Towards Correlated Sequential Rules

Lili Chen, Wensheng Gan*, Chien-Ming Chen*, Senior Member, IEEE

Abstract—The goal of high-utility sequential pattern mining (HUSPM) is to efficiently discover profitable or useful sequential patterns in a large number of sequences. However, simply being aware of utility-eligible patterns is insufficient for making predictions. To compensate for this deficiency, high-utility sequential rule mining (HUSRM) is designed to explore the confidence or probability of predicting the occurrence of consequence sequential patterns based on the appearance of premise sequential patterns. It has numerous applications, such as product recommendation and weather prediction. However, the existing algorithms, known as HUSRM, is limited to extracting all eligible rules while neglecting the correlation between the generated sequential rules. To address this issue, we propose a novel algorithm called correlated high-utility sequential rule miner (CoUSR) to integrate the concept of correlation into HUSRM. The proposed algorithm requires not only that each rule be correlated but also that the patterns in the antecedent and consequent of the high-utility rule sequence be correlated. The algorithm adopts a utility-list structure to avoid multiple database scans. Additionally, several pruning strategies are used to improve the algorithm’s efficiency and performance. Based on several real-world datasets, subsequent experiments demonstrated that CoUSR is effective and efficient in terms of operation time and memory consumption.

Impact Statement—This article contributes to a correlation-based high-utility sequential rule discovery model for data prediction and artificial intelligence analytics. To the best of our knowledge, it is the first article that proposes a realistic correlation-based solution for high-utility sequential rule mining instead of discovering patterns or rules uncorrelated in real-world databases. The successful application of rule discovery and high-utility pattern mining algorithms can bring great business value in engineering management and profit generation. Since CoUSR can accurately predict the occurrence of sequential patterns with correlation, it can be used in many different applications and domains, such as market basket analysis, risk prediction, and intrusion detection.

Index Terms—rule discovery, utility mining, sequential rules, correlation, correlated sequential rules.

I. INTRODUCTION

A wide variety of devices generate massive amounts of data, precipitating the arrival of the data age. Data mining has become a popular research area because it uncovers useful and valid information in vast amounts of data, promoting economic and social progress [1], [2]. Initially, researchers considered that items in transaction databases either existed or did not, and that the extracted patterns occurred frequently, which is called frequent pattern mining (FPM) [3], [4]. The FPM assumes that all items in a transaction occur simultaneously, but in reality, items occur in a specific order. A sequential pattern is a collection of items with a time dimension, and sequential pattern mining (SPM) is the process of discovering critical information derived from sequential databases [5], [6], [7], [8]. SPM is expected to find sequential patterns that satisfy the support requirements. Its application involves a host of aspects, such as web logs and market analysis. Agrawal and Srikant [9] first introduced the problem of SPM in the early 1990s. However, even though some frequently occurring sequential patterns are known, there is insufficient evidence to make predictions regarding the events. For example, suppose there is a frequent sequential pattern ab in which a may occur before or after b, it is unwise to apply this pattern to predict whether a occurs or b follows.

In terms of chronological properties and prediction of events, sequential rule mining (SRM) [10], [11] outperforms SPM. A sequential rule states that if some items occur as antecedents, some non-overlapping items may appear later as consequent with a given degree of confidence. The task of mining rules aims to predict patterns or events that have not yet occurred. We can predict the likelihood or relevance of a pattern in the future based on the existence of a pattern. Several algorithms have been proposed for SRM. Some of them discover rules from a single sequence, and others find rules from multiple sequences. Some of the sequential rules, both the antecedent and the consequent, are sequential patterns, while the others are composed of two unordered collections of items. Mannila et al. [12] discovered frequent events partially ordered in a sequence. Hamilton et al. [13] presented a temporal investigation method for registered record sequences. Harms et al. [13] proposed the MOWCATL algorithm to find rules whose support and confidence are both greater than the corresponding thresholds. In this study, we only consider SRM from a few sequences and partially disordered items.

In terms of FPM, it removes the limitations that items in a transaction may appear more than once and that the weight or profit of items varies in real life [15]. Therefore, FPM leads to the discovery of frequent patterns, but the profits obtained are not satisfactory, particularly for dissatisfied retailers or manufacturers. To address these issues, high-utility pattern mining (HUPM) [16] was developed, which considers not only the number of patterns but also their unit profit. In general, HUPM can be divided into two categories: two-phase and one-phase algorithms, where the former generates many candidates, but the latter does not. Some representative two-
phase algorithms include Two-Phase [17], UP-Growth [18], and UP-Growth+ [19]. Representative one-phase algorithms include HUI-Miner [20] and FHM [21]. High-utility sequential pattern mining (HUSPM) is derived by extending HUPM to the temporal aspect [22], [23]. Yin et al. [22] developed a generic framework and proposed an efficient algorithm called USpan. They designed a lexicographic quantitative sequence tree for searching full high-utility sequential patterns (HUSPs). To cope with the requirements of prediction with utility, HUSRM [24] was developed. HUSRM [24] is a one-phase algorithm that depends on the utility-table structure. Additionally, several optimization strategies have been employed to improve the performance of the algorithm.

However, because these rules are constrained only by utility and confidence, the items in those are uncorrelated, which may lead to decision-making errors [25], [26]. For example, the existing algorithm reveals that the rule of first selling a TV and bread together in a sequential database, followed by apples and sofas being sold together with a certain degree of confidence, is a HUSR. It is clear that the items in the antecedent and the consequent of the rule are hardly correlated, and the rule’s antecedent and consequent are also weakly correlated. To date, several correlated HUPM algorithms have been developed, such as HUIPM [27], FDHUP [28], and CoUPM [25]. However, the algorithms for mining no correlated HUSRs (CHUSRs) have been proposed.

To deal with the problems mentioned above, we incorporate the concept of correlation to HUSRM. Here are the new ideas and contributions of this study:

- This study develops an efficient CoUSR algorithm for mining CHUSRs from a given sequence database. To the best of our knowledge, this is the first study to involve mining correlated utility-based sequential rules.
- The correlation of rules has two aspects. On the one hand, it is a local correlation that the antecedent or consequent within the rule is constrained by the bond measure. On the other hand, it is a global correlation in which sequential rules are constrained by the lift method.
- CoUSR is a one-phase algorithm that uses a utility-list structure to maintain data and does not require multiple database scans. Two novel structures called BondMatrix and estimated sequence utility co-occurrence structure (ESUCS) are constructed, and several pruning strategies are designed to assist in pruning those unpromising sequential rules and reduce the search space.
- To evaluate the performance of the algorithm, subsequent experiments measure the efficiency and feasibility of the proposed algorithm under different strategies.

The remainder of this paper is organized as follows. Section II presents relevant literature on the proposed algorithm. Some preliminaries and problem statements are introduced in Section III. Section IV describes the details of the designed strategies and the proposed algorithm. The experimental comparison is presented in Section V. Finally, Section VI concludes this study and highlights future studies.

II. RELATED WORK

In this section, we review the related literature on the proposed algorithm, including HUSPM, SRM, and CHUPM.

