Computers, Materials & Continua

Tech Science Press

https://doi.org/10.32604/cmc.2026.077790

ARTICLE

Mining High-Quantitative Periodic Frequent Patterns across Multiple Sequences

Yan Ge[1], Zhenzhou Zhang[2] and Chien-Ming Chen[3,*]

[1]Reading Academy, Nanjing University of Information Science and Technology, Nanjing, China
[2]China Unicom (Shandong) Industrial Internet Co., Ltd., Jinan, China
[3]School of Artificial Intelligence, Nanjing University of Information Science and Technology, Nanjing, China

*Corresponding Author: Chien-Ming Chen. Email: chienmingchen@ieee.org

Received: 17 December 2025; Accepted: 17 April 2026; Published: 15 June 2026

ABSTRACT: Periodic pattern mining plays an important role in revealing recurring behavioral regularities from temporal sequence data. Most existing approaches, however, are developed for single-sequence settings and rarely account for quantitative information or sequence-level constraints when patterns recur across multiple sequences. This limits their usefulness in practical scenarios, where a pattern is expected to be not only periodic but also quantitatively significant in a sufficiently large portion of sequences. In this work, we formulate the problem of mining High-Quantitative Periodic Frequent Patterns (HQPFPS) from multi-sequence databases and propose an efficient algorithm, termed MHQPFPS. The proposed method evaluates pattern significance through a quantitative ratio within each sequence and exploits a sequence-level upper bound to effectively prune unpromising candidates during pattern growth. To support efficient evaluation, a compact list-based structure is introduced to maintain support, periodicity, and quantitative statistics, thereby avoiding repeated scans of the database. These components are combined within a depth-first exploration framework to systematically generate valid patterns while discarding those that fail to satisfy the required periodic or quantitative constraints. Experimental results on both real-world and synthetic datasets show that MHQPFPS is able to extract meaningful high-quantitative periodic patterns across multiple sequences. Moreover, the results indicate that the proposed pruning strategies substantially reduce computational cost in terms of runtime and memory consumption under a wide range of parameter settings.

KEYWORDS: Data mining; high-quantitative periodic patterns; multi-sequence databases; quantitative pattern mining

1 Introduction

Frequent pattern mining (FPM) has long been regarded as a core task in data mining [1,2]. Its original objective is to identify itemsets that occur frequently in transactional databases under a minimum support threshold [3,4]. Over the years, FPM and its variants have been applied across a wide range of domains, including, text mining [5], network flow analysis [6,7], biomedical data analysis [8], malware detection [9,10], and smart city applications [11]. Despite its success, classical FPM treats transactions as unordered collections of items and therefore ignores the temporal order in which events occur. This assumption becomes problematic in applications where the timing and sequence of events carry important semantic meaning.

Sequential pattern mining (SPM) was introduced to address this limitation by explicitly modeling temporal order [12]. By discovering frequent subsequences in ordered transaction data, SPM has found practical use in customer behavior analysis, recommendation systems, and intelligent tutoring environments [13]. Nevertheless, SPM still focuses mainly on frequency and ordering. It does not explicitly consider

whether patterns recur in a regular or near-regular manner over time. In many real-world scenarios, however, such temporal regularity is a defining characteristic. Periodic purchasing behaviors or recurring symptom combinations in clinical records are typical examples. Identifying these periodic patterns is often critical for tasks such as marketing analysis, resource planning, and anomaly detection.

To capture temporal regularity, periodic frequent pattern mining (PFPM) has been proposed to identify patterns that recur within bounded time intervals in sequences or transactional databases [14,15]. Most existing PFPM approaches are designed for single-sequence settings, where periodicity is evaluated within one long sequence. This design choice is increasingly restrictive in modern data environments. Internet of Things (IoT) systems and other multi-source platforms generate large collections of sequences, making it necessary to identify periodic patterns that recur consistently across multiple sequences [16,17]. Although several recent studies have begun to address this setting, they typically either ignore the quantitative information associated with item occurrences or fail to identify patterns that are both periodic and quantitatively significant across a large number of sequences.

In many practical applications, quantitative information [18,19] is not merely supplementary, but directly affects how periodic patterns should be interpreted. For example, in retail analytics, consider two items A and B that are both purchased weekly across 80% of customer sequences, thus satisfying the same periodicity and support thresholds. A traditional periodic frequent pattern mining method would treat A and B as equally important. However, if A is purchased only in negligible quantities per transaction while B consistently accounts for a large proportion of each transaction's total value, the two patterns imply different operational priorities. Ignoring this difference may lead to biased decisions in inventory planning and marketing allocation. Similarly, in sensor networks and event logs, patterns with higher quantitative contributions often reflect dominant or critical system behaviors. Despite this, existing periodic pattern mining methods rarely integrate quantitative measures with multi-sequence periodicity constraints. This gap limits their effectiveness in addressing real-world analysis tasks.

Although some existing methods incorporate utility-based evaluation [17,19], they rely on absolute utility values that are not normalized with respect to the total quantitative content of each sequence. As a result, it becomes difficult to compare pattern significance across sequences of different lengths or value scales, and the utility threshold may become less consistent in heterogeneous databases. The present work addresses this limitation by introducing a normalized quantitative ratio into the multi-sequence periodic pattern mining framework, so that periodicity, quantitative contribution, and cross-sequence consistency can be considered jointly.

Motivated by these observations, this paper studies the problem of discovering *High-Quantitative Periodic Frequent Patterns* (HQPFPS) across multiple sequences. The proposed formulation jointly considers periodicity constraints and quantitative contributions and requires that a pattern exhibit both regular recurrence and sufficient quantitative significance in a substantial portion of the sequence database. To efficiently mine such patterns, we propose the MHQPFPS algorithm. The algorithm introduces a quantitative ratio to evaluate pattern contributions within individual sequences, derives a sequence-level upper bound to prune unpromising candidates during pattern growth, and employs a compact HQPFPS-list structure to avoid repeated scans of the original database.

The main contributions of this paper can be summarized as follows:

- We formulate the problem of mining High-Quantitative Periodic Frequent Patterns (HQPFPS) across multiple sequences by jointly considering periodicity constraints and quantitative ratios at both the sequence and database levels.
- We propose the MHQPFPS algorithm, which combines a quantitative ratio measure, a sequence-level upper-bound pruning strategy, and a unified list-based structure that maintains support, periodicity, and quantitative statistics, enabling efficient depth-first exploration of the search space.
- We evaluate the proposed approach on real-world and synthetic datasets to analyze the impact of key parameters on runtime, the number of discovered patterns, and memory usage. The results confirm that MHQPFPS can effectively extract meaningful high-quantitative periodic frequent patterns under various parameter configurations.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 defines the HQPFPS problem. Section 4 presents the MHQPFPS algorithm. Section 5 reports the experimental evaluation. Section 6 concludes the paper and discusses future research directions.

2 Related Work

Pattern mining has been studied from multiple perspectives over the past decades, resulting in several research directions that are closely related to the problem considered in this paper. In particular, our work intersects with studies on periodic pattern mining, quantitative and high-utility pattern mining, and pattern discovery across multiple sequences. Existing surveys on frequent and sequential pattern mining provide a broad background for these topics [1–3,5]. In the following, we briefly review the most relevant studies and explain how they relate to the proposed HQPFPS framework.

2.1 Periodic Pattern Mining

Periodic pattern mining aims to identify patterns that recur at regular or near-regular intervals in temporal data. Early work introduced the concept of periodic frequent patterns and proposed pruning-based strategies to efficiently discover such patterns in transactional databases [14]. Subsequent studies demonstrated the usefulness of periodic pattern mining in real-world temporal applications, including public transportation analysis and urban mobility modeling [11,15].

While these approaches effectively capture temporal regularity, they are largely designed for single-sequence settings. In most cases, periodicity is evaluated within one long sequence, and the question of whether a pattern exhibits consistent periodic behavior across multiple sequences is not explicitly addressed. Moreover, periodic pattern mining is commonly based on binary item occurrence, without considering quantitative information associated with items. Even studies that extend periodic analysis to multiple sequences [16] typically focus on temporal aspects alone and do not incorporate quantitative constraints. This limits their ability to support applications where both periodicity and quantitative significance are important.

2.2 Quantitative and High-Utility Pattern Mining

Another relevant line of research focuses on mining patterns based on quantitative or utility-based criteria. Early work on quantitative association rules established mechanisms for evaluating item importance using numerical values rather than frequency alone [18]. This direction later evolved into weighted and high-utility pattern mining, enabling the discovery of patterns that reflect contribution or utility instead of pure occurrence counts [19]. More recently, high-utility sequential pattern mining has been explored in uncertain or dynamic environments, illustrating the flexibility of quantitative evaluation in sequential data analysis [13].

Despite their effectiveness, quantitative and high-utility approaches typically do not consider temporal periodicity. Patterns are evaluated mainly from the perspective of contribution or importance, and sequences are often processed independently. As a result, these methods cannot ensure that a pattern is both quantitatively significant and periodically recurrent across multiple sequences, which is a key requirement in many real-world scenarios.

