applied sciences

Article

# Mining Top-*k* High Average-Utility Sequential Patterns for Resource Transformation

**Kai Cao** [1,2] **and Yucong Duan** [3,*]

1 School of Cyberspace Security, Hainan University, Renmin Avenue 58, Haikou 570228, China; ck@hainanu.edu.cn
2 School of Cryptology, Hainan University, Renmin Avenue 58, Haikou 570228, China
3 School of Computer Science and Technology, Hainan University, Renmin Avenue 58, Haikou 570228, China
* Correspondence: duanyucong@hotmail.com

**Abstract:** High-utility sequential pattern mining (HUSPM) helps researchers find all subsequences that have high utility in a quantitative sequential database. The HUSPM approach appears to be well suited for resource transformation in DIKWP graphs. However, all the extensions of a high-utility sequential pattern (HUSP) also have a high utility that increases with its length. Therefore, it is difficult to obtain diverse patterns of resources. The patterns that consist of many low-utility items can also be a HUSP. In practice, such a long pattern is difficult to analyze. In addition, the low-utility items do not always reflect the interestingness of association rules. High average-utility pattern mining is considered a solution to extract more significant patterns by considering the lengths of patterns. In this paper, we formulate the problem of top-*k* high average-utility sequential pattern mining (HAUSPM) and propose a novel algorithm for resource transformation. We adopt a projection mechanism to improve efficiency. We also adopt the sequence average-utility-raising strategy to increase thresholds. We design the prefix extension average utility and the reduced sequence average utility by incorporating the average utility into the utility upper bounds. The results of our comparative experiments demonstrate that the proposed algorithm can achieve sufficiently good performance.

**Keywords:** DIKW graph; resource transformation; sequential pattern; average utility; top-*k*; pattern mining

check for updates

**Citation:** Cao, K.; Duan, Y. Mining Top-*k* High Average-Utility Sequential Patterns for Resource Transformation. *Appl. Sci.* **2023**, *13*, 12340. https://doi.org/10.3390/app132212340

Academic Editor: Keun Ho Ryu

Received: 19 September 2023
Revised: 17 October 2023
Accepted: 26 October 2023
Published: 15 November 2023

## 1. Introduction

Various resources, data, and information are increasingly being collected from large-scale intelligent devices in more diverse forms. However, some issues still need to be addressed in the processes of storing and utilizing these data, such as data security and processing efficiency. Some studies have redefined the concepts of the resources, as well as their hierarchical relationships and resource modeling, and attempted to examine privacy and security protection, model interpretability in data applications, and uncertainty or loss of control issues in artificial intelligence from this perspective [1]. There are some complex frameworks beyond the traditional DIKW pyramid that have been proposed to address the complex interrelationships between data in the real world. A novel knowledge graph system architecture, namely DIKW graphs, has been proposed, which is further subdivided into data graphs (D), information graphs (I), knowledge graphs (K), and wisdom graphs (W) [2]. The knowledge graphs describe the complex relationships between entities and concepts in a structured form, and the conceptual approaches for defining $Data_{DIK}$, $Information_{DIK}$ and $Knowledge_{DIK}$ are described in Ref. [3].

Based on the theory of relation definition semantics of everything (RDXS) [4], an extended metamodel was proposed in Ref. [5]. A conceptual framework is proposed under the axioms and theorems system in the existence computation (EC) [4,6], defining concepts such as "class/type" and "association". From the perspective of RDXS, a semantic transformation method has been proposed to convert relational semantics containing discrete independent entities to entity semantics within RDFS [3]. The conceptualization of "entity" and "relationship" have been proposed as forming the theoretical basis for assessing semantic similarity or "sameness" between the compared elements [5,7].

The established DIKW graphs model has been expanded into the more comprehensive DIKWP framework by adding proposed graphs (P) [8]. A proposed graph represents the process of extracting meaning and value from raw data, containing the objectives and intentions of artificial consciousness and driving the transformation between data, information, knowledge, and wisdom. In the DIKWP model, data are raw facts and numbers without processing. Information comprises data that have been given meaning and insight by being interpreted and understood in a particular context. Knowledge is a deeper understanding and familiarity with information, including the recognition of patterns, associations, and relationships within information. Wisdom involves applying knowledge, which involves synthesis, evaluation, and judgment to make wise decisions and take prudent actions. The collected data needs to be transformed into information through interpretation and understanding. The information is transformed into knowledge through pattern recognition, trend analysis, or contextual understanding. Through deeper analysis, classification, organization, or reasoning of information, a deep understanding of knowledge is gained, which could be transformed into wisdom [9].

In typed DIKWP resources, information consists of associated data or data combinations that contain specific semantic content that can be used for analysis and interpretation. Data can be transformed into information through specific processing, such as analyzing the frequency of data across structural, temporal, or spatial dimensions to capture their associations. The associations between data, also represented as specific relationships in a directed graph, indicate that an order exists between the data. If patterns of associated or combined data are considered, sequence pattern mining (SPM) methods applicable to sequence databases can satisfy the processing of transforming data into information. From the information perspective, due to the existence of context, each data point may have specific semantics. The frequency could reflect an internal feature of data, while context-defined data requires another external evaluation criterion to distinguish each other. The concept of utility can be used to distinguish data with different contexts. In HUSPM, different utility functions can be defined to represent the external features of data across different dimensions. Additionally, the transformation of data into information does not necessarily always possess sufficient a priori knowledge, and therefore seeking to select appropriate thresholds to optimize transformation efficiency proves challenging. Top-$k$ methods can achieve a goal-driven exploration by mapping user objectives to the parameter $k$.

SPM has found applications in web clickstream mining [10] and consumer behavior analysis [11]. However, in the frequent-oriented framework, patterns are extracted based solely on their support. Each of the items in a transaction has intrinsic value in the real world, and the value is unique. This frequent-oriented framework fails to consider the intrinsic value of individual items and does not capture the uniqueness of patterns. As is well known, the main task of retail business is to make more profits than sales. To overcome this problem, the relative importance of items was considered [12,13]. By incorporating the concept of utility, high-utility pattern mining (HUPM) [14] was introduced within a utility-oriented framework [15,16]. HUPM focuses on discovering patterns with high utility values in non-binary databases [17,18]. The utility value of an item is determined by multiplying its internal utility (which represents its quantity) by its external utility (which represents its unique value).

Sequential patterns based on pattern frequency do not always correspond to the purpose of analysis, such as selling quantity and profit. Therefore, the SPM methods do not always provide sufficient information to support the decisions of stakeholders. The obvious contrast is between luxury goods and everyday supplies. Since each item in the sequential database is accompanied by a timestamp, traditional high-utility itemset mining (HUIM) methods are unable to effectively address this aspect. To tackle this issue, HUSPM [19,20] has been developed in the field of knowledge discovery in databases. It could be applied to various scenarios [21].

In HUSPMA, a quantitative sequence database has been defined to represent the relative importance of patterns. In the context of retail business, the number of intelligent appliances or batteries purchased by a customer in a single visit is considered to be internal utility, reflecting the number of item occurrences. The unit profit of intelligent appliances or batteries is generally defined as external utility, indicating the relative importance of the item. The main task of HUSPM is to discover high-utility sequential patterns (HUSPs), which are subsequences with high utility values. Researchers have proposed many efficient algorithms for mining HUSPs [22,23], such as designing efficient pruning strategies and incorporating novel data structures. Notably, compared to SPM and HUIM, HUSPM takes into account the chronological ordering of items and their associated utility values [24]. Although HUSPM can overcome some disadvantages of SPM and HUIM, there are still some typical challenging problems in HUSPM.

The utility of a combined pattern is determined by the sum of the utilities of subpatterns constituting this combined pattern. Therefore, as the number of combined subpatterns increases, the length of the pattern also increases, leading to a higher utility value. This implies that a longer pattern has an advantage over a shorter one in terms of being a high-utility pattern. As the pattern length increases and the number of subpatterns grows, the association rules between these patterns become more difficult to comprehend [25]. Therefore, Hong et al. proposed the average-utility measurement, which offers a more comprehensive assessment of itemset utility [26]. The high average-utility itemset (HAUI) mining approach [27,28] is designed to extract patterns with high average-utility value by considering both the utility of the pattern and its length [28,29]. In this way, it is more objective to assess the utility of the pattern. A high average-utility pattern indicates that each item comprising the pattern has a high utility value. Consequently, patterns with long lengths and high utilities may not always qualify as high average-utility patterns.

There is another limitation in HAUI and HUSP mining algorithms. In most cases, it is challenging to specify an appropriate value for the minimum utility threshold that ensures sufficient but not excessive itemset mining, particularly when they are unfamiliar with the features of the database. For instance, when designing a general model or analyzing a new database, users are often unaware of the distribution of utilities and the total number of sequences, so they must try different threshold parameters and execute the algorithm multiple times until they obtain a desirable result. If the threshold is specified too low, many HUSPs with redundant and unimportant information may be discovered. Conversely, if the threshold value is set too high, the number of captured HUSPs may not provide sufficient information. Additionally, fine-tuning the threshold extraction process is time-consuming. However, in many practical applications, stakeholders are more interested in identifying the top profitable patterns rather than hundreds of thousands of results. The top-*k* HUSPM was proposed to handle this problem [30], which was desired to specify the number of patterns instead of the minimum utility threshold. It draws inspiration from the top-*k* HUIM [31]and top-*k* SPM [32]. The objective of the top-*k* HUSPM is to select patterns with the top-*k* highest utilities from a sequential database.

Some preliminary studies [33] were conducted to capture top-*k* HAUI in the transaction database. However, research in this area is still in its early stages, and these strategies are not applicable to sequential databases. The approach proposed by Thilagu et al. was limited to specific cases where each item can occur only once [34]. Tin et al. [35] proposed the algorithm EHAUSM to identify HAUSP. However, there is a gap between the overestimation of utility, which is calculated by the total sequence utility or the maximum item utility, and the actual utility, with the former being consistently higher. Furthermore, the proposed upper bounds, which are similar to those in HAUPM, require a set number of items with the highest utility in the rest of the sequence through multiple sorting. To address these challenges and to improve performance and scalability, we formulate the problem of top-*k* HAUSPM and propose a novel algorithm named TKAUS. The main contributions could be summarized as follows:

- By considering the HUSPM and the HAUIM, the concept of TKAUS is addressed, and the problem of top-*k* HAUSPM is formulated. To quickly increase the minimum average-utility threshold, we investigate the sequence average-utility-raising (SAUR) strategy.
- Based on the widely accepted definition of high average utility, a novel algorithm with utility upper bounds and corresponding pruning strategies is designed for mining HAUSP. The proposed approach not only takes into account the actual utility of prefixes but also avoids extra sorting processing.
- Comprehensive experiments demonstrate that the proposed algorithm TKAUS performs excellently for top-*k* HAUSPM, particularly in terms of execution time, unpromising candidate filtering, memory usage, and scalability.

The remainder of this paper is organized as follows. Section 2 briefly reviews various related work. In Section 3, we provide definitions and formulate the top-*k* HAUSPM problem. Section 4 presents our proposed TKAUS algorithm, including data structures and several strategies. Section 5 presents experimental results and evaluates the performance of our proposed algorithm. Finally, in Section 6, we provide a summary and discuss future work.

## 2. Related Work

### 2.1. Sequential Pattern Mining

SPM was defined by Agrawal and Srikant in 1995. The proposed algorithm AprioriAll was applied to analyze customer consumption records [36]. Later, repeated database traversal in the GSP method [11] incurred very high computational costs. According to the combinatorial property, SPADE [37] and SPAM [38] resolved the repetitive scanning problem, but when handling dense datasets, they posed a combinatorial explosion. Yang et al. [39] presented LAPIN and how to judge whether a frequent sequential pattern can be extended. Han et al. presented FreeSpan [40] and PrefixSpan [41] to resolve the problem that the Apriori-based SPM algorithms generate many unpromising candidates by utilizing projection strategies. However, the pattern-growth SPM algorithms that built projected databases recursively also incurred high computational costs. To avoid storing unpromising sequences, DISC-all [42] was proposed based on early pruning strategies.

### 2.2. High-Utility Sequential Pattern Mining

The frequency-oriented pattern mining framework does not directly correlate with significance in any circumstances. The mining framework of HUSPs considers both external and internal utilities and establishes the relationship between frequency and significance in the real world. Ahmed et al. [43] introduced the concept of utility into SPM and defined HUSPM. They proposed two two-phase algorithms: Utility Span (US) and Utility Level (UL). US utilizes a pattern-growth method without generating candidates, while UL adopts a level-wise approach to mine HUSPs. To avoid additional scans for identifying HUSPs, another efficient algorithm, UM-Span [21], utilizes a projected database-based approach. Yin et al. [20] presented the method USpan, which adopts an efficient prefixed tree structure, namely lexicographic quantitative sequence tree (LQS-tree), to avoid multiple unnecessary database scans. In LQS-tree, each node only stores the necessary information of the candidate sequence in a matrix, and its child nodes can be extended through one extension operation. Two pruning strategies were utilized: SWU and SPU. The former SWU removed unpromising items, and the latter SPU stopped USpan traversing deeper nodes. Lan et al. [44] presented an efficient projection-based HUSPM algorithm named PHUS, which utilizes the sequence-utility upper bound (SUUB) model and the maximum utility measure. More accurate upper bounds of candidate utilities could be obtained with the effective projection-based pruning strategy. The method HuspExt [45] used the cumulate rest of match (CRoM) as a tighter upper bound to eliminate candidate items early. To reduce the search space, Wang et al. [46] designed two tighter utility upper bounds, reduced sequence utility (RSU) and prefix extension utility (PEU), in algorithm HUS-Span to remove unpromising patterns early. ProUM [23] utilizes the utility-array as the projection of the original database for multiple scanning based on the prefix sequences. HUSP-ULL [24] designs a quite compact data structure, which was named UL-list, to store information about patterns. In contrast to existing data structures, these compressed store and index structures are used to calculate the utility efficiently.