A. High-Utility Sequential Pattern Mining

Frequency-based sequential pattern mining [5], [6] has attracted a lot of attention in the past decades. High-utility SPM (HUSPM for short) plays an important role in decision-making. To date, a series of relevant algorithms have been developed. The SPM was originally proposed by Agrawal and Srikant [29] to address the challenge of discovering meaningful subsequences in a collection of sequences. In terms of utility-based issues, Ahmed et al. [30] introduced two algorithms in 2010: UtilityLevel, which applies the method of candidate generation and testing, and UtilitySpan, which uses a pattern growth approach. Yin et al. [22] then developed a generic framework called USpan to address the HUSPM problem. A lexicographic quantitative sequence tree and two concatenation mechanisms were also proposed [22]. To improve the efficiency, several advanced algorithms with new data structures and new pruning strategies, such as HUSSpan [31], ProUM [32], HUSP-ULL [33] have been proposed. In addition to efficiency, the effectiveness is also the key issue of data mining algorithms, thus some studies focus on mining top- k HUSPs [23], [31], [34], incremental HUSPM [35], HUSPM over data streams [36], HUSPM with negative item values [37], mining multi-dimensional HUSPs using MDUS_{EM} and MDUS_{SD} [38], and on-shelf availability for HUSPM [39]. Additionally, MAHUSP [36] employed memory-adaptive mechanisms by taking advantage of the upper bound of memory, and the MAS-tree was also designed to maintain potential HUSPs. Other related works of utility-driven SPM and advanced utility mining tasks can be referred to the review literature [16].

B. Sequential Rule Mining

Different from sequential pattern mining (SPM) [6], [40], the SRM uses rules from frequent patterns in a set of sequences, which works well in prediction. Lo et al. [41] considered that a rule is redundant when it has the same level of support and confidence as inferred rules. Based on logical inference, a non-redundant rule was characterized, and a related algorithm was proposed. Then, Fournier-Viger et al. [11] proposed CMRules to generalize the point that there is no order for items in the antecedent and consequent of all rules. Its intention is to first transform the sequence database into a transaction database by ignoring the temporal information, then mine association rules in the transaction database, and then filter out sequential rules in these association rules through the original sequence database. Obviously, this would be inefficient. Subsequently, RuleGrowth [42] was proposed using a pattern-growth approach, which is more efficient and scalable. First, the shortest rules are found, and then the left and right expansions are used to find full sequential rules. TRuleGrowth [43] was proposed, and it created a sliding window to manage the maximum number...
of rule occurrences. The utility-driven SRM is introduced for utility-based sequential rule mining by extending sequential rule mining for utilities. To date, there are only two relevant algorithms: HUSRM [24] and e-HUNSR [43]. The HUSRM combines sequential rules with utility to establish a framework of utility-based sequential rule mining. It is primarily depth-first searching rules, and the mining procedure is similar to RuleGrowth [42]. The HUSRM algorithm employs sequence-estimated utility as an upper bound of the utility of sequential rules, which can trim off some unpromising items and rules in advance. Furthermore, using the information in the utility-list structure not only reduces the database access but also accelerates the mining process by revealing a tighter upper bound on utility. On this basis, e-HUNSR [44] considered that some events do not occur. e-HUNSR proposed some solutions to address the challenges of some intrinsic complexities.

C. Correlated High-Utility Pattern Mining

Typical HUPM algorithms use utility as the sole criterion, resulting in almost no correlation between items in the patterns. To prevent this situation, a number of algorithms have been proposed to consider both utility and correlation to discover correlated high-utility patterns (CHUPs). The HUIPM was originally proposed to mine high-utility interesting patterns with frequency affinity [27]. In this algorithm, the knowledge-weighted utilization (KWU) of a pattern is used as an upper bound to prune patterns in advance. Then, FDHUP primarily improved the efficiency of HUIPM by developing a novel EI-table and FU-table data structure [28]. Following that, FCHM-bond and FCHM-all-confidence were proposed as bond and all-confidence measures [26], [45], respectively. Both of these methods have anti-monotonicity, which makes pruning patterns easier. Subsequently, CoHUI-Miner [46], CoUPM [25], and CoHUI-Miner [47] were developed to discover high-utility patterns using the Kule measure [15]. They are all one-phase algorithms that employ projection databases and utility lists to store significant information. Clearly, the concept of correlation is important in real-world applications. The problem of mining CHUSRs is discussed in this study. Our CoUSR algorithm discovers pattern information with sequential properties on the basis of CoUPM [25] for more valuable patterns, which is an extension of the former one.

III. PRELIMINARY AND PROBLEM STATEMENT

To better explain the proposed algorithm, some essential definitions are introduced in advance. A sequential database SD consists of a set of sequences denoted as SD = \{S_1, S_2, \ldots, S_n\}, where 1, 2, \ldots, and n represent sequence identifiers. Let I = \{i_1, i_2, \ldots, i_m\} be the items in the sequential database. A sequence S is composed of a collection of itemsets with temporal order \prec, which is expressed as S \subseteq \{I_1, I_2, \ldots, I_l\}, where I_l is not only a subset of I but also disjoint subsets of I. Additionally, because there may be more than one item in a sequence S, q(i, S) is employed to indicate the number of items i in S, that is, its internal utility. Each item is associated with a value p(i) corresponding to the weight (i.e., external utility). In a sequence, there is an order between itemsets and a disorder within them. To illustrate examples, a sequence database and a utility table are shown in Tables I and II respectively.

| Item | a | b | c | d | e | f | g |
|------|---|---|---|---|---|---|---|
| Utility | 3 | 5 | 2 | 1 | 6 | 3 | 2 |

As shown in Table I there are five sequences, S_1, S_2, S_3, S_4, and S_5 in the sequence database. There are four itemsets in S_1, which are \{a, b\}, \{e, d\}, \{g\}, and \{c\}. Among them, the quantity or internal utility of a is 1, and that of b is 1. Additionally, as shown in Table II, the external utilities of a to g are $3, $5, $2, $1, $6, $3, and $2$.

**Definition 1:** Given two unordered non-empty itemsets X and Y, they can form a rule \( r: X \Rightarrow Y \) if X and Y are subsets of I and there is no intersection between them. Rule r indicates that if X appears in a sequence, then Y will also appear in the same sequence. Assume the number of items in X is k and the number of items in Y is m, implying that \(|X| = k\) and \(|Y| = m\). Then, the size of a rule \( r: X \Rightarrow Y \) is defined as \( k \cdot m \), where the symbol \( \cdot \) does not represent a product. Let the size of another rule \( r': X' \Rightarrow Y' \) be \( t \cdot u \). If \( t > k \) and \( u > m \), or \( t > k \) and \( u > m \), we consider \( r' \) to be greater than \( r \) naturally.

Consider a simple example: the size of \( r': \{a, b, c, d\} \Rightarrow \{e, f\} \) is \( 4 \cdot 2 \) and the size of \( r: \{a\} \Rightarrow \{f\} \) is \( 1 \cdot 1 \). Because \( 4 > 2 \) and \( 2 > 1 \), we can come to the conclusion that the size of \( r' \) is greater than that of \( r \).

**Definition 2:** Let a sequence \( S_c = \{I_1, I_2, \ldots, I_l\} \) contain l itemsets. If the itemset \( I_g \) is a union of some itemsets in \( S_c \), \( I_g \) occurs in \( S_c \). For a rule \( r: X \Rightarrow Y \), assuming that there exists an integer \( p \), such that \( l > p \geq 1 \), \( X \subseteq \bigcup_{i=p+1}^{l} I_i \) and \( Y \subseteq \bigcup_{i=p}^{l} I_i \), we consider that \( r \) appears in \( S_c \) or \( S_c \) supports \( r \). We denote the sequences containing a sequential rule \( r \) as sids(r), and the sequences comprising an itemset X are denoted as sids(X). The confidence of a rule \( r: X \Rightarrow Y \) is defined as \( \text{conf}(r) = |\text{sids}(r)|/|\text{sids}(X)| \).