2.3 Multi-Sequence and Cross-Sequence Pattern Mining

Pattern discovery across multiple sequences has also received increasing attention. Methods in this category aim to identify patterns that occur in a sufficient number of sequences, thereby capturing population-level or system-level behaviors. Representative examples include studies on correlated periodic patterns in multiple sequences [16] and approaches that combine periodicity with utility measures in multi-sequence settings [17]. These works move beyond single-sequence analysis and highlight the importance of cross-sequence consistency.

Despite these advances, the proposed MHQPFPS algorithm differs from the two most closely related methods, MRCPPS [16] and MHUPFPS [17], in several important aspects. MRCPPS focuses on correlated periodic pattern mining over multiple sequences, but it does not take into account the quantitative contribution of patterns, since pattern significance is evaluated mainly from occurrence information and correlation. MHUPFPS extends periodic pattern mining to the high-utility setting, but it uses absolute utility values rather than a normalized measure. Therefore, pattern significance may be difficult to compare directly across sequences with different lengths or quantitative scales. In contrast, MHQPFPS evaluates each pattern by a normalized quantitative ratio bounded in $[0,1]$, which provides a more comparable measure across heterogeneous sequences. A second difference lies in the treatment of periodicity regularity. While MRCPPS considers a standard-deviation-related constraint in the rare correlated periodic pattern setting, and MHUPFPS uses maximum periodicity bounds, neither method combines period stability control with normalized quantitative evaluation in the multi-sequence frequent pattern setting considered here. MHQPFPS introduces the period standard deviation threshold maxStd to further distinguish patterns with relatively stable periodic behavior from those that only satisfy the maximum period requirement. A third difference is related to the pruning strategy. MHQPFPS derives a sequence-level upper bound for the cross-sequence quantitative ratio of pattern extensions. This upper bound supports safe pruning during the depth-first search process and helps improve mining efficiency. While MRCPPS also employs a sequence-level upper bound (termed upBondRa) based on support and periodicity conditions, its upper bound additionally incorporates the bond correlation constraint and does not extend to quantitative measures, since MRCPPS does not incorporate quantitative evaluation. MHUPFPS does not report a comparable upper-bound pruning strategy.

Overall, these differences indicate that existing multi-sequence studies only partially cover the problem addressed in this work, which motivates the summary and gap analysis presented next.

2.4 Summary and Gap Analysis

Table 1 summarizes the main distinctions between MHQPFPS and the most closely related methods from five aspects: support for multiple sequences, periodicity consideration, quantitative measure, period stability constraint, and upper-bound-based pruning.

Table 1: Comparison of MHQPFPS with closely related methods.

Algorithm	Multi-Seq	Periodicity	Quantitative	Std Constraint	Upper Bound
PFP [14]	×	✓	×	×	×
MRCPPS [16]	✓	✓	×	✓	Partial
MHUPFPS [17]	✓	✓	Absolute utility	×	×
MHQPFPS	✓	✓	Normalized ratio	✓	✓

Note: ✓ = supported; × = not supported. For the quantitative dimension, "absolute utility" denotes evaluation based on raw utility values, whereas "normalized ratio" denotes the proposed quRa measure. For the standard-deviation dimension, MRCPPS includes a related constraint in the rare-pattern setting, but it does not incorporate quantitative evaluation. For the upper-bound dimension, MRCPPS employs a sequence-level upper bound (upBondRa) that incorporates support, periodicity, and bond constraints, but does not extend to quantitative measures; it is therefore marked as "partial".

As shown in Table 1, existing studies cover only part of the problem addressed in this work. Some methods model periodicity, whereas others emphasize quantitative importance or cross-sequence behavior. However, these aspects are usually considered separately rather than within a unified mining framework. As a result, prior studies do not jointly address periodic recurrence within each sequence, quantitative contribution in a normalized form, and consistency across multiple sequences in the frequent-pattern setting considered here.

These observations motivate the present study. Accordingly, this work formulates the problem of discovering High-Quantitative Periodic Frequent Patterns across multiple sequences, where periodicity, quantitative contribution, and cross-sequence consistency are considered jointly. To solve this problem efficiently, we further develop an algorithmic framework based on HQPFPS-list and upper-bound-guided pruning.

3 Definitions and Problem Statement

This section introduces the notation and fundamental concepts used throughout this paper and formally defines the problem of mining High-Quantitative Periodic Frequent Patterns (HQPFPS) from a multi-sequence database. Table 2 summarizes the main symbols, including computed quantities, structural components, and user-specified thresholds, used throughout this paper.

Table 2: Notation summary.

Symbol	Description
Basic structures	
$\mathcal{I}$	Set of all items
X, Y	Itemsets (subsets of $\mathcal{I}$)
S	A sequence $S = \langle T_1, T_2, \ldots, T_m \rangle$
D	Sequence database $D = \{S_1, S_2, \ldots, S_n\}$
$\|S\|$	Length of sequence S (number of transactions)
$\|D\|$	Number of sequences in the database
T_j	j-th transaction in a sequence (1-based indexing)
Support and occurrence	
$\mathrm{TR}(X, S)$	Ordered list of transaction positions where X occurs
$\sup(X, S)$	Support count of X in S

(Continued)

Table 2 (continued)

Symbol	Description
$\mathrm{supRa}(X, S)$	Support ratio = $\sup(X, S)/\lvert S\rvert$
Periodicity	
per_z	z-th period of X in S ($z = 0, 1, \ldots, k$)
$\mathrm{maxPer}(X, S)$	Maximum period of X in S
μ	Mean period = $\frac{1}{k+1}\sum_{z=0}^{k}\mathrm{per}_z$
$\mathrm{stanDev}(X, S)$	Standard deviation of periods of X in S
Quantitative measures	
$q(i, T_j, S)$	Quantitative value of item i in transaction T_j of S
$q(X, T_j, S)$	Quantitative contribution of X in T_j, = $\sum_{i\in X} q(i, T_j, S)$
$q(T_j, S)$	Total quantitative value of transaction T_j, = $\sum_{i\in T_j} q(i, T_j, S)$
$q(S)$	Total quantitative value of sequence S, = $\sum_{j=1}^{\lvert S\rvert} q(T_j, S)$
$Q_X(S)$	Total quantitative contribution of X in S
$\mathrm{quRa}(X, S)$	Quantitative ratio of X in S, = $Q_X(S)/q(S)$
Cross-sequence measures	
$\mathrm{hqPrSeq}(X)$	Set of sequences where X is an HQPFP
$\mathrm{hqSeqRa}(X)$	High-quantitative periodic sequence ratio
$\mathrm{candSeq}(X)$	Candidate sequence set of pattern X
$\mathrm{upRa}(X)$	Upper bound of hqSeqRa for pattern X
Algorithm structures	
$\mathrm{PFL}(X)$	HQPFPS-list structure for pattern X
$\mathrm{last}(P)$	The lexicographically largest item in pattern P
User-specified thresholds	
minSupRa	Minimum support ratio threshold
maxPr	Maximum period threshold
maxStd	Maximum period standard deviation threshold
minHqRa	Minimum quantitative ratio threshold
minSeqRa	Minimum sequence ratio threshold

Definition 1 (Basic Concepts): *Let I be a finite set of items. An itemset $X \subseteq I$ is a subset of items. If X contains k distinct items $\{i_1, i_2, \ldots, i_k\}$, it is referred to as a k-itemset. A sequence $S = \langle T_1, T_2, \ldots, T_m\rangle$ is an ordered list of transactions, where each transaction $T_j \subseteq I (1 \le j \le m)$. A sequence database is denoted by $D = \{S_1, S_2, \ldots, S_n\}$. An itemset X is said to occur in a transaction T_j if $X \subseteq T_j$.*

Definition 2 (Occurrences and Support): *For an itemset X in a sequence S, let*

$$\mathrm{TR}(X, S) = \langle j_1, j_2, \ldots, j_k\rangle$$

denote the ordered list of 1-based transaction indices at which X occurs. If X does not occur in S, then $\mathrm{TR}(X, S)$ is empty and $\sup(X, S) = 0$. The support count of X in S is defined as

$$\sup(X, S) = \lvert\mathrm{TR}(X, S)\rvert.$$

Definition 3 (Support Ratio): *The support ratio of an itemset X in a sequence S is defined as*

$$\mathrm{supRa}(X, S) = \frac{\sup(X, S)}{|S|},$$

where $|S|$ denotes the length of sequence S, i.e., the number of transactions it contains. This normalized measure allows support values to be compared across sequences of different lengths.

Definition 4 (Period and Maximum Period): *Let $\mathrm{TR}(X, S) = \langle j_1, j_2, \ldots, j_k \rangle$ be the ordered list of transaction positions at which itemset X occurs in sequence S. The periods of X in S are defined as:*

- *Initial boundary period:* $\mathrm{per}_0 = j_1 - 1$,
- *Inter-occurrence periods:* $\mathrm{per}_z = j_{z+1} - j_z$, *for* $1 \le z \le k - 1$,
- *Final boundary period:* $\mathrm{per}_k = |S| - j_k$.