### 2.3. Top-k High-Utility Sequential Pattern Mining

To extract the required information without using a minimum threshold, Yin et al. [30] designed a projection-based top-*k* HUSPM algorithm called TKHUS-Span. TUS, a method proposed by Yin et al. [30], is the first algorithm for top-*k* HUSPM and extends their preliminary work on USpan to discover patterns without a minimum threshold. Wang et al. [46] further developed three versions of the algorithm TKHUS-Span. All of them are based on the algorithm HUS-Span. Among these, the BFS strategy-based version of TKHUS-Span performs better than other versions. With limited memory space, the hybrid search strategy-based version demonstrates the best performance.

TU-SEQ was introduced by Zihayat et al. [47] for extracting the top-*k* gene regulation-related patterns over time from a microarray dataset. It utilizes the vertical data structure ItemUtilList and pre-evaluation using the genes and sequences (PES) strategy. Zhang et al. [48] formulated the problem of top-*k* HUSPM and proposed the TKUS algorithm for sufficiently good performance.

### 2.4. High Average-Utility Pattern Mining

The calculation of average utility in HAUIM algorithms takes into account the length of pattern. This approach helps to identify short patterns that may be missed by traditional algorithms. Two-phase average-utility (TPAU) mining [26,49] is the first HAUI mining algorithm. The average utility value does not satisfy the downward closure property. To address this limitation, TPAU proposed an average-utility upper bound auub, which maintains the property, because the value of auub is always larger than the average utility of pattern. The auub is utilized to improve mining performance by pruning. However, TPAU is bound by repeated scanning of the original database and generating a huge amount of candidate patterns.

HAUI-Growth [50] was designed to limit the generation of candidates and to overcome multiple scanning. The average utility of each itemset was maintained as an array in the node of a HAUI-tree structure. Another approach, PBAU, proposed by Lan et al. [28], is the projection-based mining algorithm, which uses index tables to reveal HAUIs. They used a tree structure to reduce the number of scans in the original database [27]. An enumeration tree structure was used in the algorithm HAUI-Tree [51]. However, the number of candidates generated by these tree-based methods is still high.

With the list structure, MHAI [52] and HAUI-Miner [53] adopted the transaction-maximum utility and the maximum average utility as the utility upper bound to prune unpromising patterns, respectively. EHAUPM [54] proposed a revised tighter upper bound model (rtub) and looser upper bound (lub) and presented MAU-list based on a list structure. In the transaction database, to reduce the unpromising patterns, TUB-HAUPM [55] proposed maximum following utility upper bound (mfuub), transaction-rival tight upper bound (trtub), and top-*k* revised transaction-maximum utility upper bound (krtmuub). By considering both vertical information of other transactions and horizontal information in the same transaction, VMHAUI [56] suggested three upper bounds by introducing vertical approaches [57]. By employing tighter upper bounds, the algorithm can hugely reduce the search space. Considering the protection of sensitive content, Le et al. designed an algorithm H-FHAUI [58] with a novel vertical utility list structure, TIU-VIU, for updating upper bounds quickly. Kim et al. [59] employed the maximum remaining average-utility upper bound and tight maximum average-utility upper bound and proposed a HAUPM approach with the list-based structure.

It is a limitation of HUSPM that the utility function tends to be biased toward discovering longer patterns. Based on the pattern-growth approach, Thilagu et al. [34] proposed an approach to reveal the effect of pattern length in a web traversal sequence. The proposed average transaction-weighted utility (atwu) used the high average-utility concept, which was introduced in Ref. [31]. The search space was divided by the transaction-weighted utility (TWU) of each pattern, and the number of candidates was reduced. However, each item can occur more than once in most cases. This approach proposed by Thilagu et al. was inapplicable to the general case [35]. Tin et al. [35] proposed anti-monotonic upper bound (AMUB), bi-directional upper bound (BiUB), and four pruning strategies in algorithm EHAUSM. Three weak upper bounds, twaub, rmwub, and tmwub, were proposed to prune all proper extensions of a sequence early.

Moreover, Lin et al. [60] considered the size of a sequence and proposed a framework of SPM for finding potential HAUSPs from the uncertain dataset. Wu et al. proposed HAOP-Miner [61] and HANP-Miner [62] to discover the sequential pattern under the one-off condition and nonoverlapping sequential pattern. The reverse filling (Rf) algorithm was designed to avoid creating redundant nodes and to calculate the support effectively. The simplified Nettree structure was adopted in depth-first search and backtracking strategies [62] to reduce the algorithm complexity of HANP-Miner. Then, Tin et al. proposed C-FHAUSPM [63], which satisfies monotonic and anti-monotonic constraints, to find constrained HAUSPs. To consider the average cost for mining, Tin et al. formalized the FLCHUSM problem and proposed algorithm FLCHUSPM [64] with an upper bound on average utility and a lower bound on average cost.

The average-utility measure of the quantitative sequence is neither anti-monotonic nor monotonic. Generally, designing an upper bound that holds for the anti-monotone property is an effective approach to reducing the search space of the algorithm. Some of these upper bounds were calculated by the total sequence utility, including the ATWU and AMUB. Therefore, there is a gap between the actual value and the upper bound, and the wider this gap is, the more candidate patterns are generated. Some other upper bounds must sort items by their utilities for each computation, e.g., BiUB and tmwub. The proposed approach designs upper bounds by considering the actual utility of prefixes such as krtmuun and rmwub. There are no extra sorting costs.

## 3. Mining Top-*k* High Average-Utility Pattern

In this section, several definitions are introduced to explain the framework of TKAUS. Table 1 shows an example.

Consider a set of distinct items $I = \{i_1, i_2, \cdots, i_M\}$, $X$ is a nonempty subset of $I$, denoted as $X \subseteq I$, and $|X|$ represents the size of $X$. A sequence $S : \langle X_1, X_2, \cdots, X_m \rangle$ is an ordered list of itemsets, where $X_k \subseteq I, (1 \leqslant k \leqslant m)$. The length of $S : \langle X_1, X_2, \cdots, X_m \rangle$ is calculated as $l = \sum_{k=1}^{m} |X_k|$, and is called $l$-sequence. The size of $S : \langle X_1, X_2, \cdots, X_m \rangle$ is $m$. Assume that there exists integers $1 \leqslant j_1 < j_2 < \cdots < j_n \leqslant m$ such that $X_v{}' \subseteq X_{j_v}, (1 \leqslant v \leqslant n)$, the sequence $S : \langle X_1, X_2, \cdots, X_m \rangle$ has a subsequence $s : \langle X_1{}', X_2{}', \cdots, X_n{}' \rangle$, notated as $S \supseteq s$ or $s \subseteq S$.

The definitions of the quantitative sequence and the quantitative sequence database have been proposed in [48]. A quantitative sequence database consists of quantitative sequences ($q$-sequences). The $q$-sequence is an ordered list of the quantitative itemset ($q$-itemset). The quantitative item ($q$-item) is expressed in the form of (item, quantity) and denoted as $(i,q)$. In the tuple, the quantity of the item is represented by the internal utility value of the item. In Table 1, $Q_1$ is an identifier of $q$-sequence, and it is unique.

**Table 1.** An illustration of the quantitative sequence database.

| SID | Q-Sequence |
|---|---|
| $Q_1$ | $\langle\{(a:4)(c:2)\},\{(e:3)(f:1)\},\{(a:1)(b:1)\},\{(d:4)(g:1)\},\{(a:1)(c:3)(e:1)(f:2)\},\{(b:1)(d:2)\}\rangle$ |
| $Q_2$ | $\langle\{(a:3)(d:2)\},\{(a:5)(f:1)\},\{(c:1)(d:3)(e:1)(g:2)\},\{(b:2)\}\rangle$ |
| $Q_3$ | $\langle\{(b:2)(c:1)\},\{(a:1)(b:1)(e:1)\},\{(a:2)(b:2)(f:2)\},\{(g:3)\}\rangle$ |
| $Q_4$ | $\langle\{(c:1)(e:1)\},\{(a:1)(b:3)(g:4)\},\{(f:2)\}\rangle$ |
| $Q_5$ | $\langle\{(c:1)(f:2)\},\{(a:1)(b:1)(f:1)(g:3)\},\{(b:1)(e:1)(g:3)\}\rangle$ |

**Definition 1 (Q-item Utility).** *Consider a q-item $(i{:}q)$ in the jth q-itemset within a q-sequence $Q$, the q-item utility is denoted as $u(i,j,Q)$ and is calculated as follows:*

$$u(i,j,Q) = q(i,j,Q) \times eu(i)$$

*where $eu(i)$ is the external utility of the item i, and $q(i,j,Q)$ is the internal utility of the item i and is a quantitative measure for the q-item.*

**Definition 2 (Q-itemset Utility).** *Let $Y:\{(i_1{:}q_1)(i_2{:}q_2)\cdots(i_n{:}q_n)\}$ is a q-itemset and is the jth q-itemset in a q-sequence Q. The q-itemset utility is the sum of the q-item utilities. It is denoted as $u(Y,j,Q)$ and is calculated as follows:*

$$u(Y,j,Q) = \sum_{u=1}^{n} q(i_u,j,Q) \times eu(i_u)$$

The profit of the item as the external utility values is provided in Table 2. For example, as Definition 1, consider the 3rd $q$-itemset of $Q_2$ in Table 1, the utility of $q$-item $d$ is $q(d,3,Q_2) \times eu(d) = 3 \times 2 = 6$. Then, we have the utility of the 3rd $q$-itemset in $Q_2$ is calculated as $u(Y,3,Q_2) = u(c,3,Q_2) + u(d,3,Q_2) + u(e,3,Q_2) + u(g,3,Q_2) = 8 + 6 + 7 + 8 = 29$. $D$ represents the $q$-sequence database in Table 1, and the overall utility of $D$ is $u(D) = 123 + 56 + 63 + 59 + 73 = 374$.

**Table 2.** The item profit table.

| Item | *a* | *b* | *c* | *d* | *e* | *f* | *g* |
|---|---|---|---|---|---|---|---|
| **Profit** | 1 | 3 | 8 | 2 | 7 | 9 | 4 |

**Definition 3 (Q-itemset Average Utility).** *Let $Y{:}\{(i_1{:}q_1)(i_2{:}q_2)\cdots(i_n{:}q_n)\}$ is the jth q-itemset in a q-sequence Q. Its average utility is denoted as $au(Y,j,Q)$ and is calculated as follows:*

$$au(Y,j,Q) = \frac{\sum_{u=1}^{n} q(i_u,j,Q) \times eu(i_u)}{|Y|} = \frac{\sum_{u=1}^{n} q(i_u,j,Q) \times eu(i_u)}{n}$$

For instance, in Table 1, $\{a,c\}$ is the 1st $q$-itemset in $Q_1$, the average utility of $\{a,c\}$ is calculated as $au(\{a,c\},1,Q_1) = \frac{(4\times1)+(2\times8)}{2} = 10$.

**Definition 4 (Match).** *Consider the q-itemset $Y{:}\{(i'_1{:}q_1)(i'_2{:}q_2)\cdots(i'_n{:}q_n)\}$ and the itemset $X{:}\{i_1,i_2,\cdots,i_m\}$, if and only if $n = m$ such that $i'_k = i_k,(1 \leqslant k \leqslant m = n)$, we say that X matches Y or Y matches X, denoted as $X \sim Y$ or $Y \sim X$. Similarly, the q-sequence $Q{:}\langle Y_1,Y_2,\cdots,Y_n\rangle$ matches the sequence $S{:}\langle X_1,X_2,\cdots,X_m\rangle$, denoted as $S \sim Q$ or $Q \sim S$, if and only if $n = m$ such that $Y_k \sim X_k,(1 \leqslant k \leqslant m = n)$.*

**Definition 5 (Contain).** *Consider the q-itemset $Y{:}\{(i'_1{:}q_1)(i'_2{:}q_2)\cdots(i'_n{:}q_n)\}$ and the itemset $X{:}\{i_1,i_2,\cdots,i_m\}$, we say that Y contains X or X is contained in Y, denoted as $Y \sqsupseteq X$ or $X \sqsubseteq Y$, if and only if X is a subset of itemset and the itemset matches the q-itemset Y.*

For instance, from Tables 1 and 2, the itemset $\{a, b, f\}$ *matches* the $q$-itemset $\{(a{:}2), (b{:}2), (f{:}2)\}$ in the sequence $Q_3$. In $Q_5$, the $q$-itemset $\{(a{:}1), (b{:}1), (f{:}1), (g{:}3)\}$ *contains* the itemset $\{a, b, f\}$.

**Definition 6 (Instance of *Q*-sequence).** *Consider the q-sequence* $Q{:}\langle Y_1, Y_2, \cdots, Y_n\rangle$ *and the sequence* $S{:}\langle X_1, X_2, \cdots, X_m\rangle$*, where* $n \geqslant m$*, we say that there is an instance of S at position* $p{:}\langle j_1, j_2, \cdots, j_m\rangle$ *in Q, if and only if there exists positive integers* $1 \leqslant j_1 < j_2 < \cdots < j_m \leqslant n$*, so that* $X_v \sqsubseteq Y_{j_v}, (1 \leqslant v \leqslant m \leqslant n)$*. Also, we can say Q contains S, denoted as* $S \sqsubseteq Q$*.*

For instance, there are two *instances* of sequence $\langle\{a, b\}, \{g\}\rangle$, $\langle\{(a{:}1), (b{:}1)\}, \{(g{:}3)\}\rangle$ and $\langle\{(a{:}2), (b{:}2)\}, \{(g{:}3)\}\rangle$, at positions $\langle 2, 4\rangle$ and $\langle 3, 4\rangle$ in $Q_3$. Therefore, we could also say the $q$-sequence $\langle\{(a{:}1), (b{:}1)\}, \{(g{:}3)\}\rangle$ or $\langle\{(a{:}2), (b{:}2)\}, \{(g{:}3)\}\rangle$ *contains* the sequence $\langle\{a, b\}, \{g\}\rangle$.