As shown in Table I the rule \( a \Rightarrow b \) occurs in sequences S_2 and S_3; therefore, \( \text{sids}(r) = \{S_2, S_3\} \). Similarly, \( \text{a} \), the antecedent of \( r \) appears in all five sequences, that is, \( \text{sids}(a) = \{S_1, S_2, S_3, S_4, S_5\} \). Therefore, the confidence level of \( r \) is calculated to be \( 2/5 = 0.4 \).

**Definition 3:** The utility of an item i in a sequence \( S_c \) is denoted as \( u(i, S_c) = q(i, S_c) \times p(i) \). The utility of a rule \( r: X \Rightarrow Y \) in \( S_c \) is defined as \( u(r, S_c) = \sum_{i \in X} u(i, S_c) + \sum_{i \in Y} u(i, S_c) \) when \( r \) occurs in \( S_c \). Moreover, we define the
utility of \( r \) in the sequence database \( SD \) as the sum of utilities of \( r \) in all sequences containing it, and is denoted as \( u(r) = \sum_{S_i \in SD} u(r, S_i) \) \[24\].

Following the above example, the utilities of items \( a \) and \( b \) in the sequence \( S_2 \) can be calculated as \(2 \times 3 = 6 \) and \(1 \times 5 = 5\), respectively. Thus, the utility of \( r \) is equal to the sum of \( a \) and \( b \) utilities in \( S_2 \), which is \$11. Furthermore, the utility of \( r \) in the sequence database is calculated as \( u(r, S_2) + u(r, S_1) = \$24\).

Definition 4: A high-utility sequential rule (HUSR) \( r \) must satisfy two conditions: its confidence must be greater than or equal to the minimum confidence threshold \( \text{minconf} \in [0, 1] \), and its utility must be greater than or equal to the minimum utility threshold \( \text{minutil} \in R^+ \), which can be denoted as \( \text{conf}(r) \geq \text{minconf} \) and \( u(r) \geq \text{minutil} \), respectively. The task of discovering HUSRs from a sequence database is to determine all valid rules.

Although HUSRs have a broad application in prediction, there are some disadvantages, such as the fact that the obtained rules are weakly correlated, which causes some decision-making issues. To address this issue, we present two types of correlation concepts, called bond and lift, and incorporate them into HUSRs to form the concept of CHUSRs.

Definition 5: We define the support of a rule \( r \) denoted as \( \text{sup}(r) = |\text{sids}(r)| \) in a sequence database. Moreover, the support of an itemset \( X \) is defined as \( \text{sup}(X) = |\text{sids}(X)| \). The disjunctive support of an itemset \( X \) is defined as the number of transactions containing any item in \( X \) and is denoted as \( \text{dissup}(X) = |\{S_c \in SD \mid X \cap S_c \neq \emptyset \}| \). The bond of an itemset \( X \) is denoted as \( \text{bond}(X) = \text{sup}(X) / \text{dissup}(X) \).

Definition 6: A rule \( r: X \rightarrow Y \) is locally correlated if and only if its antecedent and consequent satisfy that \( \text{bond}(X) \geq \text{minbond} \) and \( \text{bond}(Y) \geq \text{minbond} \), where \( \text{minbond} \in [0, 1] \) is a predetermined expert-specified minimum bond threshold.

For rule \( r: \{a, b\} \Rightarrow \{g\} \) appearing in sequence \( S_1, S_2, S_4 \), and \( S_5 \), \( \text{sup}(r) \) is calculated as four. Because itemset \( \{a, b\} \) occurs in five sequences, \( \text{sup}(\{a, b\}) \) is five. Because each sequence contains a subset of \( \{a, b\} \), \( \text{dissup}(\{a, b\}) \) is also equal to five. Therefore, \( \text{bond}(\{a, b\}) \) is calculated as \( \text{sup}(\{a, b\}) / \text{dissup}(\{a, b\}) = 1 \). Because the bond of a single item is always one, \( \text{bond}(\{g\}) = 1 \). If \( \text{minbond} \) is 0.3, \( r \) is locally correlated.

Definition 7: Given a rule \( r: X \Rightarrow Y \), the lift of \( r \) is defined as \( \text{lift}(r) = \text{conf}(r) / \text{sup}(Y) / \text{sup}(X) \) or \( \text{lift}(r) = \text{conf}(r) / \text{sup}(X) \times \text{sup}(Y) \). A rule \( r: X \Rightarrow Y \) is globally correlated if it satisfies the condition that \( \text{lift}(r) \geq \text{minlift} \), where \( \text{minlift} > 1 \) is a predefined minimum lift threshold. If a HUSR is both locally and globally correlated, we regard it as a CHUSR.

\( \text{lift}(r) \) denotes the ratio of the support of \( Y \) under the condition of \( X \) and that of \( Y \) without any condition \[15\]. If \( \text{lift}(r) = 1 \), it indicates that \( X \) and \( Y \) are independent of each other and \( X \) has no lifting effect on the occurrence of \( Y \). A larger value indicates a greater lift of \( X \) on \( Y \) and a stronger correlation between them. If the value of \( \text{lift}(r) \) is less than one, the antecedent and consequent of the rule are mutually exclusive. If the value is one, the two components of \( r \) are independent. If the value is greater than one, the rule is strongly correlated. To ensure that the derived rules are correlated, we established a lift range greater than one. In the running example, \( \text{lift}(\{a, b\} \Rightarrow \{g\}) = (5 \times 4) / (5 \times 4) = 1 \), which is equal to one. Therefore, this rule is not globally correlated. When we set \( \text{minconf} \) to 0.7, \( \text{minutil} \) to 50, \( \text{minbond} \) to 0.3, and \( \text{minlift} \) to 1.1, all the generated rules are listed in Table I.

### IV. PROPOSED CoUSR ALGORITHM

In this section, we first introduce the downward closure properties of utility and correlation, and then present a compact data structure named utility-list to store critical information. Subsequently, some pruning strategies are proposed to optimize the performance of the CoUSR. Finally, the details and pseudocode of the proposed algorithm are presented.

#### A. Anti-monotonicity of Sequence Estimated Utility and the Bond Measure

In previous studies, it was discovered that utility lacks monotonicity and anti-monotonicity, contributing to an explosion in the search space. To address this dilemma, the sequence-estimated utility is proposed to prune sequential rules.

Definition 8: We define the sequence utility as the total of utilities for each item in a sequence \( S_i \), which can be denoted as \( \text{SU}(S_i) = \sum_{i \in S_i} u(i, S_i) \). Additionally, the sequence-estimated utility of an item \( i \) is defined as the sum of sequence utilities where sequences contain \( i \). The expression can be written as \( \text{SEU}(i) = \sum_{S_i \ni i} \text{SU}(S_i) \). We define the sequence-estimated utility of a rule \( r \) as the summary of the sequence utility of the sequences containing rule \( r \), and write it as \( \text{SEU}(r) = \sum_{S_i \ni \text{sids}(r)} \text{SU}(S_i) \) \[24\].