If $\sup(X, S) = 0$, the periods are undefined and $\mathrm{maxPer}(X, S)$ is set to ∞. Otherwise, the maximum period of X in S is defined as

$$\mathrm{maxPer}(X, S) = \max_{0 \le z \le k} \mathrm{per}_z.$$

Definition 5 (Period Standard Deviation): *Given the set of periods $\{\mathrm{per}_0, \mathrm{per}_1, \ldots, \mathrm{per}_k\}$ of X in a sequence S, the period standard deviation is defined as*

$$\mathrm{stanDev}(X, S) = \sqrt{\frac{1}{k+1} \sum_{z=0}^{k} (\mathrm{per}_z - \mu)^2},$$

where $\mu = \frac{1}{k+1} \sum_{z=0}^{k} \mathrm{per}_z$ is the mean period. If $\sup(X, S) = 0$, the periods are undefined and X cannot satisfy any HQPFP constraint in S.

Definition 6 (Quantitative Values): *Each item $i \in T_j$ is associated with a positive quantitative value $q(i, T_j, S)$. The quantitative value of a transaction T_j is defined as*

$$q(T_j, S) = \sum_{i \in T_j} q(i, T_j, S),$$

and the quantitative value of a sequence S is defined as

$$q(S) = \sum_{j=1}^{|S|} q(T_j, S).$$

Definition 7 (Quantitative Ratio): *The quantitative contribution of an itemset X in a transaction T_j is defined as*

$$q(X, T_j, S) = \sum_{i \in X} q(i, T_j, S).$$

The total quantitative contribution of X across all its occurrences in sequence S is

$$Q_X(S) = \sum_{j \in \mathrm{TR}(X,S)} q(X, T_j, S).$$

The quantitative ratio of X in S is then defined as

$$\mathrm{quRa}(X,S) = \frac{Q_X(S)}{q(S)}.$$

For two disjoint itemsets X and Y with $X \cap Y = \varnothing$, the quantitative contributions satisfy the following additive properties:

1. *Transaction level:*

$$q(X \cup Y, T_j, S) = q(X, T_j, S) + q(Y, T_j, S).$$

2. *Sequence level:*

$$Q_{X\cup Y}(S) = \sum_{j\in \mathrm{TR}(X\cup Y,S)} \big(q(X, T_j, S) + q(Y, T_j, S)\big).$$

Note that $\mathrm{TR}(X \cup Y, S) \subseteq \mathrm{TR}(X, S) \cap \mathrm{TR}(Y, S)$, since the union pattern can occur only when both X and Y appear in the same transaction.

It is worth noting that the proposed quantitative ratio is not intended to redefine the concept of utility, but rather to provide a normalized quantitative measure tailored to multi-sequence periodic pattern mining. In contrast to traditional utility metrics based on absolute values, $\mathrm{quRa}(X, S)$ normalizes the quantitative contribution of a pattern with respect to the total quantity of the sequence. This design improves comparability across sequences with different lengths or quantitative scales, and allows quantitative evaluation to be integrated more naturally with periodicity and cross-sequence consistency in a unified framework.

The choice of $q(S)$ as the denominator in $\mathrm{quRa}(X, S)$ is motivated by three considerations.

(1) Bounded range. Since the total contribution of any pattern in a sequence cannot exceed the total quantitative value of that sequence, we have $Q_X(S) \le q(S)$. Therefore, $\mathrm{quRa}(X, S) \in [0, 1]$. This gives $\mathrm{quRa}(X, S)$ a clear normalized interpretation as the fraction of the total quantitative mass of S that is attributable to pattern X.

(2) Cross-sequence comparability. Normalizing by $q(S)$ reduces the dependence of the measure on sequence length and on the absolute scale of quantitative values. Alternative denominators, such as the maximum transaction quantity $\max_j q(T_j, S)$ or an average per-occurrence quantity, may overemphasize isolated high-value transactions or short sequences, which makes comparison across heterogeneous sequences less reliable. By contrast, $q(S)$ reflects the cumulative quantitative content of the entire sequence, thereby providing a more stable basis for cross-sequence comparison.

(3) Consistency with upper-bound analysis. As shown in Definition 7, the quantitative contribution satisfies the additive property $q(X \cup Y, T_j, S) = q(X, T_j, S) + q(Y, T_j, S)$ at the transaction level. Using the fixed sequence-level quantity $q(S)$ as the denominator preserves this normalization across pattern extensions and facilitates the derivation of a valid upper bound in Section 4.2. This is important for maintaining the monotonic relationship needed for safe pruning during the search process.

Definition 8 (HQPFP in a Single Sequence): *Given user-specified thresholds $minSupRa$, $maxPr$, $maxStd$, and $minHqRa$, an itemset X is called a High-Quantitative Periodic Frequent Pattern (HQPFP) in a sequence S if all of the following conditions are satisfied:*

1. $\mathrm{supRa}(X, S) \ge minSupRa$,
2. $\mathrm{maxPer}(X, S) \le maxPr$,
3. $\mathrm{stanDev}(X, S) \le maxStd$,

4. $\text{quRa}(X, S) \geq minHqRa$.

Let

$$hqPrSeq(X) = \{S \in D \mid X \text{ is an HQPFP in } S\}$$

denote the set of sequences in which X satisfies all HQPFP constraints.

Definition 9 (HQPFPS across Multiple Sequences): *The high-quantitative periodic sequence ratio of a pattern X in a database D is defined as*

$$hqSeqRa(X) = \frac{|hqPrSeq(X)|}{|D|}.$$

Given a threshold $minSeqRa$, X is called a High-Quantitative Periodic Frequent Pattern across the database (HQPFPS) if

$$hqSeqRa(X) \geq minSeqRa.$$

Problem 1: Given a sequence database D and user-specified thresholds $minSupRa$, $maxPr$, $maxStd$, $minHqRa$, and $minSeqRa$, the objective is to discover all itemsets X such that X satisfies the HQPFP constraints in the sequences where it appears and, in addition,

$$hqSeqRa(X) \geq minSeqRa.$$

The collection of all such patterns constitutes the complete set of HQPFPS in D.

Example 1: Fig. 1 illustrates a sequence database consisting of four sequences S_1–S_4. Given the thresholds

$$maxPr = 3, \quad minSupRa = 0.6, \quad minSeqRa = 0.35, \quad maxStd = 1, \quad minHqRa = 0.2,$$

we consider the pattern $X = \{a, d\}$.

Sid	Sequence
S_1	$\langle(a{:}4,\ b{:}6,\ c{:}10),\ (b{:}7,\ c{:}9,\ d{:}10),\ (a{:}6,\ b{:}5),\ (a{:}9,\ b{:}4),\ (a{:}5,\ b{:}6,\ c{:}7,\ d{:}8)\rangle$
S_2	$\langle(a{:}7,\ d{:}10),\ (a{:}5,\ b{:}8,\ c{:}3,\ d{:}4),\ (a{:}6,\ c{:}15,\ d{:}8),\ (a{:}9,\ b{:}9,\ d{:}15),\ (a{:}10,\ b{:}6,\ c{:}16,\ e{:}10)\rangle$
S_3	$\langle(b{:}7,\ d{:}10),\ (a{:}8,\ d{:}4),\ (a{:}5,\ c{:}15,\ d{:}12),\ (b{:}3,\ d{:}12,\ e{:}3),\ (a{:}9,\ b{:}11,\ d{:}12)\rangle$
S_4	$\langle(a{:}7,\ c{:}10,\ d{:}16),\ (a{:}6,\ b{:}2,\ d{:}8),\ (a{:}9,\ c{:}8,\ d{:}6),\ (b{:}2,\ d{:}9),\ (b{:}5,\ d{:}8)\rangle$

Figure 1: An example illustrating HQPFP and HQPFPS evaluation.

In sequence S_2, pattern X appears in transactions 1–4, i.e., $\text{TR}(X, S_2) = \langle 1, 2, 3, 4\rangle$, yielding $\text{supRa}(X, S_2) = 4/5$. All periods are no greater than 1, and both the period standard deviation and quantitative ratio satisfy the corresponding thresholds. Therefore, X is an HQPFP in S_2.

In S_3, X occurs at transactions 2, 3, and 5, giving $\text{TR}(X, S_3) = \langle 2, 3, 5\rangle$ and $\text{supRa}(X, S_3) = 3/5$. The maximum period equals 2, and the remaining constraints are also satisfied. Hence, X is an HQPFP in S_3.

Similarly, in S_4, X appears in transactions 1–3, resulting in $\text{supRa}(X, S_4) = 3/5$ and $\text{maxPer}(X, S_4) = 2$, while both the period standard deviation and quantitative ratio meet the required thresholds. As a result, X is an HQPFP in S_4.

Consequently,

$$hqPrSeq(X) = \{S_2, S_3, S_4\}, \quad hqSeqRa(X) = \frac{3}{4}.$$

Since $hqSeqRa(X) \geq minSeqRa$, pattern $\{a, d\}$ is identified as an HQPFPS in the database.

4 Proposed Algorithm MHQPFPS

High-Quantitative Periodic Frequent Patterns across Sequences, abbreviated as HQPFPS, refer to patterns that jointly satisfy support-ratio, periodicity, period-stability, quantitative-contribution, and cross-sequence-consistency constraints. Efficiently mining HQPFPS from a multi-sequence database therefore requires a carefully designed search strategy that combines effective pruning with compact data representation. In this section, we present the MHQPFPS algorithm, which integrates a depth-first pattern-growth framework, an upper-bound pruning strategy derived from anti-monotonic properties, and an HQPFPS-list structure that maintains detailed occurrence and quantitative information for candidate patterns across sequences.