**Definition 7 (Sequence Utility).** *Assume that the q-sequence* $\langle Y_{j_1}, Y_{j_2}, \cdots, Y_{j_m}\rangle$ *is an instance of sequence* $S{:}\langle X_1, X_2, \cdots, X_m\rangle$ *at position* $p{:}\langle j_1, j_2, \cdots, j_m\rangle$ *in* $Q{:}\langle Y_1, Y_2, \cdots, Y_n\rangle$*. The sequence utility of S in Q is the sum of the utilities of the q-items within S, denoted as* $u(S, p, Q)$ *and calculated as follows:*

$$u(S, p, Q) = \sum_{v=1}^{m} u(X_v, j_v, Q) = \sum_{v=1}^{m} u(Y_{j_v}, j_v, Q)$$

Generally, a sequence appears in one $q$-sequence more than once. Given $P(S, Q)$ denote the set of all the positions of $S$ in $Q$. The sequence utility of $S$ in $Q$ is the maximum value of instance utility at different positions and is defined as follows:

$$u(S, Q) = \max_{p \in P(S,Q)} u(S, p, Q)$$

Assume that the sequence $S$ has *instances* within different $Q$ in $q$-sequence database $D$. The sequence utility of $S$ in $D$ can be defined as follows:

$$u(S) = \sum_{Q \in D \wedge S \sqsubseteq Q} u(S, Q)$$

For example, in Table 1, $Q_1$ has an *instance* of $\langle\{e, f\}, \{g\}\rangle$ at position: $\langle 2, 4\rangle$. Then, we can calculate $u(\langle\{e, f\}, \{g\}\rangle, \langle 2, 4\rangle, Q_1) = 31 + 16 = 47$. For another case, in Table 1, $Q_1$ has five *instances* of $s_1{:}\langle a, d\rangle$, $P(s_1, Q_1) = \{\langle 1, 4\rangle, \langle 1, 6\rangle, \langle 3, 4\rangle, \langle 3, 6\rangle, \langle 5, 6\rangle\}$, $u(s_1, \langle 1, 4\rangle, Q_1) = 4 + 8 = 12$, $u(s_1, \langle 1, 6\rangle, Q_1) = 4 + 4 = 8$, $u(s_1, \langle 3, 4\rangle, Q_1) = 1 + 8 = 9$, $u(s_1, \langle 3, 6\rangle, Q_1) = 1 + 4 = 5$, and $u(s_1, \langle 5, 6\rangle, Q_1) = 1 + 4 = 5$. The utility of the sequence $s_1$ in $Q_1$ is $u(s_1, Q_1) = \max\{12, 8, 9, 5, 5\} = 12$, and $u(s_1) = u(s_1, Q_1) + u(s_1, Q_2) = 12 + 11 = 23$.

**Definition 8 (Sequence Average Utility).** *Consider the q-sequence* $\langle Y_{j_1}, Y_{j_2}, \cdots, Y_{j_m}\rangle$ *is an instance of sequence* $S{:}\langle X_1, X_2, \cdots, X_m\rangle$ *at position* $p{:}\langle j_1, j_2, \cdots, j_m\rangle$ *in* $Q{:}\langle Y_1, Y_2, \cdots, Y_n\rangle$*. The sequence average utility of S in Q is denoted as* $au(S, P, Q)$ *and is calculated as follow:*

$$au(S, p, Q) = \frac{u(S, P, Q)}{|S|} = \frac{\sum_{v=1}^{m} u(X_v, j_v, Q)}{\sum_{v=1}^{m} |X_v|} = \frac{\sum_{v=1}^{m} u(Y_{j_v}, j_v, Q)}{\sum_{v=1}^{m} |Y_{j_v}|}$$

In this paper, we define the sequence average utility based on the concept of average utility, which was defined as the pattern utility divided by its length. For instance, the average utility of $\langle\{c, e\}, \{f\}\rangle$ at position $\langle 1, 3\rangle$ in $Q_4$ is $au(\langle\{c, e\}, \{f\}\rangle, \langle 1, 3\rangle, Q_4) = \frac{15+9}{3} = 8$.

Assume that the set of all the positions of *instances* of $S$ in $Q$ is $P$:$\langle p_1, p_2, \cdots, p_w \rangle$ and is denoted as $P(S, Q)$, the sequence average utility of $S$ in $Q$ is the maximum value at different position and is defined as follows:

$$au(S, Q) = \frac{u(S, Q)}{|S|} = \max_{p \in P(S,Q)} au(S, p, Q)$$

Assume that the sequence $S$ has *instances* within different $Q$ in $q$-sequence database $D$. The average utility of $S$ in $D$ can be defined as follows:

$$au(S) = \frac{u(S)}{|S|} = \sum_{Q \in D \wedge S \sqsubseteq Q} au(S, Q)$$

For instance, in Tables 1 and 2, the sequence average utility of $s_2$:$\langle \{c\}, \{a, b\}, \{g\} \rangle$ in $Q_3$ is $au(s_2, Q_3) = \max\{au(s_2, \langle 1, 2, 4 \rangle, Q_3), au(s_2, \langle 1, 3, 4 \rangle, Q_3)\} = \max\{6, 7\} = 7$, and in the entire database $D$, is $au(s_2) = au(s_2, Q_1) + au(s_2, Q_3) + au(s_2, Q_5) = 6 + 7 + 6 = 19$.

**Definition 9 (Top-*k* High Average-Utility Sequential Pattern).** *We define a sequence $S$ as the top-k high average-utility sequence in a q-sequence database if there exist fewer than k sequences whose average-utility value is higher than $au(S)$.*

**Problem Statement.** The goal of conventional HUSPM is to find HUSPs in a sequence database with a user-specified utility threshold [46]. Zhang et al. [48] proposed that the threshold of top-*k* HUSPM can be represented as the minimum utility value of the complete set of top-*k* HUSPs, which is easily accessible.

The utility of pattern is affected by its length [24]. In addition, mining representative *k* patterns is a lossless compression of patterns [65]. Therefore, the drawback of HUPM is more prominent in top-*k* HUSPM.

## 4. Proposed TKAUS Algorithm

The fundamental technologies require multiple original database scans for the candidates. Multiple scanning also exists in utility calculation. High cost is a main problem. The projection and local search mechanism have been proposed to address the problem by constructing a projected database to keep the necessary information. Elimination of the unpromising items earlier also can greatly reduce the size of search space.

### *4.1. Data Structure and Projection Strategies*

In the framework of the proposed algorithm, the mining process first scans the given $q$-sequence database once. The lexicographic sequence tree as the complete search space has been used in SPM and HUSPM [38]. Then, USpan [19] presented the lexicographic quantitative sequence tree (LQS-tree) structure. Each node of this LQS-tree structure is associated with a candidate, which consists of the $q$-sequence and its utility. The root of this tree is null. All child nodes are arranged in ascending order, for instance, alphabetical order. The child node is a prefix-based extension of its parent node. There are two kinds of *extension positions* for the appending item in a sequence. One is the end of the last itemset. The other position is after the last itemset as a new itemset. Both $I$-Extension and $S$-Extension are operations of "*extension*" to generate new sequences. For example, consider a $l$-sequence $s$ and an appending item $i$, the size of $s$ is $k$, $I$-Extension of sequence is to append item $i$ to the end of last itemset in sequence $s$, notated as $s \oplus i$, the new generate sequence is $(l+1)$-sequence $s'$, the length of $s'$ is $(l+1)$ and the size of $s'$ still is $k$. $S$-Extension of sequence is to let itemset $i$ as a new one $\{i\}$ and append to the end of sequence $s$, notated as $s \otimes i$. In this way, the newly generated sequence also is $(l+1)$-sequence $s'$ which is $(l+1)$ in length, but the size of $s'$ is $(k+1)$.

In some cases, the *instances* of the sequence $S$ may appear more than one position, and the set of positions of *instances* is denoted as $P$:$\{p_1, p_2, \cdots, p_w\}$. Zhang et al. [48]

presented the position of the last itemset in *instance* is *extension position*. In $q$-sequence $Q$, the sequence utility of $S$ is the maximum utility of all *instances*. The last item within *instance* is the *extension item*. The *remaining sequence* is a subsequence from the item after the *extension item* to the end of $Q$, which is denoted as $rs(S, p, Q)$. Yin et al. defined the *remaining utility* of $S$ at the *extension position* $p$ in $Q$ as $u_{rs}(S, p, Q)$ [20].

$$u_{rs}(S, p, Q) = \sum_{i' \in Q \wedge i' \succ S} u(i')$$

where $i' \succ S$ indicates that the position of item $i'$ is after the *extension position* $p$. For instance, from Tables 1 and 2, the sequence $\langle\{f\}, \{g\}\rangle$ has *instances* at position $\langle 1, 3\rangle$, $\langle 2, 3\rangle$ and $\langle 1, 2\rangle$ in $Q_5$.The *extension position* of these *instances* are $\langle 3\rangle$ and $\langle 2\rangle$, and the item $g$ is the *extension item*. Thus, the *remaining utility* of $\langle\{f\}, \{g\}\rangle$ at *extension position* $\langle 3\rangle$ in $Q_5$ is $u_{rs}(\langle f, g\rangle, \langle 3\rangle, Q_5) = 0$, and at *extension position* $\langle 2\rangle$ is $u_{rs}(\langle f, g\rangle, \langle 2\rangle, Q_5) = 17$.

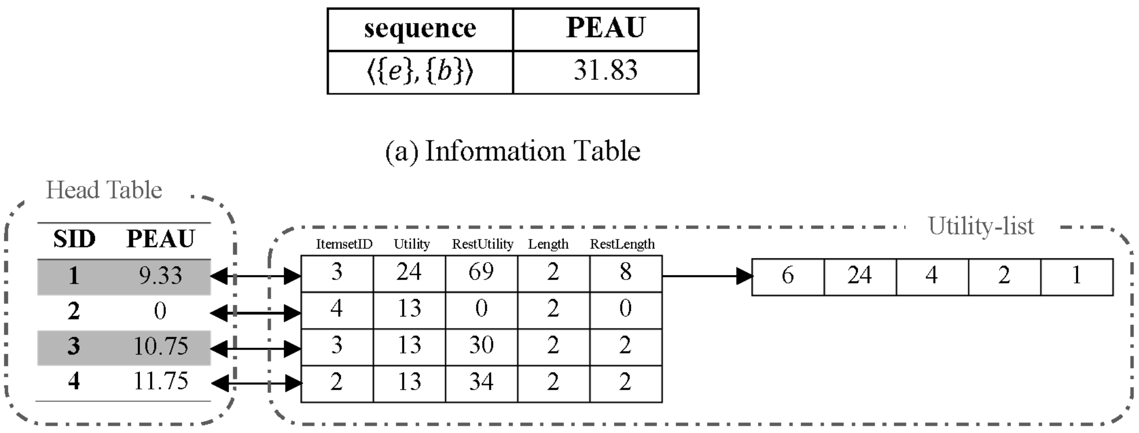
To calculate all the multiple *instances* in a $q$-sequence is time-consuming. In [46,48], the projected database was utilized to keep the necessary information in sequence. By adopting the utility-chain structure, these studies reduce calculation costs. The projected database of a sequence is composed of an information table and a utility chain. The utility chain in the structure consists of two parts: a head table and a utility list. The utility-chain structure gives the auxiliary information to facilitate pruning search space in HUS-Span [46].

In this work, the prefix extension average utility (abbreviated as PEAU) of the sequence is stored in the information table as a key piece. In addition to the maximum utility of all *instances* and the utility of *remaining sequence*, the utility list also contains the length of *instance* and *remaining sequence*. More details about PEAU are described in the subsequent section.

In the structure, the size of the utility list corresponds with the number of *extension positions* of the *instance* in the $q$-sequence. In the 1st field, the unique identity of itemset, which corresponds to the *extension position* is abbreviated as $ItemsetID$. In the 2nd and 3rd fields, $Utility$ and $RestUtility$, are the average-utility value of the *instance* and the *remaining sequence utility* at this *extension position*. In the 4th and 5th fields are length of the *instance* and *remaining sequence*, which are notated as $Length$ and $RestLength$, respectively.

The above projection mechanism constructs a precise and complete projected database to avoid multiple scanning for the original database. In the subsequent section, the algorithm adopts various pruning strategies to further limit scanning space.

For instance, Figure 1 shows the projected database of $s_3$: $\langle\{e\}, \{b\}\rangle$. There are four rows on the head table. It shows $s_3$ is contained in $Q_1$, $Q_2$, $Q_3$ and $Q_4$. The utility chain corresponds to the head table. The first line in the utility chain contains complete information about two *extension positions* in $Q_1$ for the following process. Therefore, the TKAUS algorithm avoids multiple scanning for the whole original database by the projection mechanism.



| sequence | PEAU |
|---|---|
| $\langle\{e\}, \{b\}\rangle$ | 31.83 |

(a) Information Table

Head Table

| SID | PEAU |
|---|---|
| 1 | 9.33 |
| 2 | 0 |
| 3 | 10.75 |
| 4 | 11.75 |

| ItemsetID | Utility | RestUtility | Length | RestLength |
|---|---|---|---|---|
| 3 | 24 | 69 | 2 | 8 |
| 4 | 13 | 0 | 2 | 0 |
| 3 | 13 | 30 | 2 | 2 |
| 2 | 13 | 34 | 2 | 2 |

| | | | | |
|---|---|---|---|---|
| 6 | 24 | 4 | 2 | 1 |

(b) Utility-chain

**Figure 1.** An illustration of the projected database.

### *4.2. Sequential Average-Utility-Raising Strategies*

The performance of the algorithm is limited due to more candidates being generated in the process of threshold increasing from a lower value. The sequence average-utility-raising (abbreviated as SAUR) strategy, which increases the rise rate of the minimum utility threshold, was proposed in TKAUS.