The last column in Table I shows the sequence utility of each sequence. According to the calculation results, \( \text{SU}(S_1) = \$21, \text{SU}(S_2) = \$34, \text{SU}(S_3) = \$28, \text{SU}(S_4) = \$22, and \text{SU}(S_5) = \$42 \). It is already known that the rule \( r: X \Rightarrow Y \) appears in the sequence \( S_2 \) and \( S_3 \); thus, \( \text{SU}(r) = \text{SU}(S_2) + \text{SU}(S_3) = \$62 \).

According to Definition 8, the \( \text{SEU} \) of an item or rule must be no less than its real utility, as well as the utility of its expansion rules. Hence, \( \text{SEU} \) serves as an upper bound on the utility. It is apparent that \( \text{SEU} \) has a downward closure property, which grounds the two properties introduced below.

Property 1: For an item \( i \), assuming that \( \text{SEU}(i) < \text{minutil} \), the item is said to be unpromising; otherwise, it is promising. Furthermore, if an item is unpromising, the rules containing that item are unlikely to be eligible \[24\].

### TABLE III

| CHUSR       | utility | conf | lift |
|-------------|---------|------|------|
| \{a,b,c,d\} | 55      | 1.0  | 1.25 |
| \{a,b,d\}  | 74      | 1.0  | 1.25 |
| \{a,d\}    | 54      | 1.0  | 1.25 |
| \{b,d\}    | 53      | 1.0  | 1.25 |
Property 2: For a rule \( r \), if \( SEU(r) < \text{minutil} \), \( r \) is believed to be unpromising; otherwise, \( r \) is promising. Neither are the rules derived from the expansion of this rule, which is expected to be promising \[24\].

Based on these two properties, a few pruning strategies can be used to remove some items and rules that are unsatisfactory.

Strategy 1: If the inequality \( SEU(i) < \text{minutil} \) for an item \( i \) holds, this item and all sequential rules containing \( i \) can be pruned directly.

Strategy 2: If the inequality \( SEU(r) < \text{minutil} \) for a rule \( r \) holds, \( r \) and its expansions can be pruned directly.

According to Definition \[5\] the bond measure has a downward closure property because, as the number of items in itemset \( X \) increases, the number of sequences containing \( X \) decreases, while the number of sequences containing any item in \( X \) increases, decreasing the bond of \( X \).

Property 3: For a rule \( r: X \Rightarrow Y \), assuming that \( \text{bond}(X) < \text{minbond} \) or \( \text{bond}(Y) < \text{minbond} \), this rule and all its expansions are unpromising.

We can propose the following pruning strategy based on this property.

Strategy 3: If the bond of the antecedent or consequent of a sequential rule is less than \( \text{minbond} \), then the rule and any of its expansions can be pruned directly.

B. Utility-list Structure

This subsection describes in detail a novel utility-list structure \[24\]. Before that, we must address two issues that may result in redundant rules. We observe that by varying the combinations of left- and right-expand operations, we can obtain a specific rule. For example, a rule \( r: \{a, c\} \Rightarrow \{e, g\} \) is given. If the initial rule is \( \{a\} \Rightarrow \{e\} \), the target rules can be derived not only from the primitive rule by a left expansion followed by a right expansion, but also by a right expansion followed by a left expansion. As shown in Fig. 1 different trails can lead to the same results. A simple and naive approach is to allow left expansions after right expansions but not right-expansions after left expansions. According to Fig. 2 there are no redundant rules to generate. Of course, it is also possible to allow right-expansions after left-expansions, but not to permit left-expansions after right-expansions.

Another discovery potentially influencing the accuracy of the results is that it is possible to obtain the same rule for either left- or right-expansions with various items. Take an example of the rule \( r: \{a, c\} \Rightarrow \{g\} \), which can be achieved either by \( \{a\} \Rightarrow \{g\} \) through left-expansions with \( c \), or by \( \{c\} \Rightarrow \{g\} \) through right-expansion with \( a \). To accommodate such problems, we restrict the expansion according to the order of the alphabet denoted by \( \prec \), and in the above example, there is no possibility of expanding \( \{c\} \Rightarrow \{g\} \) to \( \{a, c\} \Rightarrow \{g\} \).

Because larger rules are derived from smaller rules via left- or right-expansions, it is necessary to clarify whether an item can expand left or right. Next, the relevant definitions are provided.

Definition 9: Assume that a sequential rule \( r: X \Rightarrow Y \) appears in a sequence \( S \). An item \( i \) available for expanding \( r \) by left expansion must satisfy \( i \prec j, \forall j \in X, i \notin Y \), and \( X \cup \{i\} \Rightarrow Y \) appear in \( S \). Similarly, an item \( i \) capable of expanding \( r \) via right expansion must fulfill the condition that \( i \prec j, \forall j \in Y, i \notin X \), and simultaneously \( X \Rightarrow Y \cup \{i\} \) appear in \( S \) \[24\].

In Definition \[9\] it is discovered that in a sequence \( S \), some items can only expand rule \( r \) through left expansion, denoted as \( \text{onlyLeft}(r, S) \), and some items only expand rules via right expansion expressed as \( \text{onlyRight}(r, S) \), while some other items have both properties represented as \( \text{leftRight}(r, S) \). After laying the foundation, it is time to formally define the utility-list.

Definition 10: The utility-list of a rule \( r \) is denoted as \( UL(r) \). Each row of a utility-list consists of a five-tuple \( (S_{sid}, \text{iutil}, \text{lutil}, \text{rutil}, \text{lutil}) \), where \( S_{sid} \) indicates the identifier of the supporting sequence, the element \( \text{iutil} \) implies the real utility of \( r \) in sequence \( S_{sid} \), and the element \( \text{lutil} \) and \( \text{rutil} \) are the sum of utilities of items in \( \text{onlyLeft}(r, S_{sid}) \) and \( \text{onlyRight}(r, S_{sid}) \), respectively.

| [a] → [e] |
|------------|
| \[\text{SID} \] | \[\text{iutil} \] | \[\text{lutil} \] | \[\text{rutil} \] | \[\text{lrutil} \] |
| \[S_1\] | 9 | 5 | 2 | 0 |
| \[S_2\] | 12 | 18 | 4 | 0 |
| \[S_3\] | 15 | 10 | 0 | 3 |
| \[S_4\] | 9 | 7 | 6 | 0 |
| \[S_5\] | 15 | 5 | 11 | 0 |

Fig. 3. The utility-list of \( \{a\} \Rightarrow \{e\} \)
is clear that they have significant hidden properties that can be exploited.

Property 4: The utility of a rule \( r \) in a sequence database is equivalent to the sum of \( iutil \) in each row of \( UL(r) \) \([24]\).

Property 5: The support of the given rule \( r \) is corresponding to the number of rows in \( UL(r) \) \([24]\).

Property 6: An interesting observation in the utility-list \( UL(r) \) corresponding to rule \( r \) is that the sum of \( iutil, lutil, rutil, \) and \( lrutil \) of all tuples is always greater than \( u(r) \), as are the utilities of left- and right-expansions of \( r \). Moreover, because the sum is less than \( SEU(r) \) for good, it has a tighter upper bound than \( SEU(r) \) \([24]\).

Property 7: Another observation is that the utilities of the left-expansions of a rule \( r \) in \( UL(r) \) are no greater than the sum of \( iutil, lutil, \) and \( rutil \). Similarly, it has a more compact upper bound than \( SEU(r) \) \([24]\).