4.1 Overview of the Mining Strategy

MHQPFPS follows a depth-first pattern-growth strategy, similar to classical prefix-based mining frameworks. The mining process starts from all single-item patterns and recursively extends a current prefix pattern P by appending items with lexicographically larger identifiers. For a pattern $P = \{i_1, i_2, \ldots, i_k\}$ with items ordered lexicographically such that $i_1 < i_2 < \cdots < i_k$, we denote by $\text{last}(P)$ the largest item i_k. The extension of P therefore considers only items i satisfying $i > \text{last}(P)$, ensuring that each itemset is enumerated exactly once. For each extended pattern $P \cup \{i\}$, an HQPFPS-list is constructed to efficiently compute all required statistics of the pattern in each sequence of the database.

A key observation underlying the pruning strategy is the anti-monotonicity of the support ratio defined in Section 3. Specifically, for any two patterns X and Y such that $Y \supseteq X$, and for any sequence S, the following property holds:

$$\text{supRa}(Y, S) \leq \text{supRa}(X, S).$$

This property implies that, as a pattern is extended, its ability to satisfy support-related and periodicity-related constraints cannot improve. Based on this observation, an upper bound on the achievable cross-sequence ratio of any pattern extension can be derived, which enables the algorithm to prune unpromising branches early and significantly reduce the search space.

4.2 Upper-Bound Pruning Based on Anti-Monotonicity

Definition 10 (Candidate Sequence Set): *For a pattern X, its candidate-supporting sequence set is defined as*

$$\text{candSeq}(X) = \{S \in D \mid \text{supRa}(X, S) \geq \text{minSupRa} \ \wedge \ \text{maxPer}(X, S) \leq \text{maxPr}\}.$$

Remark: Only $\text{supRa}(X, S)$ and $\text{maxPer}(X, S)$ are included in $\text{candSeq}(X)$, since both measures satisfy anti-monotonicity with respect to pattern extension. In contrast, $\text{quRa}(X, S)$ and $\text{stanDev}(X, S)$ are not anti-monotonic and therefore cannot be used to derive valid upper bounds for pruning. This will be described later in the pruning procedure

Definition 11 (Upper-Bound Sequence Ratio): *The upper bound on the high-quantitative periodic sequence ratio achievable by a pattern X is defined as*

$$\mathrm{upRa}(X) = \frac{|\mathrm{candSeq}(X)|}{|D|}.$$

Lemma 1 (Anti-Monotonicity Properties): *For any two patterns $X \subseteq Y$ and any sequence S, the following properties hold:*

1. $\mathrm{supRa}(Y, S) \leq \mathrm{supRa}(X, S)$;
2. $\mathrm{maxPer}(Y, S) \geq \mathrm{maxPer}(X, S)$.

Proof: (1) Since $Y \supseteq X$, every occurrence of Y necessarily contains X, which implies $\mathrm{TR}(Y, S) \subseteq \mathrm{TR}(X, S)$. Therefore,

$$\mathrm{supRa}(Y, S) = \frac{|\mathrm{TR}(Y, S)|}{|S|} \leq \frac{|\mathrm{TR}(X, S)|}{|S|} = \mathrm{supRa}(X, S).$$

(2) As $\mathrm{TR}(Y, S)$ is a subsequence of $\mathrm{TR}(X, S)$, the distances between consecutive occurrences of Y cannot be smaller than those of X. Consequently,

$$\mathrm{maxPer}(Y, S) \geq \mathrm{maxPer}(X, S). \qquad \square$$

Theorem 1 (Safe Upper Bound): *For any pattern extension $Y \supseteq X$, the following inequality holds:*

$$\mathrm{hqSeqRa}(Y) \leq \mathrm{upRa}(X),$$

where $\mathrm{hqSeqRa}(\cdot)$ *is defined in Section 3.*

Proof: By Lemma 1, for any sequence $S \in D$,

$$\mathrm{supRa}(Y, S) \leq \mathrm{supRa}(X, S) \quad \text{and} \quad \mathrm{maxPer}(Y, S) \geq \mathrm{maxPer}(X, S).$$

Hence, any sequence that can satisfy both $\mathrm{supRa}(Y, S) \geq \mathrm{minSupRa}$ and $\mathrm{maxPer}(Y, S) \leq \mathrm{maxPr}$ must already belong to $\mathrm{candSeq}(X)$. It follows that

$$\mathrm{hqPrSeq}(Y) \subseteq \mathrm{candSeq}(X).$$

Dividing both sides by $|D|$ yields

$$\mathrm{hqSeqRa}(Y) = \frac{|\mathrm{hqPrSeq}(Y)|}{|D|} \leq \frac{|\mathrm{candSeq}(X)|}{|D|} = \mathrm{upRa}(X). \qquad \square$$

Corollary 1 (Pruning Condition): If

$$\mathrm{upRa}(X) < \mathrm{minSeqRa},$$

then for any extension $Y \supseteq X$,

$$\mathrm{hqSeqRa}(Y) < \mathrm{minSeqRa}.$$

In this case, the entire search branch rooted at pattern X can be safely pruned (Algorithm 1, Lines 5–6.)

Algorithm 1: DFS-Search

Require: Prefix pattern P, PFL$(\{i\})$, PFL(P), database D, thresholds $minSupRa$, $maxPr$, $maxStd$, $minHqRa$, $minSeqRa$, global Result

1: **for** each item $i \in I$ with $i > last(P)$ **do**
2: $Y \leftarrow P \cup \{i\}$
3: Construct PFL(Y) from PFL(P) and PFL$(\{i\})$ using Algorithm 2
4: Compute candSeq(Y) and upRa(Y) using PFL(Y)
5: **if** upRa$(Y) < minSeqRa$ **then**
6: **continue** {prune the branch rooted at Y}
7: **end if**
8: **if** HDPFP-CHECK $(Y, \text{PFL}(Y), D, minSupRa, maxPr, maxStd, minHqRa, minSeqRa)$ **then**
9: Result $\leftarrow$ Result $\cup \{Y\}$
10: **end if**
11: DFS-SEARCH$(Y, \text{PFL}(Y), D, minSupRa, maxPr, maxStd, minHqRa, minSeqRa, \text{Result})$
12: **end for**

Algorithm 2: Constructing PFL(Y) from PFL(P)

Require: PFL(P), item i, PFL$(\{i\})$,database D
Ensure: PFL(Y) where $Y = P \cup \{i\}$

1: Initialize PFL$(Y) \leftarrow \varnothing$
2: **for** each record $(sid, \text{TR}, q\text{-list}, supRa, maxPer, stanDev, quRa)$ in PFL(P) **do**
3: Let S be the sequence in D with identifier sid
4: Determine TR(Y, S) by intersecting TR(P, S) with TR$(\{i\}, S)$
5: **if** TR$(Y, S) \neq \varnothing$ **then**
6: Compute supRa(Y, S), maxPer(Y, S), stanDev(Y, S) and quRa(Y, S) according to Section 3
7: Build the updated q-list for Y in S
8: Insert $(sid, \text{TR}(Y, S), q\text{-list}, \text{supRa}(Y, S), \text{maxPer}(Y, S), \text{stanDev}(Y, S), \text{quRa}(Y, S))$ into PFL(Y)
9: **end if**
10: **end for**
11: **return** PFL(Y)

4.3 The HQPFPS-List Structure

To efficiently evaluate candidate patterns across multiple sequences, MHQPFPS employs a vertical representation structure referred to as the HQPFPS-list. This structure is designed to compactly maintain all information required for computing support, periodicity, and quantitative measures, thereby avoiding repeated scans of the original database.

For a pattern X and a sequence S in which X occurs, the HQPFPS-list stores the following information:

- **sid:** the identifier of sequence S;
- **tran-list:** the list TR(X, S) of transaction indices where X occurs;
- **q-list:** the quantitative contributions $q(X, T_j, S)$ associated with each occurrence of X;

- supRa(X, S): the support ratio of X in S;
- maxPer(X, S): the maximum period of X in S;
- stanDev(X, S): the standard deviation of the periods of X in S;
- quRa(X, S): the quantitative ratio of X in S.

For a pattern X, its HQPFPS-list is denoted by

$$\text{PFL}(X) = \{(sid, \text{TR}, q\text{-list}, \text{supRa}, \text{maxPer}, \text{stanDev}, \text{quRa})\}.$$

All entries in PFL(X) are computed in accordance with the formal definitions given in Section 3. When a pattern is extended during the mining process, the corresponding HQPFPS-list can be constructed and updated directly from existing lists, without rescanning the original database. This property is essential for ensuring the efficiency of the proposed depth-first search framework.

4.4 Mining Procedure

Based on the pruning strategy and the HQPFPS-list structure introduced above, the MHQPFPS algorithm enumerates all High-Quantitative Periodic Frequent Patterns from a multi-sequence database. The overall procedure is outlined in Algorithm 3.

