and efficient itemsets in incremental databases more efficiently, we need to drive the research as deeply as possible.

6.3. **Mining Algorithms for Frequent and High Utility of Quantitative Itemsets.** In the previous study, the itemset only considered frequent, but not quantity as a key factor. However, in real life, quantity is always involved as an influencing factor. High-Utility Quantitative Itemset Mining (HUQIM) [48] is a research branch that has only started in the past two years, and it belongs to a research branch of utility. It also takes into account the utility factor while also adding a consideration of the quantity within the scope of the program. Therefore, in order to better mine the information we need, frequent and high utility mining algorithms with quantitative itemsets is a research direction of practical importance.

7. **Conclusions.** In recent years, a lot of research has been done on the mining of frequent utility patterns. The purpose of these studies is to find patterns that appear frequently and have high utility from large amounts of data. With the in-depth research on frequent utility pattern mining, many good data structures and algorithms have emerged. So in order to better understand the algorithms of these studies, we conducted a survey on skyline pattern mining by reading a large number of literatures, and formed this article. This paper presents a series of summaries and methodological discussions on the problem of mining frequent high-utility patterns. First, we conduct some investigations on the algorithm from a stand-alone environment and a big data environment. The classification of the SFUP algorithm in the two environments is introduced. In the stand-alone environment, we consider algorithms based on UP-tree, utility list and extended utility list respectively. In the big data environment, we considered algorithms based on Hadoop platform and Spark platform. In addition, there are some brief discussions on the appropriate pruning strategies in the two environments. Finally, the paper also discusses several possible future research directions on frequent high-utility itemsets mining. In the future, if there is more in-depth research on mining frequent high-utility itemsets, the content of this article can be expanded.

REFERENCES

[1] M. J. Zaki, S. Parthasarathy, M. Ogihara, W. Li *et al.*, "New algorithms for fast discovery of association rules," in *In 3rd International Conference on Knowledge Discovery and Data Mining*, vol. 97, 1997, pp. 283–286.

[2] J. M.-T. Wu, Q. Teng, S. Huda, Y.-C. Chen, and C.-M. Chen, "A privacy frequent itemsets mining framework for collaboration in iot using federated learning," *ACM Transactions on Sensor Networks (TOSN)*, 2022, https://doi.org/10.1145/3532090.

[3] 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.

[4] 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.

[5] H. Yao and H. J. Hamilton, "Mining itemset utilities from transaction databases," *Data & Knowledge Engineering*, vol. 59, no. 3, pp. 603–626, 2006.

[6] J. C.-W. Lin, J. Zhang, P. Fournier-Viger, T.-P. Hong, C.-M. Chen, and J.-H. Su, "Efficient mining of short periodic high-utility itemsets," in *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC).* IEEE, 2016, pp. 003 083–003 088.

[7] 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.

[8] Q. Lin, W. Gan, Y. Wu, J. Chen, and C.-M. Chen, "Joint utility and frequency for pattern classification," in *2021 IEEE International Conference on Big Data (Big Data).* IEEE, 2021, pp. 5524–5533.

[9] ——, "Smart system: Joint utility and frequency for pattern classification," *arXiv preprint arXiv:2206.04269*, 2022.

[10] 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.

[11] S.-J. Yen and Y.-S. Lee, "Mining high utility quantitative association rules," in *International Conference on Data Warehousing and Knowledge Discovery*. Springer, 2007, pp. 283–292.

[12] V. Podpecan, N. Lavrac, and I. Kononenko, "A fast algorithm for mining utility-frequent itemsets," *Constraint-Based Mining and Learning*, p. 9, 2007.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] V. Kumar, R. Kumar, V. Kumar, A. Kumari, and S. Kumari, "Ravcc: Robust authentication protocol for rfid based vehicular cloud computing," *Journal of Network Intelligence*, vol. 7, pp. 526–543, 2022.

[18] W. Weng, T. Li, J.-C. Liao, F. Guo, F. Chen, and B.-W. Wei, "Similarity-based attention embedding approach for attributed graph clustering," *Journal of Network Intelligence*, vol. 7, pp. 848–861, 2022.

[19] 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.

[20] 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.

[21] 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.

[22] R. Agrawal, R. Srikant *et al.*, "Fast algorithms for mining association rules," in *Proceedings of 20th International Conference on Very Large Data Bases (VLDB 1994)*, vol. 1215. Santiago, Chile, 1994, pp. 487–499.