With a more compact utility upper bound, the above two properties can be used in the expanding operation to reduce the search space even more.

Strategy 4: Let a sequential rule \( r \) correspond to a utility-list \( UL(r) \). If the sum of utilities in \( UL(r) \) is less than \( minutil, r \) and its right expansions are capable of being pruned ahead of time \([24]\).

Strategy 5: Let the utility-list of sequential rule \( r \) be \( UL(r) \). If the sum of utilities except \( rutil \) in \( UL(r) \) is less than \( minutil, r \) and its left expansions are capable of being pruned in advance \([24]\).

There is no need to explore the database again to construct the utility-lists of larger rules after building the utility-lists of all 1*1 rules. We can derive the utility-lists of larger rules by the utility-lists of larger rules after building the utility-lists of in advance \([24]\).

\[
\text{Strategy 4: Let a sequential rule } r \text{ correspond to a utility-list } UL(r). \text{ If the sum of utilities in } UL(r) \text{ is less than } minutil, r \text{ and its right expansions are capable of being pruned ahead of time } [24].
\]

\[
\text{Strategy 5: Let the utility-list of sequential rule } r \text{ be } UL(r). \text{ If the sum of utilities except } rutil \text{ in } UL(r) \text{ is less than } minutil, r \text{ and its left expansions are capable of being pruned in advance } [24].
\]

C. Optimization with Bit Vectors

The preceding discussion focused on the properties and upper bounds of the utility; however, some improvements are made in other constrained measures. To calculate the confidence, bond, and lift of rule \( r: X \Rightarrow Y \), we need to know \( sup(r), sup(X), sup(Y), dissup(X), \) and \( dissup(Y) \). In fact, \( sup(r) \) is easily accessible from the utility-list of \( r \), where \( sup(r) \) is the number of rows in the utility-list. However, the others are difficult to calculate because it is not feasible to scan the database every time they are needed.

A practical and efficient approach is to create bit vectors for any items existing in the sequences at the beginning. Assuming a single item appears in the \( jth \) sequence, the \( jth \) bit is set to one; otherwise, it is set to zero. The support of an itemset \( X \) is the length of the intersection of the bit vectors of the items in \( X \). By analogy, the disjunctive support of \( X \) is equal to the length of the union of the bit vectors of the items in it. For example, \( bv(a) = 11111 \) and \( bv(c) = 01001 \). Note that \( sup(ac) = |bv(a) \cap bv(c)| = |01001| = 2 \) and \( dissup(ac) = |bv(a) \cup bv(c)| = |11111| = 5 \).

This study follows the rule of allowing left expansions after right expansions but not right expansions after left expansions. If an alternative order of expansion is adopted, which allows right expansions after left expansions but not left expansions after right expansions, a critical property in terms of confidence can be discovered. The performance of the developed algorithm can be significantly improved by employing this property for pruning unqualified rules.

Property 8: If the confidence of a rule is less than \( \minconf \), there is no need to expand it via the right expansion.

Proof: Let the rules \( r: X \Rightarrow Y \) and \( r': X \Rightarrow Y \cup \{c\} \) be present in the database. Two equations follow from the definition of confidence: \( conf(r) = \frac{|sids(r)|}{|sids(X)|} \) and \( conf(r') = \frac{|sids(r')|}{|sids(X)|} \) hold. Therefore, the inequality \( conf(r) \geq conf(r') \) holds because \( |sids(r')| \leq |sids(r)| \).

D. The Designed Pruning Strategies

In this subsection, we propose two novel data structures based on utility and bond to propose some pruning strategies and to abandon unpromising rules in advance. This can assist in improving the efficiency and performance of the proposed algorithm. The details are as follows:

Because the bond measure is anti-monotonicity, a structure similar to EUCS \([21]\) is designed to maintain the bond value between two items during the second scans of the sequence database. The Bond Matrix is developed as a series of triples of the form \( \text{BondMatrix}(a, b) = c \), where \( a \in I, b \in I, a \prec b \) by dictionary order, and \( c \in [0, 1] \). This implies that \( c \) is the bond between \( a \) and \( b \), which is the bond \( (ab) \). For example, Fig. 4 depicts the bond matrix of the items in Table I.

Strategy 6: Assume that the last items of the antecedent and consequent of a rule \( r: X \Rightarrow Y \) are \( x \) and \( y \), respectively. If \( r \) is left extended with an item \( i \) to form \( r': X \cup \{i\} \Rightarrow Y \) such that \( \text{BondMatrix}(x, i) = 0 \) or \( \text{BondMatrix}(x, i) < \minbond \), \( r' \) and all its expansions can be directly pruned. Similarly, if \( r \) is right extended with an item \( i \) to form \( r': X \Rightarrow Y \cup \{i\} \) such
Algorithm 1: The CoUSR algorithm

Input: SD, a sequence database; minutil, the minimum utility threshold; minconf, the minimum confidence threshold; minbond, the minimum bond threshold; minlift, the minimum lift threshold.

Output: a collection of CHUSRs.

1. scan SD to calculate the SEU of each item \( i \in I \);
2. \( I^* \leftarrow \{ i \mid i \in I \land SEU(i) \geq \text{minutil} \} \);
3. filter out items from SD that \( j \in I \) such that \( j \notin I^* \);
4. scan SD to calculate the bit vector of each item \( i \in I^* \) and build the bond matrix, BondMatrix;
5. scan SD to calculate \( R \), the set of rules of the form \( i \Rightarrow j(i, j \in I^*) \) appearing in SD and calculate \( SEU(r), \text{sids}(r) \) and meanwhile build ESUCS;
6. \( R^* \leftarrow \{ r \mid r \in R \land SEU(r) \geq \text{minutil} \} \);
7. construct the initial utility-lists \( UL(r) \), \( r \in R^* \);
8. for each rule \( r \in R^* \) do
9. \( \text{if } UL(r).\text{lrutil} \geq \text{minlift} \land \text{conf}(r) \geq \text{minconf} \land \text{minconf} \land \text{minbond} \geq \text{minlift} \) then
10. \( \text{CHUSR} \leftarrow \text{CHUSR} \cup r \);
11. end
12. \( \text{if } (UL(r).\text{lrutil} + UL(r).\text{ultil}) \geq \text{minutil} \) then
13. \( \text{call RightExpansion}(r, SD, \text{minutil}, \text{minconf}, \text{minbond}, \text{minlift}, \text{sidsX}, \text{sidsY}, \text{SIDsORY}, \text{SIDsORX}) \);
14. end
15. \( \text{if } (UL(r).\text{lrutil} + UL(r).\text{ultil}) \geq \text{minutil} \) then
16. \( \text{call LeftExpansion}(r, SD, \text{minutil}, \text{minconf}, \text{minbond}, \text{minlift}, \text{sidsX}, \text{sidsY}, \text{SIDsORX}) \);
17. end
18. end