Algorithm 3: MHQPFPS

Require: Database D, thresholds $minSupRa$, $maxPr$, $maxStd$, $minHqRa$, $minSeqRa$
Ensure: All HQPFPS in D

```
1: Result ← ∅
2: for each item i ∈ I do
3:    Construct PFL({i}) from D
4:    Compute candSeq({i}) and upRa({i}) using PFL({i})
5:    if upRa({i}) ≥ minSeqRa then
6:       if HQPFP-CHECK({i}, PFL({i}), D, minSupRa, maxPr, maxStd, minHqRa, minSeqRa)
         then
7:          Result ← Result ∪{{i}}
8:       end if
9:       DFS-SEARCH({i}, PFL({i}), D, minSupRa, maxPr, maxStd, minHqRa, minSeqRa, Result)
10:   end if
11: end for
12: return Result
```

Given a database D and user-specified thresholds $(minSupRa, maxPr, maxStd, minHqRa, minSeqRa)$, the algorithm begins with a single scan of D to construct the HQPFPS-list PFL$(\{i\})$ for each 1-itemset $\{i\}$. For every item $i \in I$ and sequence $S \in D$, the statistics supRa$(\{i\}, S)$, maxPer$(\{i\}, S)$, stanDev$(\{i\}, S)$, and quRa$(\{i\}, S)$ are computed directly from PFL$(\{i\})$. These values are then used to derive the candidate sequence set candSeq$(\{i\})$ and the corresponding upper bound upRa$(\{i\})$ = |candSeq$(\{i\})$|/|D|.

A 1-itemset $\{i\}$ is considered promising only if upRa$(\{i\}) \geq$ minSeqRa (Algorithm 3, Line 5). For such patterns, the procedure HQPFP-CHECK (Algorithm 4 Lines 1–12) is invoked to verify whether $\{i\}$ satisfies all single-sequence constraints (Definition 8) as well as the cross-sequence constraint (Definition 9). If $\{i\}$ is an HQPFPS, it is added to the global result set. The algorithm then calls the recursive procedure DFS-SEARCH (Algorithm 1) to further extend $\{i\}$.

Algorithm 4: HQPFP-Check

Require: Pattern X, PFL(X), $|D|$, thresholds $minSupRa$, $maxPr$, $maxStd$, $minHqRa$, $minSeqRa$
Ensure: Whether X is an HQPFPS

```
validSeq ← 0
for each (sid, TR, q-list, supRa, maxPer, stanDev, quRa) in PFL(X) do
    if supRa ≥ minSupRa and maxPer ≤ maxPr and stanDev ≤ maxStd and quRa ≥ minHqRa
        then
            validSeq ← validSeq + 1
    end if
end for
hqSeqRa(X) ← validSeq/|D|
if hqSeqRa(X) ≥ minSeqRa then
    return true
else
    return false
end if
```

The DFS-SEARCH procedure follows a depth-first pattern-growth strategy. Given a current prefix pattern P, it generates extensions by appending items that are lexicographically larger than the last item in P. For each extension $Y = P \cup \{i\}$, a new HQPFPS-list PFL(Y) is constructed by combining PFL(P) and PFL$(\{i\})$ using Algorithm 2. Since both lists record transaction identifiers, PFL(Y) can be constructed efficiently through intersection operations, without scanning the original database.

From PFL(Y), the algorithm computes the candidate sequence set candSeq(Y) and the corresponding upper bound upRa(Y). If upRa$(Y) < minSeqRa$, Corollary 1 guarantees that no extension of Y can satisfy the HQPFPS condition, and the entire branch rooted at Y is pruned (Algorithm 1, Lines 5–6). Otherwise, HQPFP-CHECK (Algorithm 4) is applied to Y. If Y satisfies all constraints, it is reported as an HQPFPS, and DFS-SEARCH is recursively invoked on Y.

Algorithm 2 details the construction of PFL(Y) from PFL(P). For each record $(sid$, TR, q-list, supRa, maxPer, stanDev, quRa$)$ in PFL(P), the transaction list TR(Y, S) is obtained by intersecting TR(P, S) and TR$(\{i\}, S)$ (Algorithm 2, Line 4). If TR(Y, S) is non-empty, the sequence-level statistics supRa(Y, S), maxPer(Y, S), stanDev(Y, S), and quRa(Y, S) are recomputed, a new q-list is generated, and the corresponding record is inserted into PFL(Y).

Finally, the procedure HQPFP-Check (Algorithm 4) counts the number of sequences in which a pattern satisfies all single-sequence constraints using PFL(X). Dividing this count by $|D|$ yields hqSeqRa(X), which determines whether X is an HQPFPS.

5 Experimental Evaluation

This section evaluates the performance and effectiveness of the proposed MHQPFPS algorithm. The experimental study focuses on three aspects that are critical for pattern mining algorithms: (1) runtime efficiency, (2) the number of discovered patterns, and (3) memory consumption. To better understand the behavior of the proposed method, the experiments include both parameter-based analysis and comparisons with reduced variants of MHQPFPS, which help illustrate the effect of its main components on efficiency and output behavior. All experiments are conducted under varying settings of $minSupRa$ and $minSeqRa$ in order to assess the impact of support-based and cross-sequence constraints on algorithmic behavior.

5.1 Experimental Setup

All experiments were carried out in a Python 3 environment using PyCharm on a MacBook Air running macOS, equipped with an Apple M4 processor and 16 GB of unified memory. The MHQPFPS algorithm was implemented in a single-threaded manner to provide a fair and controlled evaluation of its computational characteristics.

Four datasets were used in the experimental study, including three real-world datasets—*FIFA*, *Bike*, and *Leviathan*—and one synthetic dataset, *T23L68KD15K*. All datasets were obtained from the SPMF Library. The real-world datasets represent typical application scenarios of sequential and periodic pattern mining, while the synthetic dataset allows controlled adjustment of data characteristics, facilitating a systematic evaluation of algorithm behavior under different conditions.

The primary goal of the experiments is to investigate how the parameters $minSupRa$ and $minSeqRa$ influence both the efficiency and the output of MHQPFPS. We denote by MHQPFPS(x, y, z) an instance of the algorithm configured with $minSeqRa = x$, $minHqRa = y$, and $maxPr = z$. Unless otherwise specified, the parameter $maxStd$ is fixed throughout the experiments, and $minSupRa$ is varied as reported in the corresponding figures.

To provide a reference point for comparison, a *baseline* configuration is defined using relatively loose thresholds, allowing a large number of high-quantitative periodic patterns to be generated: $maxPr = 20$, $maxStd = 10$, $minHqRa = 0.01$, $minSeqRa = 0$. All other experimental configurations are evaluated relative to this baseline in terms of runtime, number of discovered patterns, and memory consumption.

5.2 Influence of minSupRa and minSeqRa

Runtime: Fig. 2 reports the runtime of MHQPFPS under different combinations of $minSupRa$ and $minSeqRa$ across the four datasets. Overall, runtime decreases steadily as minSupRa increases, because a higher support-ratio threshold reduces the number of candidate patterns and correspondingly shrinks the effective search space.

This behavior is most pronounced on the Bike and T23L68KD15K datasets, where the runtime curves decrease monotonically with minSupRa under configurations such as MHQPFPS $(0.001, 0.15, 10)$. The parameter $minSeqRa$ also plays a critical role in runtime performance. When minSeqRa is set to a small positive value (e.g., 0.0001 or 0.001), the upper-bound pruning mechanism eliminates unpromising candidates at an earlier stage of the search, thereby reducing the number of HQPFPS-lists that must be constructed and evaluated.

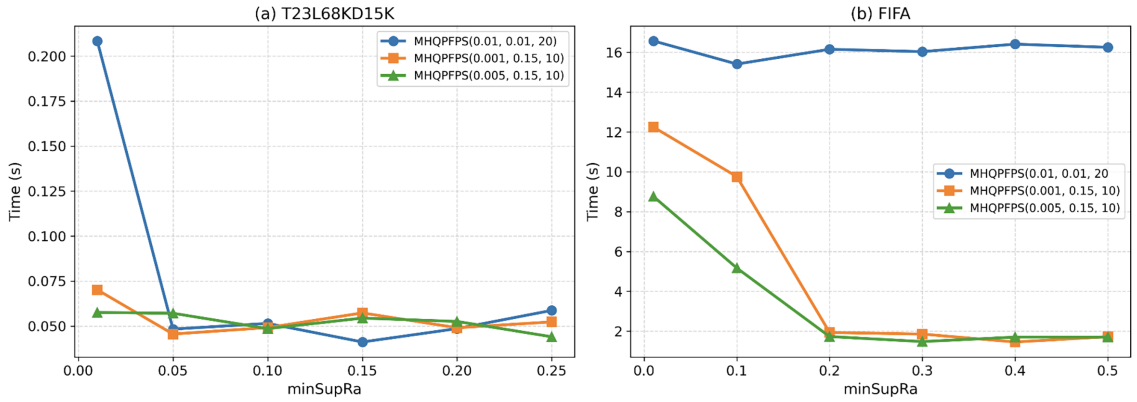

Figure 2: (Continued)