**Definition 10 (SAUR Strategy).** *Assume that $TKList = \{s_1, s_2, \cdots, s_n\}$ is the set of sequences contained in the q-sequence database D. Let $au_i$ denote the $i^{th}$ highest average-utility value in TKList. Before the mining task, we can safely increase the minimum average-utility threshold, which is a variable denoted as minau, to $au_k$, where k is the desired input number of patterns.*

**Proof.** Let *minau* is the minimum utility threshold of the mining process. When the variable *minau* increases to $au_k$ from 0, the set of HAUSPs, denoted as $H_{minau}$, is reduced from $H_0$ to $H_{au_k}$ ($H_0 \supseteq H_{au_k}$), and all the sequences whose average utility is not lower than $au_k$ must be in the set of $H_{au_k}$. According to the definition of top-*k* HAUSP, there are more than *k* sequences in $H_0$ whose average-utility values are not lower than $au_k$. Thus, the SAUR strategy effectively increases the *minau* to a rational level. In addition, we prove that the method with the SAUR strategy will not miss any top-*k* HAUSPs. □

### *4.3. Upper Bounds and Pruning Strategies*

There is an inherent time order between items in the sequence database. The process of HUSPM must deal with the critical combinatorial explosion of the search space [48]. If the maximum sequence utility is much lower than the minimum utility threshold, even the sum of sequence utility and utilities of items in the *remaining sequence* still cannot reach the threshold. The sequence is "*unpromising*". Zhang et al. [48] proposed the upper bound and corresponding strategy to reduce unpromising patterns and unnecessary operations. Several existing studies have proposed some sequence pattern mining upper bounds. Three typical upper bounds: sequence extension utility (SEU) [22], sequence projected utilization (SPU) [29], and sequence-weighted utilization (SWU) [19] had been introduced compared in ref. [23]. SWU is a loose upper bound and is an overestimation, SPU loses some real HUSPs, and SEU, which is a modified real upper bound, could further limit the search scope. Wang et al. [46] proposed PEU upper bound and RSU upper bound to speed up HUSPM. The PEU upper bound was defined as $PEU(t, p, s) = u(t, p, s) + ru(s/_{(t,p)})$.

The existing research work is instructive to improve the performance of HAUSPM by reducing the *unpromising* pattern. The transaction-maximum average-utility upper bounds designed by Kim et al. [59] or proposed by Wu et al. [55] need to sort the set of remaining items by utility descending order. However, in a quantitative sequence database, each *q*-itemset has a special order, so sorting among *q*-items that contain different *q*-itemsets is not an effective method.

In fact, each item in *remaining sequence* is helpful to add up to higher sequence utility by the "*extension*" operation for HUSPM and algorithm TKUS. The item is *promising* if the value of PEAU is not lower than the minimum average-utility threshold. However, in HAUSPM and our proposed algorithm, the average utility of the sequence, which is generated by the "*extension*" operation, is not higher than one of its prefixes if the utility of the appending item is lower than the average utility of the prefix.

**Definition 11 (Increment of Sequence Utility).** *Let $u(s)$ be the sequence utility of s. If $u(s)$ is in excess of the minimum average-utility threshold minau, let $u_{INCR}(s)$ be the rising proportion of the excess sequence-utility value in $u(s)$, and is calculated as follows:*

$$u_{INCR}(s) = u(s) - minau \times |s|,$$

Assume that there is an *instance* of sequence *S* at the *extension position* $p : <j, j_2, \cdots, j_m>$ in *q*-sequence *Q*, the corresponding *remaining sequence* is *rs* which is the rest after *p* to the end. Let the $rs'$ be a subsequence consisting of items that have a higher utility value than *minau* in *rs*. The rising proportion of the excess sequence-utility value in *rs* is denoted as $u_{INCR}(rs)$, and is calculated as follows:

$$u_{INCR}(rs') = \sum_{Q \in D \wedge S \sqsubseteq Q} u_{rs'}(S, j_m, Q) - minau \times |rs'|,$$

where the $|rs'|$ is the maximum length of $rs'$ of the prefix $S$ in database $D$. Please note that $j_m$ is determined by the *extension position* of sequence $S$.

Therefore, we define the $u_{INCR}(rs')$. It is a measure for the promotion of the *remaining sequence* utility on the utility of the generated candidates. The concept of "*unpromising*", which has been mentioned at the beginning of this section, can be redefined. The appending item is *unpromising* if the average utilities of all the generated sequences containing this item still are lower than *minau*, i.e., all the promotion of its prefix and its *remaining sequence* will not increase the sequence average utility to the value of *minau*. We will, therefore, give the following upper bounds and pruning strategies.

**Definition 12 (*PEAU* Upper Bound).** *Assume that there is an instance of sequence S at the extension position p:*$\langle j, j_2, \cdots, j_m \rangle$ *in q-sequence Q, the corresponding remaining sequence is rs which is the rest after p to the end. Let rs′ be a subsequence of rs and consist of items that have a higher utility value than minau. The prefix extension average utility of S in q-sequence database D is defined as follows:*

$$PEAU(S) = \begin{cases} \frac{u(S)+u_{INCR}(rs')}{|S|}, & u_{INCR}(rs') > 0 \\ 0, & otherwise \end{cases}$$

*The $PEAU(S)$ is also calculated as*

$$PEAU(S) = \begin{cases} \frac{PEU_{Revised}(S) - minau \times |rs'|}{|S|}, & |rs'| \geqslant 0 \\ 0, & otherwise \end{cases}$$

*where $|rs'|$ is the maximum length of rs′ of the prefix S in D. The $PEU_{Revised}(S)$ is a revised prefix extension utility of the sequence S in D. The revised prefix extension utility of S at the extension position p:*$\langle j, j_2, \cdots, j_m \rangle$ *in Q is defined as follows:*

$$PEU_{Revised}(S, p, Q) = \begin{cases} u(S, p, Q) + u_{rs'}(S, j_m, Q), & u_{rs}(S, j_m, Q) > 0 \\ 0, & otherwise \end{cases}$$

*Then, let $p_i$ is one position of S in Q, we define the revised prefix extension utility of S in Q is $PEU_{Revised}(S, Q) = \max\{PEU_{Revised}(S, p_i, Q)\}$. And we define the revised prefix extension utility of S in D is $PEU_{Revised}(S) = \sum_{S \sqsubseteq Q \wedge Q \in D} PEU_{Revised}(S, Q)$.*

For example, consider the sequence $s_4$:$\langle \{c\}, \{b\} \rangle$, there is an instance in $Q_2$, and $PEAU$ $(s_4, \langle 3, 4 \rangle, Q_2)$ equal to 0 because $u_{rs'}(s_4, 4, Q_2) = 0$. In addition, $s_4$ has two *instances* in $Q_3$. When $minau = 20$, for all $q$-sequence $Q$ with $s_4 \sqsubseteq Q \wedge Q \in D$, the utilities of items $a$, $b$ and $d$ in *remaining sequence* of $s_4$ are 3, 12 and 8, which are lower than *minau*, respectively. Therefore, $a \notin rs'$,$b \notin rs'$ and $d \notin rs'$. We have $PEU_{Revised}(s_4, \langle 1, 2 \rangle, Q_3) = (8 + 3) + (7 + 18 + 12) = 48$ and $PEU_{Revised}(s_4, \langle 1, 3 \rangle, Q_3) = (8 + 6) + (18 + 12) = 44$, so that $PEU_{Revised}(s_4, Q_3) = \max\{PEU_{Revised}(s_4, \langle 1, 2 \rangle, Q_3), PEU_{Revised}(s_4, \langle 1, 3 \rangle, Q_3)\} = \max\{48, 44\} = 48$. Finally, the prefix extension average utility of any sequence extended by $s_4$ is $PEU_{Revised}(s_4) = PEU_{Revised}(s_4, Q_1) + PEU_{Revised}(s_4, Q_2) + PEU_{Revised}(s_4, Q_3) + PEU_{Revised}(s_4, Q_4) + PEU_{Revised}(s_4, Q_5) = 72 + 0 + 48 + 51 + 51 = 222$. The maximum length of $rs'$ is $|rs'| = \max\{4, 0, 3, 2, 4\} = 4$, so that we have $PEAU(s_4) = \frac{PEU_{Revised}(s_4) - minau \times |rs'|}{|s_4|} = \frac{222-204}{2} = 71$.

**Proof.** Let the sequence $S$ is a prefix of sequence $S'$, and the *remaining sequence* $rs = \{i_1, i_2, \cdots, i_n\}$. Set a subsequence $rs' \subseteq rs$ is $\{i'_1, i'_2, \cdots, i'_m\}$, where $1 \leqslant m \leqslant n$, and $\forall i' \in rs', u(i') \geqslant minau$.

(1) if $rs'$ is not null, then $u'_{rs} > 0$ and $|rs'| \neq 0$. Let $i_k$ is an item in $rs$, then $i_k \leqslant \max\{i'_1, i'_2, \cdots, i'_m\}$. We have

$$\begin{aligned}
au(S') &= \frac{\sum_{S \sqsubseteq Q \wedge Q \in D}(u(S,p,Q) + u(i_k, j_k, Q))}{|S|+1} \\
&\leqslant \frac{\sum_{S \sqsubseteq Q \wedge Q \in D} \max\{u(S,p,Q) + u(i_k, j_k, Q)\}}{|S|+1} \\
&\leqslant \frac{u(S) + u(i_k)}{|S|+1} \\
&= \frac{u(S) + (u_{INCR}(i_k) + minau)}{|S|+1} \\
&\leqslant \frac{u(S) + (u_{INCR}(rs') + minau)}{|S|+1} \\
&= \frac{(u_{INCR}(S) + minau \times |S|) + (u_{INCR}(rs') + minau)}{|S|+1} \\
&= \frac{u_{INCR}(S) + u_{INCR}(rs')}{|S|+1} + minau \\
&\leqslant \frac{u_{INCR}(S) + u_{INCR}(rs')}{|S|} + minau \\
&= \frac{u_{INCR}(S) + u_{INCR}(rs') + minau \times |S|}{|S|} \\
&= \frac{(u_{INCR}(S) + minau \times |S|) + u_{INCR}(rs')}{|S|} \\
&= \frac{u(S) + u_{INCR}(rs')}{|S|} \\
&= PEAU(S).
\end{aligned}$$

(2) if $rs'$ is null, but $rs$ is not null, then $u_{rs'}(S, j_m, Q) = 0$ and $|rs'| = 0$. Consider an item $i_k$ in $rs$ and $i_k < minau$, we obtain

$$\begin{aligned}
au(S') &= \frac{\sum_{S \sqsubseteq Q \wedge Q \in D}(u(S,p,Q) + u(i_k, j_k, Q))}{|S|+1} \\
&\leqslant \frac{\sum_{S \sqsubseteq Q \wedge Q \in D}\max\{u(S,p,Q) + u(i_k, j_k, Q)\}}{|S|+1} \\
&\leqslant \frac{u(S) + u(i_k)}{|S|+1} \\
&< \frac{u(S) + minau}{|S|+1} \\
&= \frac{(u(S) - minau \times |S|) + minau \times |S| + minau}{|S|+1} \\
&= \frac{u_{INCR}(S) + minau \times |S| + minau}{|S|+1} \\
&= \frac{u_{INCR}(S)}{|S|+1} + minau \\
&\leqslant \frac{u_{INCR}(S)}{|S|} + minau \\
&= \frac{u_{INCR}(S) + minau \times |S|}{|S|} \\
&= \frac{u(S)}{|S|} \\
&= PEAU(S).
\end{aligned}$$

Therefore, we obtain $PEAU(S) \geqslant au(S')$. □

Based on the above derivation, the average utility of any sequence extended from $S$ is less than $PEAU(S)$. Then, the $PEAU(S)$ is the maximum average-utility upper bound of $S$ and its descendant. Thus, if the value of $PEAU(S)$ is less than $minau$, any descendants of $S$ will not be the high average-utility sequential pattern, and these descendants could be safely pruned.

**Definition 13 (*RSAU* Upper Bound).** *Let $PEAU(S)$ be the prefix extension average utility of $S$ in database $D$. Assume that a sequence $S'$ is extended from $S$ through one I-Extension or S-Extension operation, i.e., consider the node representing $S$ in LQS-tree, the node representing $S'$ is its child node. The reduced sequence average utility of $S'$ in q-sequence database $D$, denoted as $RSAU(S')$, is defined as*

$$RSAU(S') = \begin{cases} PEAU(S), & S \sqsubseteq Q \wedge S' \sqsubseteq Q \wedge Q \in D \\ 0, & otherwise \end{cases}$$

**Proof.** Assume the sequence $S'$ is a prefix of $S''$ or $S'' = S'$, and the sequence $S'$ could be extended from sequence $S$ through one "*extension*" operation, such that the sequence $S$ is also a prefix of $S''$.