Initially, the database is iterated once to obtain the \( SEU \) of all items in \( I \), and filter out some unsatisfactory items whose \( SEU \) is less than \( \text{minutil} \) to obtain \( I^* \) (according to Strategy 1). The database is scanned again to record the bit vector of each item in \( I^* \) and construct the BondMatrix structure for later reference according to \( I^* \). Subsequently, one more database scan is performed to calculate the \( SEU(r) \) and the set of sequences \( \text{sids}(r) \) containing the rules \( r \), whose size is one star one. Simultaneously, the ESUCS is built during this database search. It is reasonable to discard unpromising rules whose \( SEU \) is less than \( \text{minutil} \) and access \( R^* \) (according to Strategy 2). At this point, initial utility-lists \( UL(r) \) for each promising sequential rule \( r \) can be constructed. Subsequently, the search process is initiated. This is the depth-first search algorithm. The utility of each rule \( r \) in \( R^* \) can be easily obtained using \( UL(r) \), and the confidence and lift values can be obtained by intersecting and combining the bit vectors. If a rule \( r \) simultaneously satisfies that the utility is greater than \( \text{minutil} \), the confidence is greater than \( \text{minconf} \), and the list is greater than \( \text{minlift} \), it is a CHUSR and stored in the CHUSR collection. If the sum of utilities in \( UL(r) \) satisfies

Because the sequence-estimated utility of a rule has a downward closure property, we design an estimated sequence utility co-occurrence structure (ESUCS) to maintain the \( SEU \) between two items. The ESUCS is a set of tuples of the form \( \text{ESUCS}(a, b) = c \), where \( a \in I \), \( b \in I \), and \( c \in R^+ \). Note that \( a \) and \( b \) belong to different itemsets, with \( a \) appearing first in temporal order and \( b \) appearing later. For example, the ESUCS formed by a running example is illustrated in Fig. 5. ESUCS is similar to EUCS [21] except that with the addition of the order between items, the values of ESUCS \( (a, b) \) and ESUCS \( (b, a) \) are not identical, and they represent the \( SEU \) of the rules \( a \Rightarrow b \) and \( b \Rightarrow a \), respectively. While there is no order between items in EUCS, EUCS \( (a, b) \) is equal to EUCS \( (b, a) \), which is an upper triangular structure with symmetry.

Strategy 7: Assume that the last items of the antecedent and consequent of a rule \( r \): \( X \Rightarrow Y \) are \( x \) and \( y \), respectively. If \( r \) is left extended with an item \( i \) to form \( r' \): \( X \cup \{ i \} \Rightarrow Y \); thus, \( \text{ESUCS}(i, y) = \text{null} \) or \( \text{ESUCS}(i, y) < \text{minutil} \), then \( r' \) and all its expansions can be immediately trimmed. Similarly, if \( r \) is right extended with an item \( i \) to form \( r' \): \( X \Rightarrow Y \cup \{ i \} \), resulting in \( \text{ESUCS}(x, i) = \text{null} \) or \( \text{ESUCS}(x, i) < \text{minutil} \), then \( r' \) and all its expansions can also be immediately trimmed.

E. The Proposed Algorithm

Algorithm 1 exhibits the core pseudocode of the CoUSR, integrating the critical aspects from previous discussions. The CoUSR algorithm has five inputs, and they are a sequence database \( SD \), the minimum utility threshold \( \text{minutil} \), the minimum confidence threshold \( \text{minconf} \), the minimum bond threshold \( \text{minbond} \), and the minimum lift threshold \( \text{minlift} \). The algorithm assumes complete CHUSRs.
Algorithm 2: RightExpansion(r, SD, minutil, minconf, minbond, minlift, SIDsX, SIDsY, SIDsORX, SIDsORY)

Input: r: X ⇒ Y, a sequence rule; SD; minutil; minconf; minbond; minlift; SIDsX, the set of sequences containing X; SIDsY, the set of sequences containing Y; SIDsORX, the set of sequences containing any one of the items in X; SIDsORY, the set of sequences containing any one of the items in Y.

for sequence S ∈ sids(r) do
  for rule e: X ⇒ Y \{i\} \ni leftRight(r, S) \cup onlyRight(r, S) \cup onlyLeft(r, S) do
    x is the last item in X;
y is the last item in Y;
    if ESUCS(x, i) < minutil then
      continue;
    end
    if BondMatrix(y, i) < minbond then
      continue;
    end
    if bond(Y \cup i) ≥ minbond then
      newSIDsY = SIDsY \cap sids(i);
      newSIDsORY = SIDsORY \cup sids(i);
      conf(e) = |sids(e)| / |SIDsX|;
      lift(e) = ([SD] \times |sids(e)|)/(|SIDsX| \times |newSIDsY|);
      if UL(e).iutil ≥ minutil ∧ conf(e) ≥ minconf ∧ lift(e) ≥ minlift then
        CHUSR ← CHUSR \cup e;
      end
      if (UL(e).iutil + UL(e).iutil + UL(e).iutil) ≥ minutil then
        call RightExpansion(e, SD, minutil, minconf, minbond, minlift, SIDsX, newSIDsY, SIDsORX, SIDsORY);
      end
    end
  end
end

Algorithm 3: LeftExpansion(r, SD, minutil, minconf, minbond, minlift, SIDsX, SIDsY, SIDsORX)

Input: r: X ⇒ Y; SD; minutil; minconf; minbond; minlift; SIDsX; SIDsY; SIDsORX.

for sequence S ∈ sids(r) do
  for rule e: X \cup \{i\} ⇒ Y \ni leftRight(r, S) \cup onlyLeft(r, S) do
    x is the last item in X;
y is the last item in Y;
    if ESUCS(i, y) < minutil then
      continue;
    end
    if BondMatrix(x, i) < minbond then
      continue;
    end
    if bond(X \cup i) ≥ minbond then
      newSIDsX = SIDsX \cap sids(i);
      newSIDsORX = SIDsORX \cup sids(i);
      conf(e) = |sids(e)| / |newSIDsX|;
      lift(e) = ([SD] \times |sids(e)|)/(|newSIDsX| \times |SIDsY|);
      if UL(e).iutil ≥ minutil ∧ conf(e) ≥ minconf ∧ lift(e) ≥ minlift then
        CHUSR ← CHUSR \cup e;
      end
      if UL(e).iutil + UL(e).iutil + UL(e).iutil) ≥ minutil then
        call LeftExpansion(e, SD, minutil, minconf, minbond, minlift, newSIDsX, SIDsY, newSIDsORX);
      end
    end
  end
end

the Strategy RightExpansion is called. At the same level, if the sum of utilities except rutil in UL(r) satisfies the Strategy LeftExpansion is called.

Algorithm 2 provides details of the right expansion. The RightExpansion function takes the rules to be extended r: X ⇒ Y, a sequence database SD, the four restraint thresholds minutil, minconf, minbond, and minlift, the collection of sequences containing X SIDsX, the collection of sequences containing Y SIDsY, the collection of sequences containing any item in X SIDsORX, and the collection of sequences containing any item in Y SIDsORY as input. Let r expand with an item i in onlyLeft(r, s), leftRight(r, s) or onlyRight(r, s) by right, x is the last item in X that is the greatest item, and y is the last item in Y. If ESUCS(x, i) < minutil holds, then the rule and its expansions cannot be CHUSRs (according to Strategy 7). If BondMatrix(y, i) < minbond holds, it can also be ignored (according to Strategy 8). If bond(Y \cup i) ≥ minbond holds, the extended rule e is a promising rule (according to Strategy 3). If the rule simultaneously satisfies the utility, confidence, and lift constraints, it is added to the CHUSR collection. As described above, the rule is expanded when the right or left expansion property is satisfied. The LeftExpansion procedure depicted in Algorithm 3 is similar to the RightExpansion procedure. However, the main difference is that the right expansion cannot be performed after the left expansion.