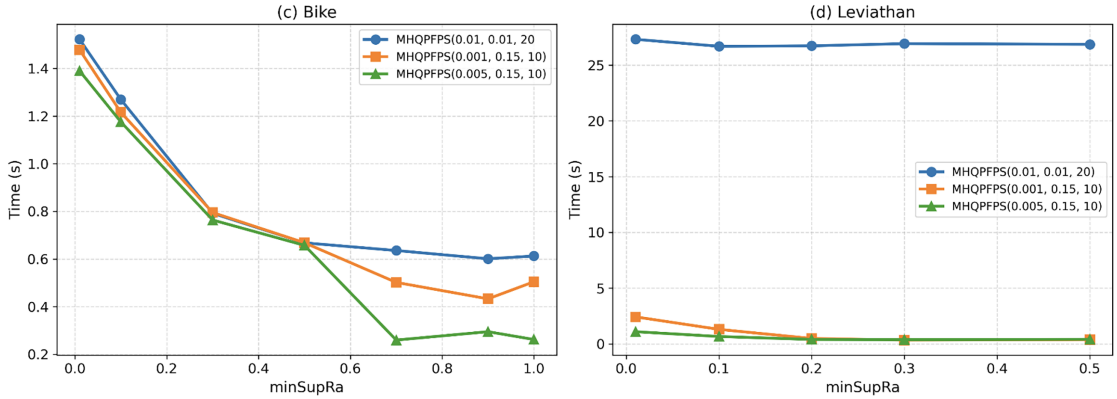

Figure 2: Runtime of MHQPFPS under different *minSupRa* and *minSeqRa* settings. The results show that runtime generally decreases as the thresholds become stricter, indicating that stronger support-based and cross-sequence constraints help reduce the search space.

For example, on the *FIFA* dataset, setting $minSeqRa = 0.0001$ reduces runtime by nearly an order of magnitude compared with the baseline configuration. This result indicates that incorporating cross-sequence constraints into the upper-bound pruning strategy can substantially reduce runtime, particularly when the candidate search space is large.

Number of patterns: Fig. 3 illustrates how the number of discovered HQPFPS varies with $minSupRa$ and $minSeqRa$. Increasing either threshold leads to a smaller set of discovered patterns, which is consistent with the stricter constraints imposed on periodicity, support ratio, and cross-sequence occurrence. On the *FIFA* dataset, the baseline configuration consistently produces a large number of patterns (approximately 3000). However, when minSeqRa is increased to 0.0001, the number of discovered patterns decreases to fewer than 100, representing a reduction of more than 96% relative to the baseline.

A similar trend is observed on the *Bike* dataset: raising $minSupRa$ from 0.01 to 0.7 reduces the number of output patterns from 67 to 48. These results indicate that a considerable proportion of high-quantitative periodic patterns occur in only a limited number of sequences. The cross-sequence constraint enforced by $minSeqRa$ is therefore essential for filtering out such sequence-specific patterns and retaining only those that are more representative at the database level.

Memory usage: Fig. 4 shows the memory consumption of MHQPFPS under different parameter configurations. On the *FIFA* dataset, memory usage under the baseline setting $\text{MHQPFPS}(0, 0.01, 20)$ decreases from approximately 366 to 164 MB as $minSupRa$ increases from 0.01 to 0.5. This behavior is consistent with the reduction in the number of maintained patterns, since higher support thresholds limit the number of HQPFPS-list that must be stored simultaneously.

For the synthetic *T23L68KD15K* dataset, the influence of $minSeqRa$ on memory consumption is marginal, and the corresponding curves remain nearly flat. A plausible explanation is that this dataset contains relatively few sequences and transactions. Consequently, variations in $minSeqRa$ primarily affect the depth of the search process rather than the total number of pattern lists retained in memory.

Overall, these results indicate that $minSupRa$ is the dominant factor governing memory consumption, whereas $minSeqRa$ mainly contributes to runtime reduction by effectively shrinking the search space through early pruning.

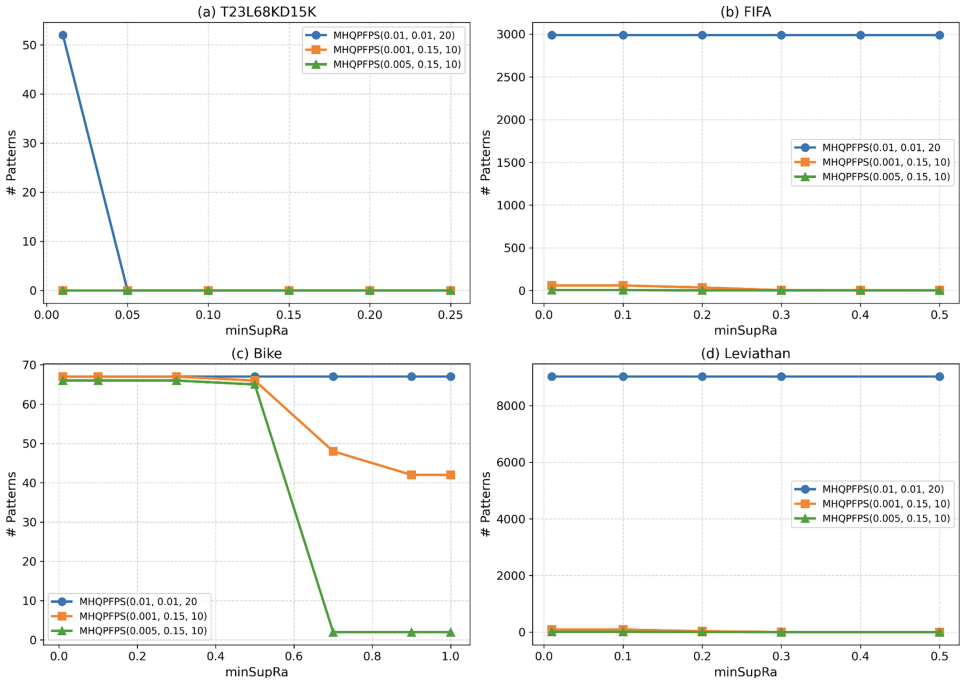

Figure 3: Number of discovered HQPFPS under different *minSupRa* and *minSeqRa* settings. The results show that increasing either threshold reduces the number of valid patterns, reflecting the stronger filtering effect of support and cross-sequence constraints.

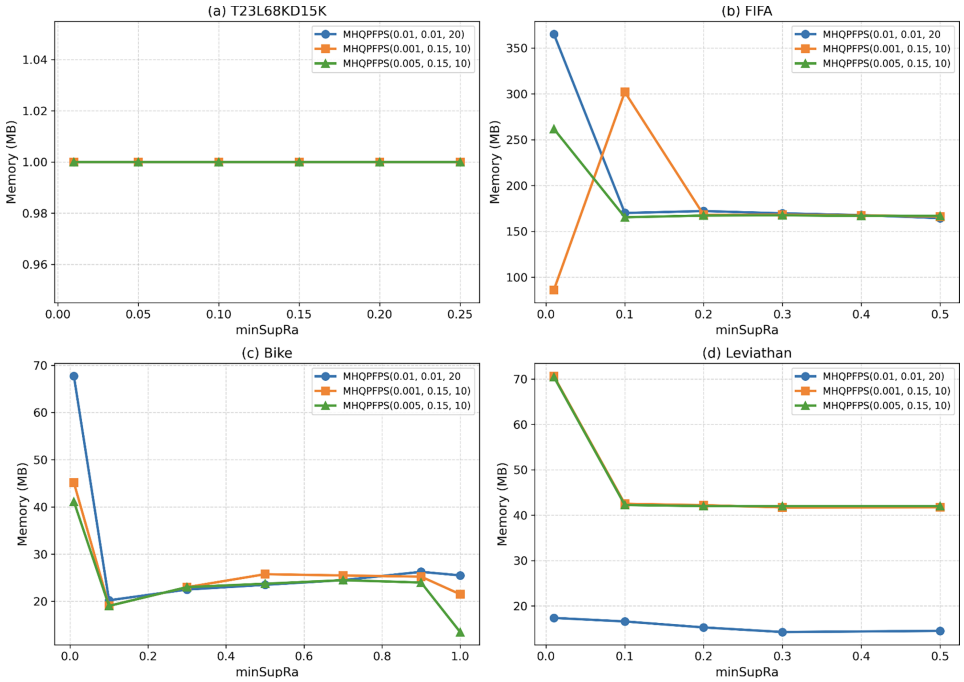

Figure 4: Memory consumption of MHQPFPS under different *minSupRa* and *minSeqRa* settings on different datasets. The results indicate that memory usage generally decreases as the search space becomes smaller, with *minSupRa* showing a stronger influence than *minSeqRa*.

5.3 Ablation Study

To isolate the effect of the main components of MHQPFPS, we compare the full method with two reduced variants: **MHQPFPS-noUB**, which disables the upper-bound pruning condition while keeping all other components unchanged, and **MHQPFPS-noQ**, which removes the quantitative ratio constraint $\text{quRa}(X, S) \geq \text{minHqRa}$. All three variants are evaluated on the same four datasets under identical parameter settings.