Based on the Definition 11, we obtain

$$au(S'') \geqslant PEAU(S).$$

According to the Definition 12, we have

$$RSAU(S') = PEAU(S),$$

such that the average of the sequence is

$$au(S'') \geqslant RSAU(S').$$

□

Thus, if the value of $RSAU(S')$ is less than $minau$, any descendants of $S$ are not the high average-utility sequential pattern, and these descendants could also be safely pruned. But RSAU still overestimates the upper bound. This upper bound is similar to RSU in HUS-Span, and more details can be found in Ref. [46].

## 4.4. Proposed TKAUS Algorithm

By adopting the aforementioned mechanism and strategy, the proposed TKAUS is described as follows. The main algorithm is represented as Algorithm 1. Algorithm 2 represents a recursive procedure, and Part 3 of the whole algorithm procedure represents the updating of the set of HAUSPs and the threshold.

In the beginning, a given quantitative database is read, and the number of desired HAUSPs is specified. The algorithm utilizes the variable $minau$ to keep the current minimum average-utility threshold (Line 1) and engages the $TKList$, which is a sorted list of size $k$ to maintain the top-$k$ HAUSPs (Line 2). From lines 3 to 7, scan the original database once, calculate the utility value of each item (1-sequences), and construct the projected databases of all items. In line 8 and line 9, following the SAUR strategy, the algorithm increases $minau$ to the $k$th highest average-utility value. From lines 10 to 14, for each item whose utility is greater than $minau$, update to the $TKList$. Then, adopting the upper bound, for each *promising* item, the *projection*-TKAUS algorithm is called recursively (Lines 15 to 19). When the TKAUS algorithm is over, the $TKList$ is returned, which stores all the HAUSPs.

**Algorithm 1** The TKAUS algorithm

```
Input:  A quantitative database, $D$.
        The number of desired HAUSPs, $k$.
Output: Top-$k$ HAUSPs in $D$.
 1: initializing $minau = 0$
 2: initializing $TKList = \varnothing$
 3: //First scanning (for the original database)
 4: for each sequence $Q$ in $D$
 5:     (1) calculating and storing the average utility of all 1-sequences
 6:     (2) constructing projected databases of all 1-sequences
 7: end for
 8: sorting the average-utility values of all 1-sequences in descending order, then getting
    the $k$th highest average-utility value $minau_0$
 9: $minau = minau_0$
10: for each sequence $S \in$ 1-sequences
11:     if $S.au \geqslant minau$
12:         update $TKList$
13:     end if
14: end for
15: for each sequence $S \in$ 1-sequences
16:     if $PEAU(S) \geqslant minau$
17:         calling projection-TKAUS $(S, D_S)$
18:     end if
19: end for
20: return $TKList$
```

**Algorithm 2** *projection*-TKAUS algorithm

```
Input: A sequence as prefix, $S$.
    The projected database of $S$, $D_S$.
Output: Top-$k$ HAUSPs in $D$.
1:  scanning $D_S$ once to:
2:  (1) adding $I$-Extension items of $S$ to $ilist$
3:  (2) adding $S$-Extension items of $S$ to $slist$
4:  for each item $i \in ilist$
5:      $S' \leftarrow \langle S \oplus i \rangle$
6:      if $RSAU(S') < minau$
7:          removing $i$ from $ilist$ (Pruning the unpromising item in $ilist$)
8:      end if
9:      constructing projection database of $S'$,$D_{S'}$
10:     if $S'.au \geqslant minau$
11:         calling $update(S', minau, TKList)$
12:         putting $S'$ to $seqlist$
13:     end if
14: end for
15: for each item $i \in slist$
16:     $S' \leftarrow \langle S \otimes i \rangle$
17:     if $RSAU(S') < minau$
18:         removing $i$ from $slist$ (Pruning the $unpromising$ item in $slist$)
19:     end if
20:     constructing projection database of $S'$, $D_{S'}$
21:     if $S'.au \geqslant minau$
22:         calling $update(S', minau, TKList)$
23:         putting $S'$ to $seqlist$
24:     end if
25: end for
26: for each sequence $S' \in seqlist$
27:     if $PEAU(S') \geqslant minau$
28:         Calling $Projection$-TKAUS$(S', D_{S'})$
29:     end if
30: end for
31: return $TKList$
```

As shown in Algorithm 2, in line 1, the *projection*-TKAUS algorithm identifies items that can be extended by scanning the projected database. Each input sequence is regarded as a prefix of a pattern. From lines 4 to 25, for each extended sequence, if the RSAU value is not less than the threshold, the extended sequence could be a top-*k* HAUSP or its prefix. After that "*extension*" operation, we compare all generated sequences in seqlist to the new minimum average-utility threshold according to their PEAU values. Then, the *projection*-TKAUS algorithm is called recursively. Algorithm 3 is $update(S', minau, TKList)$ algorithm. The lowest average-utility sequence is removed from $TKList$ when the size exceeds the specified parameter. Meanwhile, the *minau* is updated to the sequence average utility of *k*th HAUSP in $TKList$.

**Algorithm 3** $update(S', minau, TKList)$ algorithm

```
Input: A generated sequence from $S$, $S'$.
    The number of desired HAUSPs, $k$.
Output: A minimum average-utility threshold, $minau$.
    A set of top-$k$ high average-utility sequential patterns, $TKList$.
1: scanning $TKList$ once to:
2: (1) sorting all sequences by the sequence average utility in descending order, then
   getting the sequence with the lowest average-utility value $lowest.au.seq$
3: (2) getting the number of sequential patterns in $TKList$, $k'$
4: if $k' \geqslant k$
5:     Inserting $S'$ into $TKList$
6:     $minau$ = the average utility of $k^{th}$ HAUSP in $TKList$
7:     Removing $lowest.au.seq$ from $TKList$
8: else
9:     Inserting $S'$ into $TKList$
10: end if
11: return $minau$,$TKList$
```

### 4.5. Complexity Analysis of the Proposed TKAUS Algorithm

Suppose the number of quantitative sequences in a quantitative sequential database is $|D|$, $|Q|$ is the average number of items of the quantitative sequence in the database, and $|I|$ is the number of distinct items in the database. $k$ represents the desired number of HAUSPs. It is a user-specified parameter. Therefore, in the first scan of the original database, $\mathcal{O}(|D| \times |Q|)$ $q$-items are processed to calculate the information of each one. This information is stored in a projected database, and $\mathcal{O}(|D| \times |Q|)$ is also the memory complexity of the first scan. Then, in TKAUS, the $k^{th}$ highest average-utility value of all 1-sequences is set as the initial threshold, so it takes $\mathcal{O}(|I|\log|I|)$ in order to sort all 1-sequences in an average utility descending order using the quick sorting algorithm. In the sorting step, the memory complexity is $\mathcal{O}(\log|I|)$. TKAUS next updates $TKList$, which is used to store HAUSPs. At this time, in the worst case, it takes $\mathcal{O}(|D| \times |Q|)$ to calculate the actual sequence average utility for each sequence. Meanwhile, TKAUS calls the recursive function *projection*-TKAUS. The required processing for comparing the threshold and the PEAU of each sequence is $\mathcal{O}(|D| \times |Q|)$. In *projection*-TKAUS, the $(k+1)$-sequence is generated from the $k$-sequence by appending each item. In addition, two conditional lists are constructed to store the appending items. It takes $\mathcal{O}(|D| \times |Q|)$ to scan the projected database and to mark the unpromising items with low RSAU. Then, the unpromising items will be deleted from the lists. The largest size of a conditional list is $|I|$. The algorithm next traverses the reduced lists and appends each item to the prefix. Meanwhile, TKAUS finds the HAUSPs and calls the $update$ function. The required amount of processing to calculate the actual sequence average utility is $\mathcal{O}(|D| \times |Q|)$. The generated sequence with high PEAU is set as a new prefix and is inserted in the projected database. Repeat the above process for all the prefixes. The corresponding time complexity is $\mathcal{O}(|D||Q|) + \mathcal{O}(|D||Q|) + \mathcal{O}(|D||Q|)$, which equals $\mathcal{O}(|D||Q|)$. Its memory complexity is $\mathcal{O}(|D||Q|) + \mathcal{O}(|D||Q|) + \mathcal{O}(|D||Q|) + \mathcal{O}(|I|) + \mathcal{O}(|I|)$, which equals $\mathcal{O}(|D||Q| + |I|)$. As for the $update$ function, the generated sequence is inserted into $TKList$, and the sequence with the lowest average utility is deleted. Its time complexity of the process is $\mathcal{O}(k\log k)$. In addition, the memory complexity of the update function is $\mathcal{O}(\log k)$.

Suppose $L$ is the sequence length of the longest generated one in the above process. Then, $|Q| \leqslant L$. Therefore, the maximum number of recursive calls is $|I|^L$ in the function *projection*-TKAUS. According to the above, the time complexity and memory complexity of TKAUS are $\mathcal{O}(|D||Q| + |I|\log|I|) + |I|^L \times \mathcal{O}(|D||Q|) + |I|^L \times |I| \times \mathcal{O}(k\log k)$ and $\mathcal{O}(|D||Q| + \log|I|) + L \times \mathcal{O}(|D||Q| + |I|) + L \times |I| \times \mathcal{O}(logk)$, respectively. In the worst case where $|Q| = L$, the time complexity and memory complexity are $\mathcal{O}(|I|^L|D|L + |I|\log|I| + |I|^(L+1)k\log k)$ and $\mathcal{O}(L^2|D| + \log|I| + L|I| + L|I|\log k)$.

## 5. Experiments

In this section, we conducted experiments to compare and contrast the performance of the proposed TKAUS algorithm. The algorithms are implemented in Java and developed in Eclipse IDE, and the experiments are conducted on a cloud virtual machine equipped with an Intel(R) Xeon(R) Platinum 8255c CPU and Linux version 3.10.0-1160.31.1.el7.x86_64 operating system.

### *5.1. Experimental Settings*

To evaluate performance, the experiments are conducted on seven datasets with different characteristics, which are obtained from SPMF (http://www.philippe-fournier-viger.com/spmf/), accessed on 20 December 2022. Currently, there is a scarcity of existing works focusing on top-*k* HAUSPM. For the performance tests, we assess the execution time, space overhead, the total number of candidates generated, and scalability toward large datasets by comparing with the TKUS algorithm [48]. The details of the experiments are given as follows.

The Bible is a conversion dataset from part of the Bible, and each word is considered to be an item. Each sentence is considered to be a sequence, and the lengths of sequences are mainly medium in the Bible. The Leviathan is also a conversion dataset from the novel Leviathan by Thomas Hobbes (1651). It is similar to the previous one. The special word in the novel was replaced by the digital item. The Sign is a new format of the sign language utterance dataset, and the original one was created by the National Center for Sign Language and Gesture Resources at Boston University. There are approximately 730 sequences, and most of them are long. The Kosarak is a clickstream dataset from a Hungarian online news portal. It is difficult to deal with the extremely high sequence length in this dataset. Yoochoose is also clickstream data from e-commerce in a new format. It is composed of a single-itemset or multi-itemset sequence and has been used in Ref. [48].

For all datasets, five parameters are used for the feature description. $|D|$ indicates the number of $q$-sequences in dataset. $|I|$ indicates the number of distinct items in a dataset. $AvgLen$ and $MaxLen$ are the average length and the maximal length of $q$-sequence in dataset, respectively. The average number of $q$-itemsets in one $q$-sequence is denoted as $AvgSeqSize$. The average number of $q$-items in one $q$-itemset is denoted as $AvgSetSize$. The details of datasets are given in Table 3.

**Table 3.** Features of datasets.

| Dataset | $\|D\|$ | $\|I\|$ | $AvgLen$ | $MaxLen$ | $AvgSeqSize$ | $AvgSetSize$ |
|---|---|---|---|---|---|---|
| Bible | 36,369 | 13,905 | 21.64 | 100 | 17.85 | 1.0 |
| Leviathan | 5834 | 9025 | 33.81 | 100 | 26.34 | 1.0 |
| Sign | 730 | 267 | 52 | 94 | 51.99 | 1.0 |
| Yoochoose | 234,300 | 16,004 | 2.25 | 112 | 1.14 | 1.97 |
| Kosarak_10K | 10,000 | 10,094 | 8.14 | 608 | 8.14 | 1.0 |
| SynDataset_80K | 79,718 | 7584 | 6.19 | 18 | 26.69 | 4.32 |
| Scalability_10K | 10,000 | 7313 | 6.22 | 18 | 6.22 | 4.35 |

### *5.2. Strategy Performance and Effectiveness Analysis*

In this subsection, the experimental results will be used to assess the algorithm's performance. The top-*k* EHAUSM was selected as a comparison based on EHAUSM, which is the first approach for HAUSPM in general case [35]. However, it is not clear whether the performance of the EHAUSM proposed by Tin et al. [35] is better than the one for this experiment. They mentioned and demonstrated $HAUSPs \subseteq HUSPs$ by comparing HAUSM to HUSM in the experiment. We also evaluate the proposed algorithm TKAUS against TKUS because these new upper bounds correspond to efficient pruning strategies inspired by TKUS. They have examined the impact of each pruning strategy by comparing the distinct variants of the algorithm. The results and more details can be found in Ref. [48]. In the TKAUS algorithm, we take a similar idea. Moreover, we modify and redefine the upper bound and pruning strategies to define a new problem for TKAUSM.

Combined with the previous section and the comparative results here, the proposed upper bounds PEAU and RSAU benefit the algorithm to obtain better performance. Meanwhile, considering the existing research on HAUIM and the characteristics of "*extension*" operation, we adopt a new threshold to stop TKSUA scanning *unpromising* node in the LQS-tree. As we expected, the results in the same scenarios are listed in Tables 4 and 5, which proves that, in most cases, the combination pruning strategy plays a more effective role in reducing the generated candidates and the running time to achieve an improvement in performance.