V. EXPERIMENTS

To evaluate the effectiveness and efficiency of the proposed algorithm and strategies, we conduct experiments in terms of runtime and memory consumption, the number of generated
sequential rules, and pruned rules. Note that this is the first study to discover CHUSRs and that the benchmark algorithm is HUSRM. Although the CoUSR algorithm is proposed on the basis of the HUSRM algorithm, we do not intend to use HUSRM as a comparison because CoUSR has added two parameters that are not comparable: minbond and minlift. Additionally, when minbond and minlift are both fixed at zero, the number of rules derived is the same as that of the HUSRM. However, the runtime of HUSRM is often much longer than that of CoUSR when the same minutil and minconf are used, but minbond and minlift differ by tens of thousands of seconds in some cases. Therefore, we do not compare them here.

The efficiency of Strategies 1, 2, 3, 4, and 5 has been verified in this study with respect to the HUSRM algorithm [24] and FCHM [45], and it is not repeated here. In this study, we compare Strategies 6 and 7 focusing on four versions of the CoUSR algorithm (CoUSR, CoUSR$_6$, CoUSR$_7$, CoUSR$_{6.7}$). CoUSR denotes the designed algorithm without Strategies 6 and 7; CoUSR$_6$ and CoUSR$_7$ imply CoUSR with Strategies 6 and 7, respectively, whereas CoUSR$_{6.7}$ incorporates both strategies. For the convenience of plotting and illustration, we denote $\alpha$ for minutil, $\beta$ for minconf, $\gamma$ for minbond, and $\delta$ for minlift. Because there are four restriction parameters in this algorithm, to comprehensively analyze the algorithm, we mainly consider the variations in utility as representatives to evaluate the performance of the algorithm.

### A. Experimental Environment and Datasets

All codes are implemented in the Java language. The experiments were performed on a PC with an Intel(R) Core(TM) i5-8500 CPU @3.00 GHz and 24 GB of RAM, which uses the Windows 10 operating system. Six realistic datasets with different characteristics were applied to ensure the superiority of the proposed algorithm from all aspects (sparse and dense). These algorithms are Bible, BMS, Kosarak10k, Sign, Scalability10K, and Yoochoose, and the specific descriptions of their features can be found in Table IV. $|D|$ and $|I|$ represent the number of sequences and distinct items in a database, respectively. $\text{avg}(D)$ and $\text{avg}(I)$ imply the average length of sequences and itemsets, respectively.

| Dataset   | $|D|$ | $|I|$ | $\text{avg}(D)$ | $\text{avg}(I)$ |
|-----------|------|------|----------------|----------------|
| Bible     | 36,369 | 13,905 | 21.64 | 1 |
| Sign      | 730  | 267  | 51.99 | 1 |
| Kosarak10k| 10,000 | 10,094 | 8.14 | 1 |
| BMS       | 59,601 | 3,340 | 4.62 | 1 |
| Scalability10K | 1,000 | 7,312 | 6.22 | 4.35 |
| Yoochoose | 234,300 | 16,004 | 1.13 | 1.97 |

### B. Runtime Comparison

As is well known, the runtime of algorithms is an important aspect of effectiveness analysis. Therefore, we compare the execution times of the methods under the assumption that $\alpha$ varies, while the other parameters are fixed. Because each dataset has different characteristics, the corresponding parameters differ for each dataset. We obtain the variation in runtime in different contexts when $\alpha$ evolves, as shown in Fig. 6. It is easy to notice that as $\alpha$ increases, the execution time of the progression decreases, as do the gaps between them. Overall, CoUSR takes the longest time without the two pruning strategies, followed by CoUSR$_{6}$ and CoUSR$_7$, and CoUSR$_{6.7}$ takes the shortest running time for both strategies. Additionally, note that CoUSR$_7$ takes less time than CoUSR$_6$. This demonstrates that the correlation-based pruning strategy does not outperform the utility-based pruning strategy. Moreover, note that in most cases, the operation times of CoUSR and CoUSR$_6$ are close, as are the operation times of CoUSR$_7$ and CoUSR$_{6.7}$. For example, in the BMS dataset, as shown in Fig. 6(b), $\beta$ is set to 0.7, $\gamma$ to 0.17, $\delta$ to 5, and $\alpha$ is varied from 1000 to 6000 with 1000 increments. The runtime of CoUSR decreases from 1682s to 1511s, while that of CoUSR$_{6.7}$ decreases from 230s to 197s, which is much less than the former. When the operation times of CoUSR$_6$ and CoUSR$_7$ are compared, CoUSR$_7$ plays a more significant role.

### C. Memory Consumption

As illustrated in Fig. 7, CoUSR$_{6.7}$ consumes the most memory because it is required to construct both SEUSCs and BondMatrix. Because the number of items in the datasets is closely related to the size of these two structures, CoUSR$_{6.7}$ consumes the most memory. However, in Fig. 7(b), CoUSR consumes more memory than CoUSR$_{6.7}$ in the BMS dataset when the utility value is low, possibly because of the large number of utility-list constructions in the initial stage. When comparing the memory consumption of the CoUSR algorithm with only Strategies 6 and 7, CoUSR$_7$ consumes significantly more memory than CoUSR$_6$, in some cases nearly twice as much. The reason for this phenomenon is that SEUSCS occupies significantly more memory than BondMatrix, with the former having a size of $|I|^2$ and the latter having a size of $(|I| - 1)^2$. Although CoUSR does not need to construct either structure, its memory consumption is not the lowest, with CoUSR$_{6}$ consuming the least. For example, as shown in Fig. 7(e) when $\alpha$ is low, the memory consumption of CoUSR and CoUSR$_6$ is similar; however, as $\alpha$ increases, the memory consumption of CoUSR$_6$ decreases, and the difference from CoUSR increases. The reason for this is that the pruning capabilities of CoUSR$_6$ are not outstanding when $\alpha$ is low and require the construction of the BondMatrix structure, which consumes a lot of memory space. However, as $\alpha$ increases, the number of rules that can be pruned by Strategy 6 increases, resulting in the construction of fewer redundant utility-lists, thereby consuming less memory than CoUSR.

### D. Rules and Pruning Strategies Analysis

In this subsection, we analyze the number of CoUSRs generated by the proposed algorithm with or without the designed strategies, as well as the variations of pruned rules. Table VI shows that as the utility threshold increases, the
number of derived rules decreases as the restrictions on the rules tighten, resulting in fewer rules being generated.