Fig. 5 reports the runtime comparison. MHQPFPS consistently runs faster than MHQPFPS-noUB across all datasets. The speedup is most pronounced on Leviathan, where MHQPFPS-noUB maintains a near-constant runtime of approximately 7.3 s regardless of *minSupRa*, while MHQPFPS decreases to approximately 0.3 s at *minSupRa* = 0.1, corresponding to a speedup of about 20×. On T23L68KD15K, MHQPFPS-noUB maintains a runtime of approximately 0.5 s, compared with 0.05 s for MHQPFPS, corresponding to a reduction of about 10×. MHQPFPS-noQ closely follows the full MHQPFPS in all runtime curves, indicating that the efficiency gain mainly comes from the upper-bound pruning mechanism.

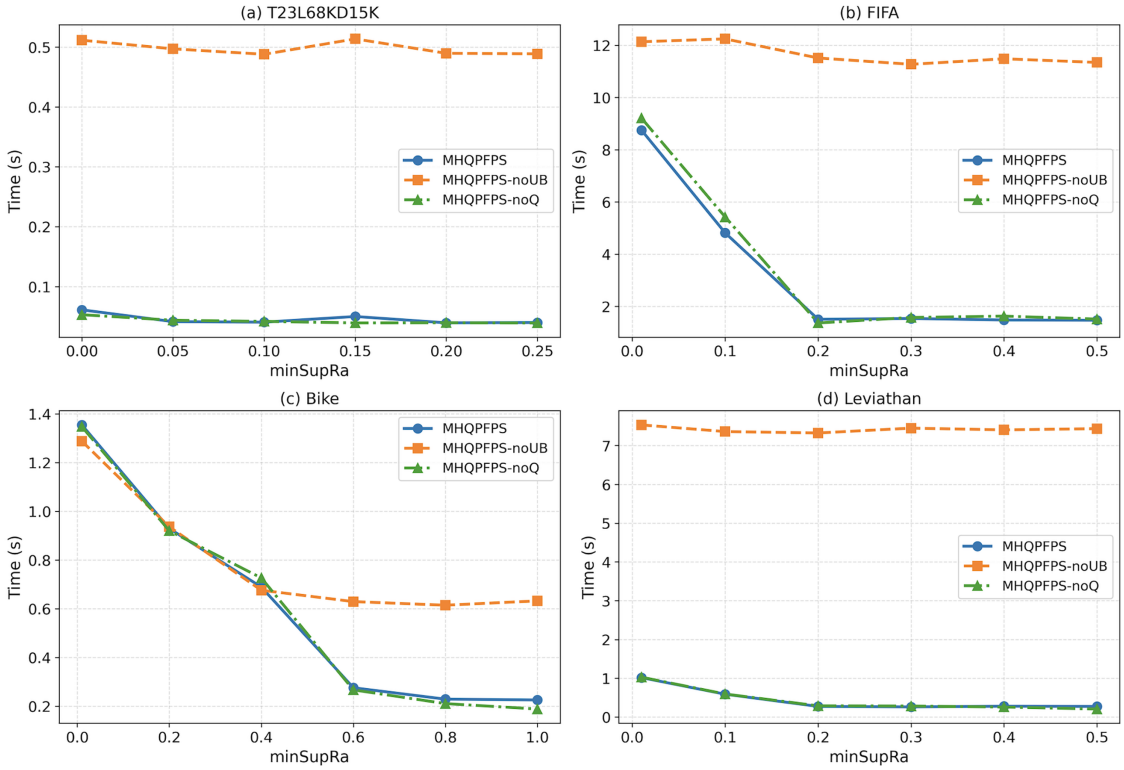

Figure 5: Runtime comparison of MHQPFPS and its reduced variants under different *minSupRa* settings. The results show that removing the upper-bound pruning strategy leads to a substantial increase in runtime, demonstrating its effectiveness in improving efficiency.

Fig. 6 shows the number of discovered patterns. Across all datasets and parameter settings, MHQPFPS and MHQPFPS-noUB discover exactly the same pattern set, which empirically supports the correctness of the pruning strategy. MHQPFPS-noQ consistently discovers more patterns than the full MHQPFPS, since it retains periodically frequent patterns regardless of quantitative contribution. On FIFA, MHQPFPS-noQ returns up to 375 patterns at *minSupRa* = 0.01, while MHQPFPS discovers none under the same setting. On Leviathan, MHQPFPS-noQ discovers up to 250 patterns, compared with approximately 10 for MHQPFPS.

The gap between the two variants corresponds to patterns that satisfy periodicity and support constraints but do not satisfy the quantitative ratio requirement.

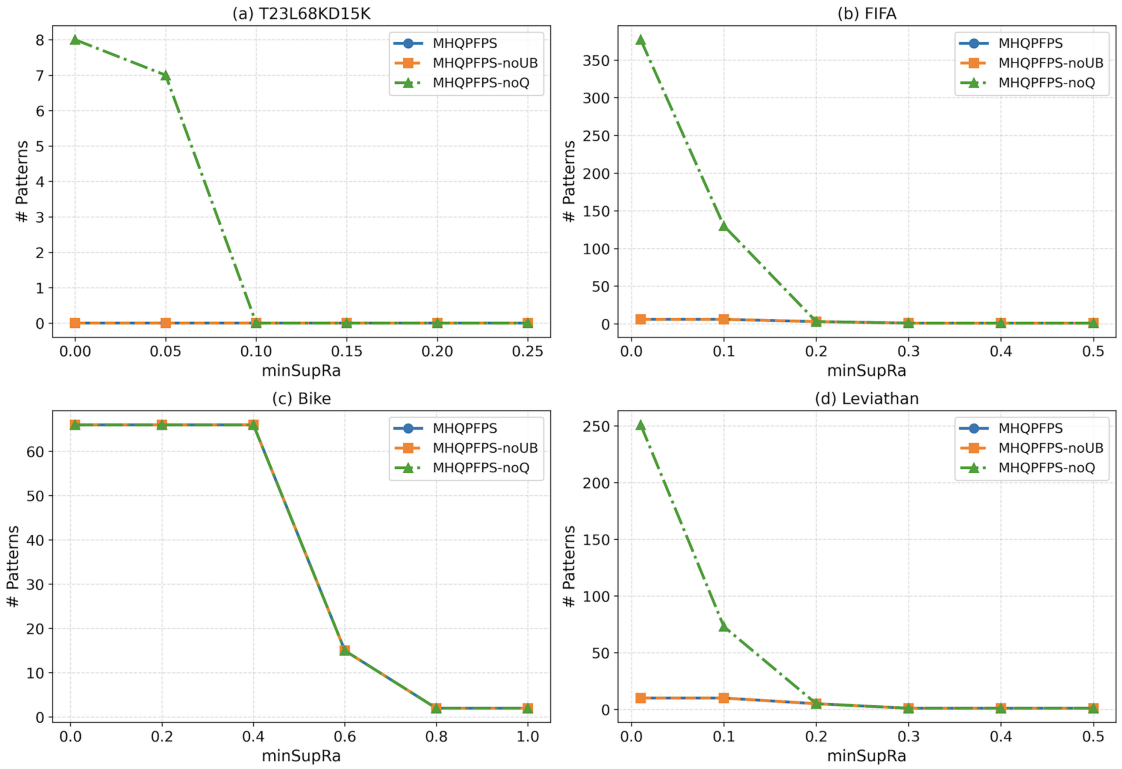

Figure 6: Number of discovered patterns for MHQPFPS and its reduced variants under different *minSupRa* settings. The results show that disabling the quantitative ratio constraint produces more patterns, whereas removing upper-bound pruning does not change the discovered pattern set.

Fig. 7 shows memory consumption. MHQPFPS-noUB generally uses more memory than MHQPFPS on most datasets, since the absence of pruning forces the algorithm to maintain a larger number of active HQPFPS-list simultaneously. An exception is observed on FIFA at *minSupRa* = 0.01, where MHQPFPS exhibits a transient memory peak before the effect of pruning becomes apparent; as *minSupRa* increases, this peak quickly disappears and MHQPFPS remains below MHQPFPS-noUB. Overall, the results show that the upper-bound pruning mechanism reduces runtime and memory consumption without changing the discovered pattern set.

5.4 Sensitivity Analysis of maxStd and minHqRa

To examine the influence of the two threshold parameters *maxStd* and *minHqRa*, Figs. 8 and 9 report the sensitivity of MHQPFPS to these two parameters, respectively, with all other parameters fixed at *minSupRa* = 0.1, *maxPr* = 10, and *minSeqRa* = 0.001.