**Table 4.** Effectiveness of Top-*k*EHAUSM and TKAUS.

| **Dataset** | **Time** | | **Max Memory** | | **Candidates** | |
|---|---|---|---|---|---|---|
| | **Top-*k*EHAUSM** | **TKAUS** | **Top-*k*EHAUSM** | **TKAUS** | **Top-*k*EHAUSM** | **TKAUS** |
| Bible k = 200 | 1621.854 | 22.548 | 410.481 | 323.106 | 15,724 | 6143 |
| Leviathan k = 500 | 1490.095 | 16.152 | 351.635 | 319.743 | 73,712 | 40,918 |
| Sign k = 500 | 3420.941 | 79.63 | 353.422 | 312.315 | 649,694,319 | 990,105 |
| Yoochoose k = 8000 | 13,127.643 | 2429.662 | 777.194 | 397.227 | 29,703,530,233 | 37,185,721 |
| Kosarak_10K k = 9 | 1.79 | 1.251 | 102.335 | 39.050 | 54 | 348 |
| SynDataset_80K k = 200 | 1177.576 | 65.639 | 480.374 | 373.288 | 2574 | 5630 |

**Table 5.** Effectiveness of TKAUS and TKUS.

| **Dataset** | **Time** | | **Max Memory** | | **Candidates** | |
|---|---|---|---|---|---|---|
| | **TKAUS** | **TKUS** | **TKAUS** | **TKUS** | **TKAUS** | **TKUS** |
| Bible k = 200 | 22.548 | 888 | 323.106 | 4247 | 6143 | 17,059 |
| Leviathan k = 500 | 16.152 | 339 | 319.743 | 2893 | 40,918 | 40,918 |
| Sign k = 500 | 79.63 | 193 | 312.315 | 2682 | 990,105 | 193,477 |
| Yoochoose k = 8000 | 2429.662 | 35 | 397.227 | 2780 | 37,185,721 | 180,641 |
| Kosarak_10K k = 9 | 1.251 | 13 | 39.050 | 2279 | 348 | 2001 |
| SynDataset_80K k = 200 | 65.639 | 839 | 373.288 | 5483 | 5630 | 972,569 |

### *5.3. Speed Performance and Efficiency Analysis*

The execution time consists of two components: the processing time utilized by the CPU and the access time necessary for input/output operations. There is a longer execution time for multiple database scanning and candidate processing. The comparative results showed in Ref. [48] demonstrated that both the projection mechanism and effective pruning strategies are of great significance in improving the performance of the mining algorithm. Figure 2 shows the execution times variation of the TKAUS algorithm with different values of parameter $k$ on each dataset. With the increase in parameter $k$, the growth of the threshold is slowing, and more candidates are promising, and, in consequence, the execution time is bound to increase substantially. The mining process for HAUSP can be accelerated by the strategy with a tighter upper bound. However, for each prefix, top-$k$ EHAUSM always requires more time to find the highest utility item in *remaining sequence* than TKAUS. The utility raising strategy is effective at reducing the search space fast in the top-$k$ HUPM. Please note that when $k \geqslant 1000$ on the Sign dataset, the runtime consumption of top-$k$ EHAUSM exceeds 27 h. It is far beyond what we think possible. The proposed algorithm will take more time than TKUS with an increase in parameter $k$ in Figure 2c,d. But apart from that, for HAUSPM, when the desired sequence average utility is a fixed value, the sequence utility also keeps an extra steady rise with its length increasing. Hence, the TKAUS will have better performance than TKUS, too.

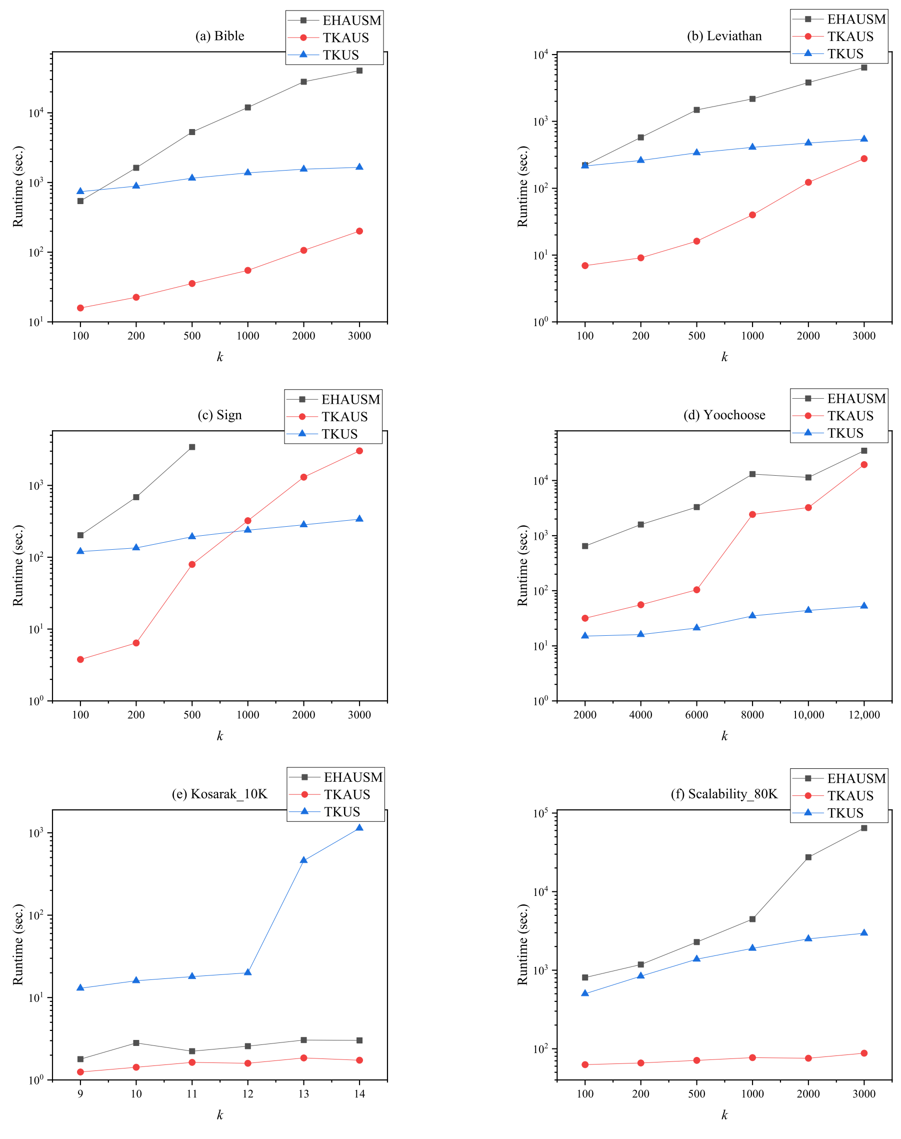


**Figure 2.** Runtime for various *k* in different dataset.

## 5.4. Number of Candidates

The number of candidates could be regarded as a crucial measure of the search space. The larger dataset generates more candidates from a macro perspective. In fact, TKUS and TKAUS generate fewer candidates than other early algorithms by adopting the projection mechanism. Zhang et al. in [48] defined search space shrinkage rate (SSSR) for evaluation and validated that TKUS generates fewer candidates than TKHUS-Span in the experiment.

Moreover, among the different datasets, the longer the average length of the sequence is, the greater the number of candidates is. The algorithms also generate more candidates as the number of sequences increases. With increasing parameter *k*, the threshold for mining decreases, and the quantity of candidates increases.

The upper bound proposed in EHAUSM overestimates the candidate sequence utility based on the maximum item utility in the *remaining sequence* or the utility of the whole sequence so that it is a larger value far beyond the actual utility of the candidate sequence. Finally, more candidates were generated by the "*extension*" operation. But in Figure 3e, the number of candidates in top-*k* EHAUSM is indeed less than in TKAUS. If the sequences in the database vary considerably in length, and most of the item is interesting to the user and has a high utility value, the TKAUS may lose its advantages.

The main objective of TKUS is to find top-*k* HUSPs, but the purpose of the proposed TKAUS is to identify top-*k* HAUSPs. The average utility of pattern not only depends on its utility but also involves its length. Then, patterns with low utility also have great potential to be high average-utility ones, which have a corresponding short length, and in Figure 3, it is reasonable that the number of candidates generated in TKAUS is less than in TKUS, especially in Figure 3f. Compared with the feature of the dataset, this could be a reason that the value of item utility in the dataset is relatively concentrated, i.e., the utility of item is the same level, and the length of pattern has a great effect on the sum of utility than the average utility of items in a pattern.

Moreover, note that when $k = 500$ on the Sign dataset, the search space of the top-*k* EHAUSM algorithm is far beyond the other two algorithms. Therefore, we do not show more experiments. The proposed algorithm will generate more candidates than TKUS with increasing parameter *k* in Figure 2c,d. The average length of the sequence and the maximal length of the sequence have great effects on HAUSPM. In addition, that is why the results of runtime have similar trends.

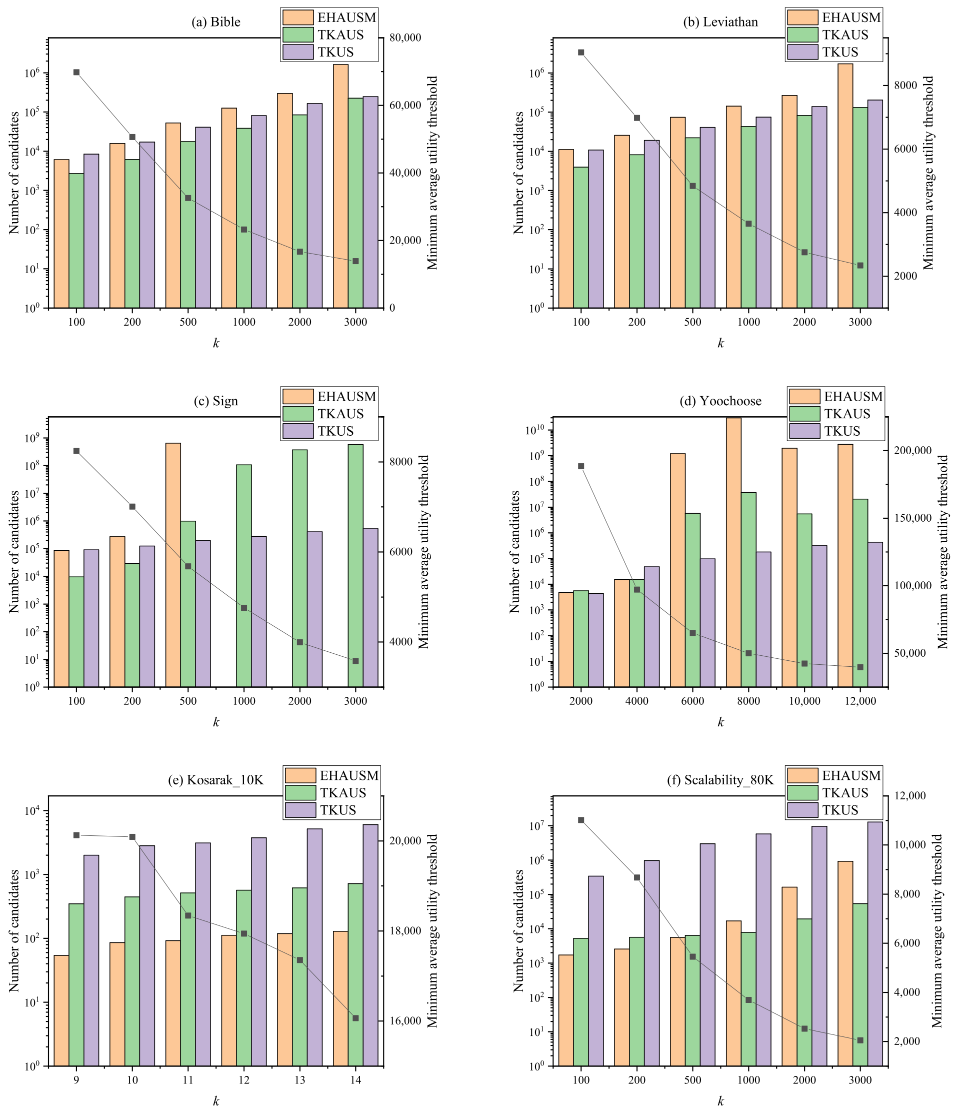


**Figure 3.** Number of candidates and threshold of top-*k* HAUSPs in different datasets.

## 5.5. Memory Overhead Evaluation

We can see from Figure 4 that TKAUS generally outperforms top-*k* EHAUSM and TKUS in terms of memory usage. Figure 3d,e clearly illustrates that both EHAUSM and

TKUS have more candidates than top-*k* EHAUSM in some cases. Although the gap is still obvious, the process of sorting brings more memory consumption in top-*k* EHAUSM. The more candidates are generated, the more sorting operations are needed. In TKUS, the threshold is the sum of utilities. It is harder to avoid generating the candidate, which consists of some low-utility items. This is different from top-*k* EHAUSM or TKAUS, and it will incur a high memory usage. Because of that, both EHAUSM and TKUS have good performance by adopting efficient strategies, but TKAUS has greater improvement. Likewise, the memory consumption becomes larger when the parameter *k* increases.