Table V shows the variation in the number of rules pruned for different conditions, which provides a clear representation of the effectiveness of the proposed pruning strategies. Pruning rules can entail avoiding the construction of utility-lists that cannot be the CHUSRs. CoUSR_6,7 eliminates the greatest number of ineligible rules because it adopts Strategies 6 and 7. The number of pruned rules decreases with increasing utility for essentially all datasets in Table V, which is because fewer candidate rules are generated; hence, fewer rules are pruned away. Consistent with the runtime and memory consumption results above, CoUSR_7 prunes out most of the non-qualified rules and dominates the pruning process. Note that the number of pruning rules in Scalability10K for CoUSR_6 is increasing. This is because the upper bound on the sequence-estimated utility of the items and rules already filters many candidate rules in the previous series of pruning strategies; however, as the utility increases, fewer rules are pruned by the previous strategies, allowing the number of rules pruned by the later Strategy 6 to increase.
TABLE V
PRUNING RULES NUMBER WITH VARYING minutil

| Bible | 4,000 | 5,000 | 6,000 | 7,000 | 8,000 | 9,000 |
|-------|-------|-------|-------|-------|-------|-------|
| CoUSR _g_ | 119,425,679 | 115,865,502 | 111,955,889 | 107,574,049 | 103,017,057 | 98,966,669 |
| CoUSR _v_ | 317,832,011 | 288,298,290 | 263,979,229 | 243,325,432 | 225,761,570 | 210,762,251 |
| CoUSR _v,v,v_ | 320,028,765 | 290,577,799 | 266,336,522 | 245,730,140 | 223,636,503 | 213,267,557 |
| BMS | 10,000 | 2,000 | 5,000 | 4,000 | 5,000 | 6,000 |
| CoUSR _g_ | 125,983 | 291,731 | 367,902 | 2,427,132 | 4,987,798 | 5,847,064 |
| CoUSR _v_ | 639,670,352 | 631,128,769 | 615,083,532 | 586,086,025 | 574,104,943 | 574,135,650 |
| CoUSR _v,v,v_ | 639,670,927 | 631,129,369 | 615,083,532 | 600,099,905 | 586,086,573 | 584,519,588 |

| Kosarak10k | 21,000 | 21,200 | 21,400 | 21,600 | 21,800 | 22,000 |
|------------|-------|-------|-------|-------|-------|-------|
| CoUSR _g_ | 844,097,317 | 393,824,642 | 179,873,521 | 102,417,155 | 61,701,758 | 56,130,360 |
| CoUSR _v_ | 439,150,757 | 183,304,554 | 99,371,882 | 58,762,678 | 36,407,977 | 23,615,658 |
| CoUSR _v,v,v_ | 837,605,098 | 342,069,760 | 182,541,259 | 105,004,783 | 64,209,099 | 58,519,588 |

| Sign | 23,000 | 24,000 | 25,000 | 26,000 | 27,000 | 28,000 |
|-------|-------|-------|-------|-------|-------|-------|
| CoUSR _g_ | 73,372,239 | 70,781,542 | 69,410,641 | 65,179,889 | 60,999,879 | 56,130,360 |
| CoUSR _v_ | 813,582,223 | 756,826,592 | 700,969,676 | 652,646,596 | 606,221,986 | 569,154,898 |
| CoUSR _v,v,v_ | 816,227,798 | 759,384,140 | 703,545,572 | 655,073,816 | 608,539,612 | 571,474,179 |

| Scalability10k | 100 | 200 | 300 | 400 | 500 | 600 |
|---------------|-----|-----|-----|-----|-----|-----|
| CoUSR _g_ | 465,313 | 11,051,082 | 42,259,333 | 66,508,272 | 73,546,959 | 75,581,955 |
| CoUSR _v_ | 523,857,749 | 342,171,500 | 232,033,718 | 172,937,045 | 139,872,875 | 117,868,255 |
| CoUSR _v,v,v_ | 586,858,286 | 342,310,284 | 232,335,631 | 173,293,302 | 140,192,238 | 118,134,902 |

| Yoochoose | 4,000 | 5,000 | 6,000 | 7,000 | 8,000 | 9,000 |
|-----------|-------|-------|-------|-------|-------|-------|
| CoUSR _g_ | 4,748 | 4,072 | 3,389 | 3,030 | 2,633 | 2,166 |
| CoUSR _v_ | 480,310 | 461,787 | 263,237 | 56,161 | 8,957 | 4,591 |
| CoUSR _v,v,v_ | 481,479 | 462,934 | 264,242 | 57,167 | 9,805 | 5,354 |

VI. CONCLUSION AND FUTURE STUDIES

In this study, we incorporate the concept of correlation into HUSR and propose a novel algorithm called CoUSR. We introduce two parameters, bond and lift, to limit the derived rules locally and globally. To consolidate important information in sequence databases and prevent multiple scans of the given database, utility-lists are proposed. Furthermore, we introduce several pruning strategies to improve the efficiency of the algorithm using the designed BondMatrix and ESUCS structures. Subsequent experiments on realistic datasets show that these strategies are indeed functional, and that the proposed algorithm performs well. In the future, we intend to broaden the application environment of this algorithm to include uncertain data or dynamic circumstances. Furthermore, owing to the problem of difficult threshold selection, it is prudent to design a top-k CoUSR algorithm. Of course, performance improvement is another critical consideration.

REFERENCES

[1] A. Feelders, “Data mining in economic science,” *Dealing with the Data Flood*, pp. 166–175, 2002.
[2] M. S. Chen, J. Han, and P. S. Yu, “Data mining: an overview from a database perspective,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 8, no. 6, pp. 866–883, 1996.
[3] R. Agrawal, R. Srikant et al., “Fast algorithms for mining association rules,” in *International Conference on Very Large Data Bases*. Citeseer, 1994, pp. 487–499.
[4] J. Han, J. Pei, and Y. Yin, “Mining frequent patterns without candidate generation,” *ACM SIGMOD Record*, vol. 29, no. 2, pp. 1–12, 2000.
[5] P. Fourrier-Viger, J. C. W. Lin, R. U. Kian, Y. S. Koh, and R. Thomas, “A survey of sequential pattern mining: Data Science and Pattern Recognition, vol. 1, no. 1, pp. 54–77, 2017.
[6] W. Gan, J. C. W. Lin, P. Fourrier-Viger, H. C. Chao, and P. S. Yu, “A survey of parallel sequential pattern mining,” *ACM Transactions on Knowledge Discovery from Data*, vol. 13, no. 3, pp. 1–34, 2019.
[7] J. Han, J. Pei, B. Mortazavi-Asl, Q. Chen, U. Dayal, and M. C. Hsu, “FreeSpan: frequent pattern-projected sequential pattern mining,” in *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 355–359.
[8] Y. Wu, C. Zhu, Y. Li, L. Guo, and X. Wu, “NetNCSF: Non-overlapping closed sequential pattern mining,” *Knowledge-based systems*, vol. 196, p. 105812, 2020.
[9] R. Srikant and R. Agrawal, “Mining sequential patterns: generalizations and performance improvements,” in *International Conference on Extending Database Technology*. Springer, 1996, pp. 1–17.
[10] P. Fourrier-Viger, T. Gueniche, S. Zida, and V. S. Tseng, “ERMiner: sequential rule mining using equivalence classes,” in *International Symposium on Intelligent Data Analysis*. Springer, 2014, pp. 108–119.
[11] P. Fourrier-Viger, U. Faghri, R. Nkambou, and E. M. Nguifo, “CMRules: Mining sequential rules common to several sequences,” *Knowledge-Based Systems*, vol. 29, no. 2, pp. 1–12, 2000.
[12] H. Mannila, H. Toivonen, and A. I. Verkamo, “Discovery of frequent episodes in event sequences,” *Data Mining and Knowledge Discovery*, vol. 1, no. 3, pp. 259–289, 1997.
[13] J. H. Hamilton and K. Karimi, “The TIMERS ii algorithm for the discovery of causality,” in *Pacific-Asia Conference on Knowledge Discovery and Data Mining*. Springer, 2005, pp. 744–750.
[14] S. K. Harms, J. Deogun, and T. Tadesse, “Discovering sequential association rules with constraints and time lags in multiple sequences,” in *International Symposium on Methodologies for Intelligent Systems*. Springer, 2002, pp. 432–441.