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

[24] 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.

[25] 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.

[26] 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.

[27] J. M.-T. Wu, M. Wei, G. Srivastava, C.-M. Chen, and J. C.-W. Lin, "Mining large-scale high utility patterns in vehicular ad hoc network environments," *Transactions on Emerging Telecommunications Technologies*, vol. 33, no. 10, p. e4168, 2022.

[28] J. Liu, K. Wang, and B. C. Fung, "Direct discovery of high utility itemsets without candidate generation," in *2012 IEEE 12th International Conference on Data Mining*. IEEE, 2012, pp. 984–989.

[29] ——, "Mining high utility patterns in one phase without generating candidates," *IEEE Transactions on Knowledge and Data Engineering*, vol. 28, no. 5, pp. 1245–1257, 2015.

[30] 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.

[31] J. M.-T. Wu, Q. Teng, J. C.-W. Lin, and C.-F. Cheng, "Incrementally updating the discovered high average-utility patterns with the pre-large concept," *IEEE Access*, vol. 8, pp. 66 788–66 798, 2020.

[32] S. Zida, P. Fournier-Viger, J. C.-W. Lin, C.-W. Wu, and V. S. Tseng, "Efim: a fast and memory efficient algorithm for high-utility itemset mining," *Knowledge and Information Systems*, vol. 51, no. 2, pp. 595–625, 2017.

[33] 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.

[34] 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.

[35] C.-W. Lin, T.-P. Hong, and W.-H. Lu, "Efficiently mining high average utility itemsets with a tree structure," in *Asian Conference on Intelligent Information and Database Systems*. Springer, 2010, pp. 131–139.

[36] H. M. Nguyen, A. V. Phan, and L. Van Pham, "Fskymine: a faster algorithm for mining skyline frequent utility itemsets," in *2019 6th NAFOSTED Conference on Information and Computer Science (NICS)*. IEEE, 2019, pp. 251–255.

[37] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[38] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," in *2nd USENIX Workshop on Hot Topics in Cloud Computing (HotCloud 10)*, 2010.

[39] S. Salloum, R. Dautov, X. Chen, P. X. Peng, and J. Z. Huang, "Big data analytics on apache spark," *International Journal of Data Science and Analytics*, vol. 1, no. 3, pp. 145–164, 2016.

[40] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauly, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A {Fault-Tolerant} abstraction for {In-Memory} cluster computing," in *9th USENIX Symposium on Networked Systems Design and Implementation (NSDI 12)*, 2012, pp. 15–28.

[41] M. Armbrust, T. Das, A. Davidson, A. Ghodsi, A. Or, J. Rosen, I. Stoica, P. Wendell, R. Xin, and M. Zaharia, "Scaling spark in the real world: performance and usability," *Proceedings of the VLDB Endowment*, vol. 8, no. 12, pp. 1840–1843, 2015.

[42] Y. Benlachmi and M. L. Hasnaoui, "Big data and spark: Comparison with hadoop," in *2020 Fourth World Conference on Smart Trends in Systems, Security and Sustainability (WorldS4)*. IEEE, 2020, pp. 811–817.

[43] M. Dessokey, S. M. Saif, S. Salem, E. Saad, and H. Eldeeb, "Memory management approaches in apache spark: A review," in *International Conference on Advanced Intelligent Systems and Informatics*. Springer, 2020, pp. 394–403.

[44] 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.

[45] 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 Exngineering*, vol. 28, no. 1, pp. 54–67, 2015.

[46] W. Gan, S. Wan, J. Chen, C.-M. Chen, and L. Qiu, "Tophui: Top-k high-utility itemset mining with negative utility," in *2020 IEEE International Conference on Big Data (Big Data)*. IEEE, 2020, pp. 5350–5359.

[47] C.-W. Lin, W. Gan, T.-P. Hong, and C.-M. Chen, "Maintaining high-utility itemsets in dynamic databases," in *2014 International Conference on Machine Learning and Cybernetics*, vol. 2. IEEE, 2014, pp. 469–474.

[48] L. Chen, W. Gan, Q. Lin, J. Miao, and C.-M. Chen, "Mining on-shelf high-utility quantitative itemsets," in *2021 IEEE International Conference on Big Data (Big Data)*. IEEE, 2021, pp. 5491–5500.