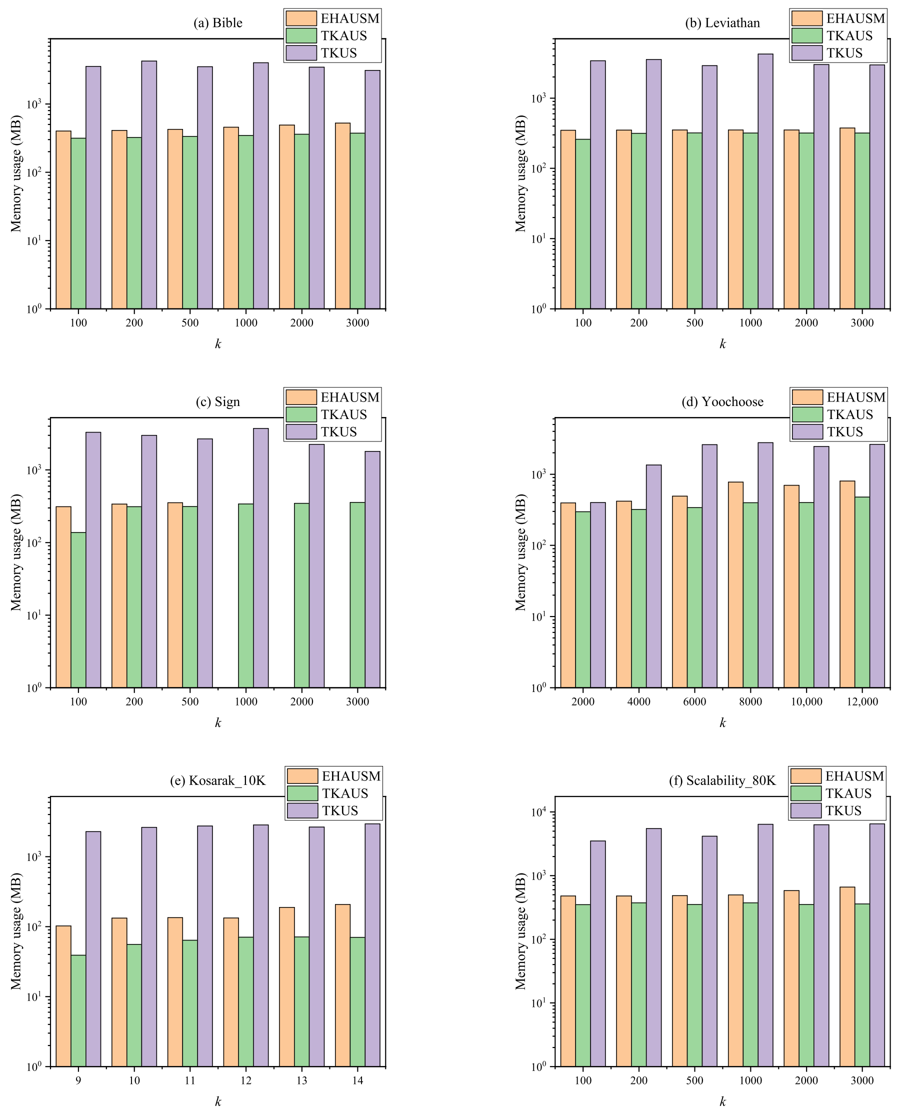


**Figure 4.** Memory usage for various *k* in different dataset.

## 5.6. Scalability Evaluation

The scalability of TKAUS is evaluated in this part. The series of datasets with varying sizes is built by combining the duplication of dataset Scalability_10k repeatedly. The experimental results, including execution time, memory consumption, and the number of candidates with $k = 500$, are depicted in Figure 5. The runtime increases linearly as the database size grows. The increase in memory consumption is likewise linear. The number of candidates keeps a constant value. It means that variations in dataset size did not cause significant volatility or anomalies in algorithm efficiency based on the experimental results. This supports the idea that the proposed method can provide relatively stable performance when processing datasets of diverse magnitudes. According to the experiments, the proposed TKAUS shows a better scalability performance.

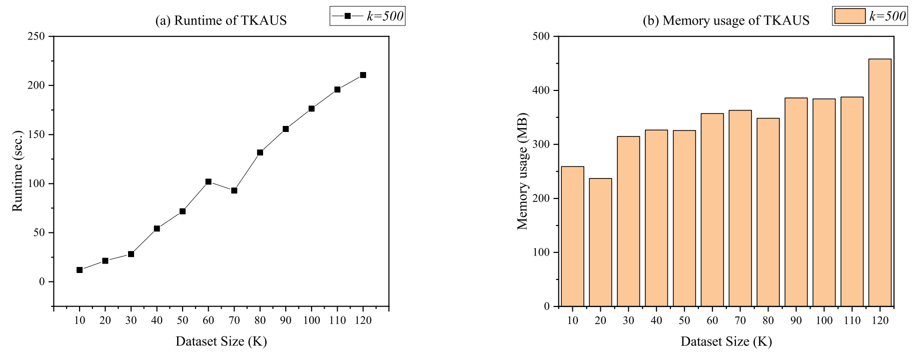


**Figure 5.** Scalability of TKAUS in Scalability_10k when $k = 500$.

### *5.7. Limitations*

The experimental results presented in the preceding sections highlight the varying degrees of efficiency advantages exhibited by the proposed algorithm across datasets with diverse features. Although both the proposed method and the compared algorithms are capable of handling datasets of the same type, the efficiency advantages may differ when applied to datasets with different characteristics, e.g., the dataset Sign and Yoochoose. The efficiency of the algorithm is significantly influenced by the specific characteristics of the datasets. Although the experimental section includes discussions of six commonly used datasets with diverse features, it is important to acknowledge that the quantity and characteristics of the experimental datasets do not encompass the entire spectrum. Furthermore, the experimentation parameter $k$ has limited values, allowing for only a partial representation of trends. The proposed algorithm is motivated by fundamental aspects of the research problem and identified optimization potential in existing algorithms. However, it demonstrates notable advantages in terms of algorithm efficiency. The computation of average-utility upper bounds remains somewhat complex. This complexity leaves room for further research to propose even more superior algorithms in this regard. In addition, as research on SPM and the transformation of typified resources within DIKWP structures progresses, it is expected that additional relevant factors will be identified and considered. This will necessitate further studies to enhance and potentially adjust existing algorithms accordingly.

## 6. Conclusions

In this paper, the problem of top-$k$ HAUSPM is formulated, and a novel algorithm called TKAUS is proposed to handle information resource transformation in DIKWP graphs. The projection mechanism, two new upper bounds and their corresponding strategies are adopted in the proposed algorithm. As the quick minimum utility threshold increases, the extraction efficiency for top-$k$ HAUSPs also improves. By the "*extension*" operation, there is a special correlation between the average utility of pattern and the utility of the length of the pattern. A novel measure tailored for the remaining sequences in HAUSPM is designed. Leveraging this measure, an improved upper bound is designed for HAUSPM that avoids the multiple sorting of items, which is a limitation of existing approaches. Corresponding pruning strategies and proofs are provided as well. Experimental results demonstrate that the proposed methods deliver stable advantages. Runtime improvements of an order of magnitude or more are achieved compared to alternative algorithms, with this advantage expected to grow more pronounced as the number of patterns to be mined increases. However, the performance of the algorithm is still dependent on dataset characteristics and the parameter $k$, which determines the desired number of patterns. It is important to note that the advantages of the algorithm may be limited when dealing with very small values of $k$ or high thresholds. Additionally, results indicate that the proposed algorithm is particularly well suited for datasets involving longer sequences. Our future work involves designing more efficient approaches for resource transformation in DIKWP graphs.

**Author Contributions:** Writing—original draft preparation, K.C.; writing—review and editing, Y.D. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was funded by Hainan Province Key R&D Program (No. ZDYF2022GXJS007, ZDYF2022GXJS010), Hainan Province Higher Education and Teaching Reform Research Project (No. Hnjg2021ZD-3), Natural Science Foundation of Hainan Province (No. 620RC561) and Hainan Province Key Laboratory of Meteorological Disaster Prevention and Mitigation in the South China Sea (No. SCSF202210).

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data are contained within the article.

**Conflicts of Interest:** The authors declare no conflict of interest.

## References

1. Li, Y.; Li, Z. Duan, Y.; Spulber, A.-B. Physical artificial intelligence (PAI): The next-generation artificial intelligence. *Front. Inf. Technol. Electron. Eng.* **2023**, *24*, 1231–1238. [CrossRef]
2. Duan, Y.; Shao, L.; Hu, G. Specifying Knowledge Graph with Data Graph, Information Graph, Knowledge Graph, and Wisdom Graph. *Int. J. Softw. Innov.* **2018**, *6*, 10–25. [CrossRef]
3. Duan, Y.; Sun, X.; Che, H.; Cao, C.; Li, Z.; Yang, X. Modeling Data, Information and Knowledge for Security Protection of Hybrid IoT and Edge Resources. *IEEE Access* **2019**, *7*, 99161–99176. [CrossRef]
4. Duan, Y. Existence Computation: Revelation on Entity vs. Relationship for Relationship Defined Everything of Semantics. In Proceedings of the 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2019, Toyama, Japan, 8–11 July 2019; Nakamura, M., Hirata, H., Ito, T., Otsuka, T., Okuhara, S., Eds.; IEEE Computer Society: Washington, DC, USA, 2019; pp. 139–144. [CrossRef]
5. Duan, Y. Applications of Relationship Defined Everything of Semantics on Existence Computation. In Proceedings of the 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2019, Toyama, Japan, 8–11 July 2019; Nakamura, M., Hirata, H., Ito, T., Otsuka, T., Okuhara, S., Eds.; IEEE Computer Society: Washington, DC, USA, 2019; pp. 184–189. [CrossRef]
6. Duan, Y. Towards a Periodic Table of conceptualization and formalization on State, Style, Structure, Pattern, Framework, Architecture, Service and so on. In Proceedings of the 20th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing, SNPD 2019, Toyama, Japan, 8–11 July 2019; Nakamura, M., Hirata, H., Ito, T., Otsuka, T., Okuhara, S., Eds.; IEEE Computer Society: Washington, DC, USA, 2019; pp. 133–138. [CrossRef]
7. Gao, H.; Duan, Y.; Shao, L.; Sun, X. Transformation-based processing of typed resources for multimedia sources in the IoT environment. *Wirel. Netw.* **2021**, *27*, 3377–3393. [CrossRef]
8. Duan, Y.; Pham, V.T.; Song, M.; Nguyen, H.D. Ultimate of Digital Economy: From Asymmetric Data Economy to Symmetric Knowledge and Wisdom Economy. In *New Trends in Intelligent Software Methodologies, Tools and Techniques, Proceedings of the 22nd International Conference on New Trends in Intelligent Software Methodologies, Tools and Techniques SoMeT2023, Naples, Italy, 20–22 September 2023*; Fujita, H., Guizzi, G., Eds.; IOS Press: Amsterdam, The Netherlands, 2023; Volume 371, pp. 85–96. [CrossRef]
9. Duan, Y. DIKWP Artificial Consciousness Hypothesis, Nature and Principles (Empirical Description). 2023. Available online: https://www.researchgate.net/publication/372140686_DIKWPrengongyishijiashebenzhiyuyuanlijingyanxingmiaoshu?channel=doi&linkId=64a68bc2c41fb852dd556bf9&showFulltext=true (accessed on 31 July 2023).
10. Fournier-Viger, P.; Gueniche, T.; Tseng, V.S. Using Partially-Ordered Sequential Rules to Generate More Accurate Sequence Prediction. In Proceedings of the Advanced Data Mining and Applications, 8th International Conference, ADMA 2012, Nanjing, China, 15–18 December 2012; Zhou, S., Zhang, S., Karypis, G., Eds.; Proceedings; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2012; Volume 7713, pp. 431–442. [CrossRef]
11. Srikant, R.; Agrawal, R. Mining Sequential Patterns: Generalizations and Performance Improvements. In Proceedings of the 5th International Conference on Extending Database Technology: Advances in Database Technology, Avignon, France, 25–29 March 1996; Proceedings; Springer: Berlin/Heidelberg, Germany, 1996; EDBT'96; pp. 3–17.
12. Gan, W.; Lin, J.C.W.; Fournier-Viger, P.; Chao, H.C.; Tseng, V.S.; Yu, P.S. A Survey of Utility-Oriented Pattern Mining. *IEEE Trans. Knowl. Data Eng.* **2021**, *33*, 1306–1327. [CrossRef]
13. Nguyen, L.T.; Vu, V.V.; Lam, M.T.; Duong, T.T.; Manh, L.T.; Nguyen, T.T.; Vo, B.; Fujita, H. An efficient method for mining high utility closed itemsets. *Inf. Sci.* **2019**, *495*, 78–99. [CrossRef]

14. Liu, J.; Zhang, X.; Fung, B.C.; Li, J.; Iqbal, F. Opportunistic mining of top-n high utility patterns. *Inf. Sci.* **2018**, *441*, 171–186. [CrossRef]
15. Yao, H.; Hamilton, H.J.; Butz, C.J. A Foundational Approach to Mining Itemset Utilities from Databases. In Proceedings of the Fourth SIAM International Conference on Data Mining, Lake Buena Vista, FL, USA, 22–24 April 2004; Berry, M.W., Dayal, U., Kamath, C., Skillicorn, D.B., Eds.; SIAM: University City, PA, USA, 2004; pp. 482–486. [CrossRef]
16. Gan, W.; Lin, J.C.W.; Fournier-Viger, P.; Chao, H.C.; Hong, T.P.; Fujita, H. A Survey of Incremental High-Utility Itemset Mining. *Wiley Interdiscip. Rev. Data Min. Knowl. Discov.* **2018**, *8*, e1242. [CrossRef]
17. Yun, U.; Nam, H.; Kim, J.; Kim, H.; Baek, Y.; Lee, J.; Yoon, E.; Truong, T.; Vo, B.; Pedrycz, W. Efficient transaction deleting approach of pre-large based high utility pattern mining in dynamic databases. *Future Gener. Comput. Syst.* **2020**, *103*, 58–78. [CrossRef]
18. Ryang, H.; Yun, U. Indexed List-Based High Utility Pattern Mining with Utility Upper-Bound Reduction and Pattern Combination Techniques. *Knowl. Inf. Syst.* **2017**, *51*, 627–659. [CrossRef]
19. Zhang, C.; Zu, Y.; Nie, J.; Du, L. Two Efficient Algorithms for Mining High Utility Sequential Patterns. In Proceedings of the 2019 IEEE Intl Conf on Parallel and Distributed Processing with Applications, Big Data and Cloud Computing, Sustainable Computing and Communications, Social Computing and Networking, ISPA/BDCloud/SocialCom/SustainCom 2019, Xiamen, China, 16–18 December 2019; IEEE: Piscataway, NJ, USA, 2019; pp. 905–911. [CrossRef]
20. Yin, J.; Zheng, Z.; Cao, L. USpan: An Efficient Algorithm for Mining High Utility Sequential Patterns. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 12 August 2012; KDD'12; pp. 660–668. [CrossRef]
21. Shie, B.; Hsiao, H.; Tseng, V.S.; Yu, P.S. Mining High Utility Mobile Sequential Patterns in Mobile Commerce Environments. In Proceedings of the Database Systems for Advanced Applications—16th International Conference, DASFAA 2011, Hong Kong, China, 22–25 April 2011; Proceedings, Part I; Lecture Notes in Computer Science; Yu, J.X., Kim, M., Unland, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; Volume 6587, pp. 224–238. [CrossRef]
22. Zhang, C.; Zu, Y. An Efficient Parallel High Utility Sequential Pattern Mining Algorithm. In Proceedings of the 21st IEEE International Conference on High Performance Computing and Communications; 17th IEEE International Conference on Smart City; 5th IEEE International Conference on Data Science and Systems, HPCC/SmartCity/DSS 2019, Zhangjiajie, China, 10–12 August 2019; Xiao, Z., Yang, L.T., Balaji, P., Li, T., Li, K., Zomaya, A.Y., Eds.; IEEE: Piscataway, NJ, USA, 2019; pp. 2798–2803. [CrossRef]
23. Gan, W.; Lin, J.C.W.; Zhang, J.; Chao, H.C.; Fujita, H.; Yu, P.S. ProUM: Projection-based utility mining on sequence data. *Inf. Sci.* **2020**, *513*, 222–240. [CrossRef]
24. Gan, W.; Lin, J.C.W.; Zhang, J.; Fournier-Viger, P.; Chao, H.C.; Yu, P.S. Fast Utility Mining on Sequence Data. *IEEE Trans. Cybern.* **2021**, *51*, 487–500. [CrossRef] [PubMed]
25. Lee, C.; Ryu, T.; Kim, H.; Kim, H.; Vo, B.; Lin, J.C.W.; Yun, U. Efficient approach of sliding window-based high average-utility pattern mining with list structures. *Knowl.-Based Syst.* **2022**, *256*, 109702. [CrossRef]
26. Hong, T.; Lee, C.; Wang, S. Mining High Average-Utility Itemsets. In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, San Antonio, TX, USA, 11–14 October 2009; IEEE: Piscataway, NJ, USA, 2009; pp. 2526–2530. [CrossRef]
27. Lan, G.C.; Hong, T.P.; Tseng, V.S. Efficiently Mining High Average-Utility Itemsets with an Improved Upper-Bound Strategy. *Int. J. Inf. Technol. Decis. Mak.* **2012**, *11*, 1009–1030. [CrossRef]
28. Lan, G.; Hong, T.; Tseng, V.S. A Projection-Based Approach for Discovering High Average-Utility Itemsets. *J. Inf. Sci. Eng.* **2012**, *28*, 193–209.
29. Kim, D.; Yun, U. Efficient Algorithm for Mining High Average-Utility Itemsets in Incremental Transaction Databases. *Appl. Intell.* **2017**, *47*, 114–131. [CrossRef]
30. Yin, J.; Zheng, Z.; Cao, L.; Song, Y.; Wei, W. Efficiently Mining Top-K High Utility Sequential Patterns. In Proceedings of the 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, 7–10 December 2013; pp. 1259–1264. [CrossRef]
31. Wu, C.W.; Shie, B.E.; Tseng, V.S.; Yu, P.S. Mining Top-K High Utility Itemsets. In Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'12, Beijing, China, 12–16 August 2012; ACM: New York, NY, USA, 2012; pp. 78–86. [CrossRef]
32. Tzvetkov, P.; Yan, X.; Han, J. TSP: Mining Top-K Closed Sequential Patterns. In Proceedings of the Third IEEE International Conference on Data Mining, ICDM'03, Melbourne, FL, USA, 22 November 2003; IEEE Computer Society: Washington, DC, USA, 2003; pp. 347–354.
33. Wu, R.; He, Z. Top-k High Average-Utility Itemsets Mining with Effective Pruning Strategies. *Appl. Intell.* **2018**, *48*, 3429–3445. [CrossRef]
34. Thilagu, M.; Nadarajan, R. Efficiently Mining of Effective Web Traversal Patterns with Average Utility. *Procedia Technol.* **2012**, *6*, 444–451. [CrossRef]
35. Truong, T.; Duong, H.; Le, B.; Fournier-Viger, P. EHAUSM: An efficient algorithm for high average utility sequence mining. *Inf. Sci.* **2020**, *515*, 302–323. [CrossRef]
36. Agrawal, R.; Srikant, R. Mining Sequential Patterns. In Proceedings of the Eleventh International Conference on Data Engineering, Taipei, Taiwan, 6–10 March 1995; ICDE'95; pp. 3–14.
37. Zaki, M.J. SPADE: An Efficient Algorithm for Mining Frequent Sequences. *Mach. Learn.* **2001**, *42*, 31–60. [CrossRef]

38. Ayres, J.; Flannick, J.; Gehrke, J.; Yiu, T. Sequential PAttern Mining Using a Bitmap Representation. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'02, Edmonton, AB, Canada, 23–26 July 2002; ACM: New York, NY, USA, 2002; pp. 429–435. [CrossRef]
39. Yang, Z.; Wang, Y.; Kitsuregawa, M. LAPIN: Effective Sequential Pattern Mining Algorithms by Last Position Induction for Dense Databases. In Proceedings of the International Conference on Database Systems for Advanced Applications, DASFAA'07, Bangkok, Thailand, 9–12 April 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 1020–1023.
40. Han, J.; Pei, J.; Mortazavi-Asl, B.; Chen, Q.; Dayal, U.; Hsu, M.C. FreeSpan: Frequent Pattern-Projected Sequential Pattern Mining. In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD'00, Boston, MA, USA, 20–23 August 2000; ACM: New York, NY, USA, 2000; pp. 355–359. [CrossRef]
41. Pei, J.; Han, J.; Mortazavi-Asl, B.; Pinto, H.; Chen, Q.; Dayal, U.; Hsu, M. PrefixSpan: Mining Sequential Patterns by Prefix-Projected Growth. In Proceedings of the 17th International Conference on Data Engineering, Heidelberg, Germany, 2–6 April 2001; IEEE Computer Society: Washington, DC, USA, 2001; pp. 215–224.
42. Chiu, D.Y.; Wu, Y.H.; Chen, A. An efficient algorithm for mining frequent sequences by a new strategy without support counting. In Proceedings of the 20th International Conference on Data Engineering, Boston, MA, USA, 2 April 2004; IEEE Computer Society: Washington, DC, USA, 2004; pp. 375–386. [CrossRef]
43. Ahmed, C.F.; Tanbeer, S.K.; Jeong, B.S. A Novel Approach for Mining High-Utility Sequential Patterns in Sequence Databases. *Etri J.* **2010**, *32*, 676–686. [CrossRef]
44. Lan, G.C.; Hong, T.P.; Tseng, V.S.; Wang, S.L. Applying the maximum utility measure in high utility sequential pattern mining. *Expert Syst. Appl.* **2014**, *41*, 5071–5081. [CrossRef]
45. Alkan, O.K.; Karagoz, P. CRoM and HuspExt: Improving Efficiency of High Utility Sequential Pattern Extraction. *IEEE Trans. Knowl. Data Eng.* **2015**, *27*, 2645–2657. [CrossRef]
46. Wang, J.; Huang, J.; Chen, Y. On efficiently mining high utility sequential patterns. *Knowl. Inf. Syst.* **2016**, *49*, 597–627. [CrossRef]
47. Zihayat, M.; Davoudi, H.; An, A. Top-k utility-based gene regulation sequential pattern discovery. In Proceedings of the 2016 IEEE International Conference on Bioinformatics and Biomedicine (BIBM), Shenzhen, China, 15–18 December 2016; pp. 266–273. [CrossRef]
48. Zhang, C.; Du, Z.; Gan, W.; Yu, P.S. TKUS: Mining top-*k* high utility sequential patterns. *Inf. Sci.* **2021**, *570*, 342–359. [CrossRef]
49. Hong, T.P.; Lee, C.H.; Wang, S.L. Effective utility mining with the measure of average utility. *Expert Syst. Appl.* **2011**, *38*, 8259–8265. [CrossRef]
50. Lin, C.W.; Hong, T.P.; Lu, W.H. Efficiently Mining High Average Utility Itemsets with a Tree Structure. In Proceedings of the Intelligent Information and Database Systems, Hue City, Vietnam, 24–26 March 2010; Nguyen, N.T., Le, M.T., Świątek, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2010; pp. 131–139.
51. Lu, T.; Vo, B.; Nguyen, H.T.; Hong, T. A New Method for Mining High Average Utility Itemsets. In Proceedings of the Computer Information Systems and Industrial Management–13th IFIP TC8 International Conference, CISIM 2014, Ho Chi Minh City, Vietnam, 5–7 November 2014; Saeed, K., Snásel, V., Eds.; Proceedings; Lecture Notes in Computer Science; Springer: Berlin/Heidelberg, Germany, 2014; Volume 8838, pp. 33–42. [CrossRef]
52. Yun, U.; Kim, D. Mining of high average-utility itemsets using novel list structure and pruning strategy. *Future Gener. Comput. Syst.* **2017**, *68*, 346–360. [CrossRef]
53. Lin, J.C.W.; Shao, Y.; Fournier-Viger, P.; Djenouri, Y.; Guo, X. Maintenance Algorithm for High Average-Utility Itemsets with Transaction Deletion. *Appl. Intell.* **2018**, *48*, 3691–3706. [CrossRef]
54. Lin, J.C.W.; Ren, S.; Fournier-Viger, P.; Hong, T.P. EHAUPM: Efficient High Average-Utility Pattern Mining With Tighter Upper Bounds. *IEEE Access* **2017**, *5*, 12927–12940. [CrossRef]
55. Wu, J.M.T.; Lin, J.C.W.; Pirouz, M.; Fournier-Viger, P. TUB-HAUPM: Tighter Upper Bound for Mining High Average-Utility Patterns. *IEEE Access* **2018**, *6*, 18655–18669. [CrossRef]
56. Truong, T.; Duong, H.; Le, B.; Fournier-Viger, P.; Yun, U. Efficient high average-utility itemset mining using novel vertical weak upper-bounds. *Knowl.-Based Syst.* **2019**, *183*, 104847. [CrossRef]
57. Truong, T.; Duong, H.; Le, B.; Fournier-Viger, P. Efficient Vertical Mining of High Average-Utility Itemsets Based on Novel Upper-Bounds. *IEEE Trans. Knowl. Data Eng.* **2019**, *31*, 301–314. [CrossRef]
58. Le, B.; Truong, T.C.; Duong, H.V.; Fournier-Viger, P.; Fujita, H. H-FHAUI: Hiding frequent high average utility itemsets. *Inf. Sci.* **2022**, *611*, 408–431. [CrossRef]
59. Kim, H.; Yun, U.; Baek, Y.; Kim, J.; Vo, B.; Yoon, E.; Fujita, H. Efficient list based mining of high average utility patterns with maximum average pruning strategies. *Inf. Sci.* **2021**, *543*, 85–105. [CrossRef]
60. Lin, J.C.; Li, T.; Pirouz, M.; Zhang, J.; Fournier-Viger, P. High average-utility sequential pattern mining based on uncertain databases. *Knowl. Inf. Syst.* **2020**, *62*, 1199–1228. [CrossRef]
61. Wu, Y.; Lei, R.; Li, Y.; Guo, L.; Wu, X. HAOP-Miner: Self-adaptive high-average utility one-off sequential pattern mining. *Expert Syst. Appl.* **2021**, *184*, 115449. [CrossRef]
62. Wu, Y.; Geng, M.; Li, Y.; Guo, L.; Li, Z.; Fournier-Viger, P.; Zhu, X.; Wu, X. HANP-Miner: High average utility nonoverlapping sequential pattern mining. *Knowl.-Based Syst.* **2021**, *229*, 107361. [CrossRef]
63. Truong, T.C.; Duong, H.V.; Le, B.; Fournier-Viger, P.; Yun, U. Frequent high minimum average utility sequence mining with constraints in dynamic databases using efficient pruning strategies. *Appl. Intell.* **2022**, *52*, 6106–6128. [CrossRef]

64. Truong, T.C.; Duong, H.V.; Le, B.; Fournier-Viger, P.; Yun, U. Mining interesting sequences with low average cost and high average utility. *Appl. Intell.* **2022**, *52*, 7136–7157. [CrossRef]
65. Han, J.; Cheng, H.; Xin, D.; Yan, X. Frequent pattern mining: Current status and future directions. *Data Min. Knowl. Discov.* **2007**, *432*, 55–86. [CrossRef]