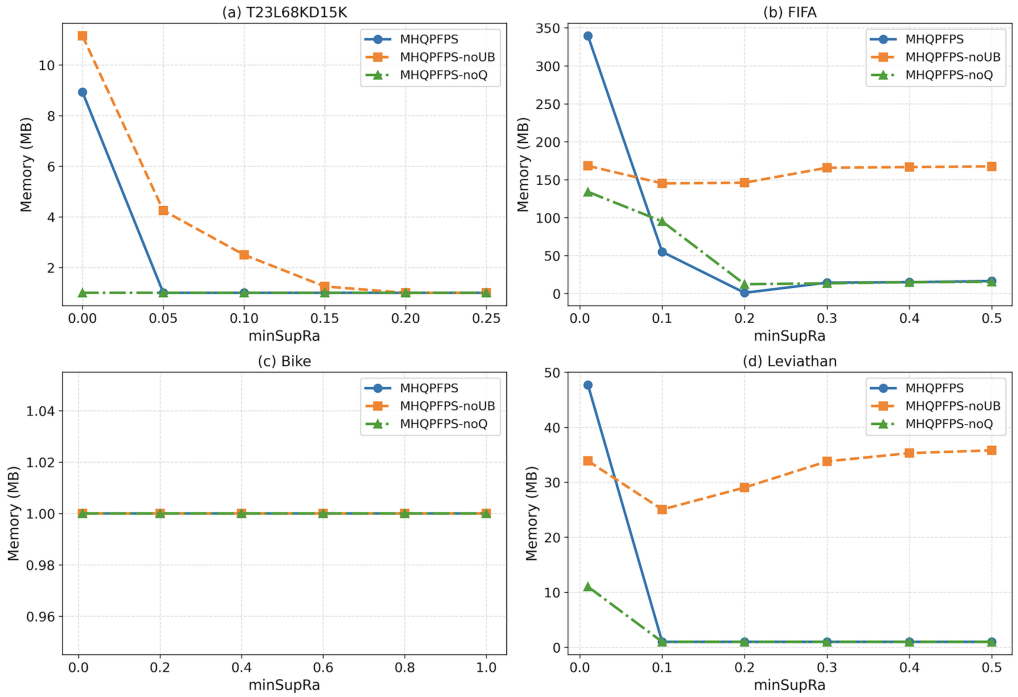

Figure 7: Memory consumption of MHQPFPS and its reduced variants under different *minSupRa* settings. The results indicate that the upper-bound pruning strategy generally reduces memory usage by limiting the number of active candidate lists maintained during the search.

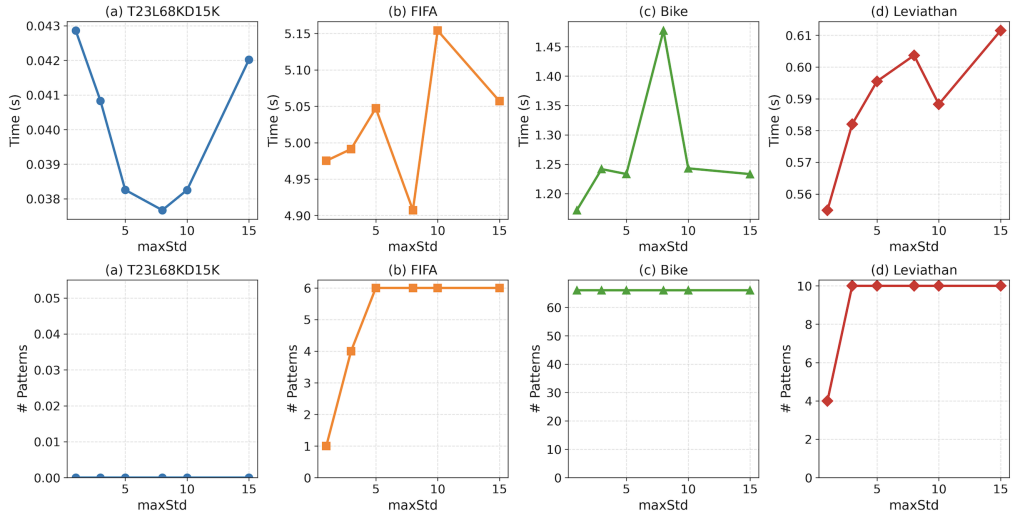

Figure 8: Sensitivity of MHQPFPS to *maxStd* in terms of runtime (top row) and number of discovered patterns (bottom row) across four datasets: (**a**) T23L68KD15K, (**b**) FIFA, (**c**) Bike, and (**d**) Leviathan. The results show that increasing *maxStd* mainly affects the number of valid patterns, while its influence on runtime remains limited.

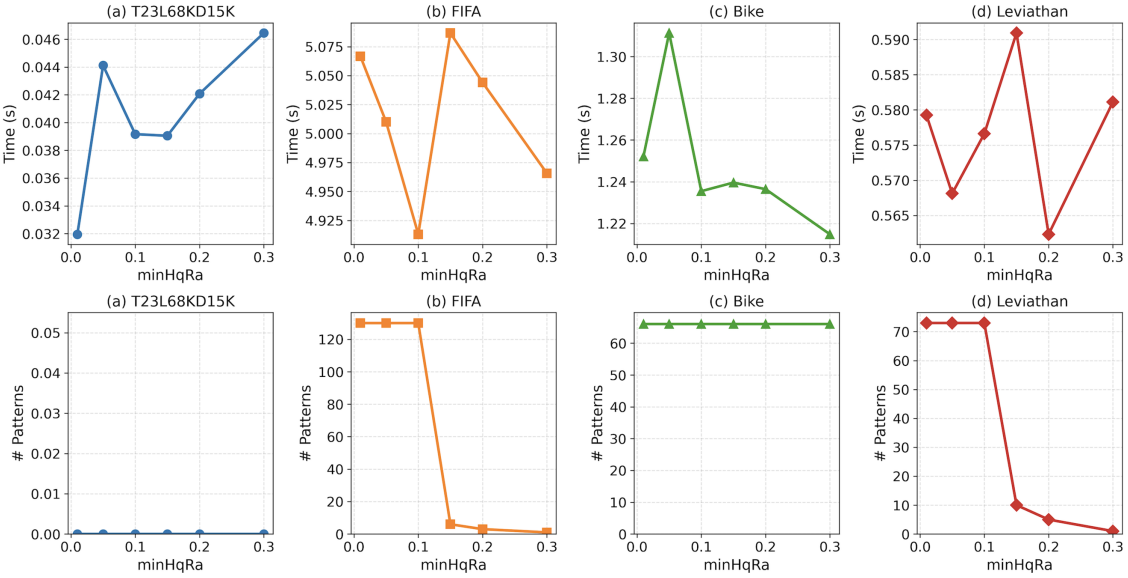

Figure 9: Sensitivity of MHQPFPS to *minHqRa* in terms of runtime (top row) and number of discovered patterns (bottom row) across four datasets: (**a**) T23L68KD15K, (**b**) FIFA, (**c**) Bike, and (**d**) Leviathan. The results show that increasing *minHqRa* significantly reduces the number of discovered patterns, while runtime remains relatively stable.

Runtime. Across both parameters and all four datasets, runtime shows no clear monotonic trend and varies only within a narrow range. This behavior is consistent with the algorithm design: the upper-bound pruning condition depends only on *supRa* and *maxPer*, which are anti-monotonic, whereas *maxStd* and *minHqRa* affect only the final HQPFP verification step and do not alter the candidate search space explored during DFS.

Pattern count. For *maxStd* (Fig. 8), FIFA and Leviathan exhibit a clear increase in pattern count followed by saturation. On FIFA, the count rises from 1 at *maxStd* = 1 to 6 at *maxStd* = 5 and remains stable thereafter, indicating that no patterns in this dataset have a period standard deviation exceeding 5. On Leviathan, saturation occurs at *maxStd* = 3 with a maximum of 10 patterns.

For *minHqRa* (Fig. 9), FIFA and Leviathan exhibit a sharp decline in pattern count. On FIFA, the pattern count drops from 126 at *minHqRa* = 0.10 to fewer than 10 at *minHqRa* = 0.15, a reduction of over 92% within a single threshold step. On Leviathan, the count decreases from 71 to approximately 10 over the same interval. This suggests that a large fraction of periodically frequent patterns in these datasets carry quantitative ratios concentrated in the interval $[0.10, 0.15]$.

Bike remains insensitive to both parameters across all tested values, reflecting that its patterns inherently exhibit low period variance and high quantitative contribution. T23L68KD15K yields no patterns under the experimental configuration, consistent with the results reported in Figs. 3 and 6.

Overall, these results show that *maxStd* and *minHqRa* mainly act as quality-filtering thresholds whose influence on the output is data-dependent, while their effect on runtime remains limited.

5.5 Sensitivity to Quantitative Value Distribution

To examine the sensitivity of MHQPFPS to quantitative values, we conduct a controlled experiment on T23L68KD15K by replacing all item quantities with values drawn from three distributions while preserving the original sequence structure: (A) *Uniform* $\mathcal{U}(1, 10)$, in which all values are evenly distributed; (B) *Normal* $\mathcal{N}(5, 1.5)$, truncated to $[1, 10]$, in which values concentrate around the mean; and (C) *Long-tail* (Pareto with

α = 1.5), in which the majority of values are small while a minority are disproportionately large. All other parameters are fixed at minSupRa = 0.05, maxPr = 10, maxStd = 10, minSeqRa = 0.001, and minHqRa is varied across $\{0.01, 0.05, 0.10, 0.15, 0.20, 0.25, 0.30\}$.

Fig. 10 shows the number of discovered patterns under each distribution. Under the long-tail distribution, the pattern count drops sharply at low minHqRa values, since a small number of high-quantity transactions dominate the sequence-level total $q(S)$, causing most patterns to exhibit low quRa values. Even a modest increase in minHqRa therefore eliminates a large fraction of candidates. Under the uniform distribution, the decline is more gradual, as quantitative contributions are spread more evenly across transactions and no single transaction disproportionately inflates $q(S)$. The normal distribution exhibits intermediate sensitivity, consistent with its moderate degree of concentration around the mean.

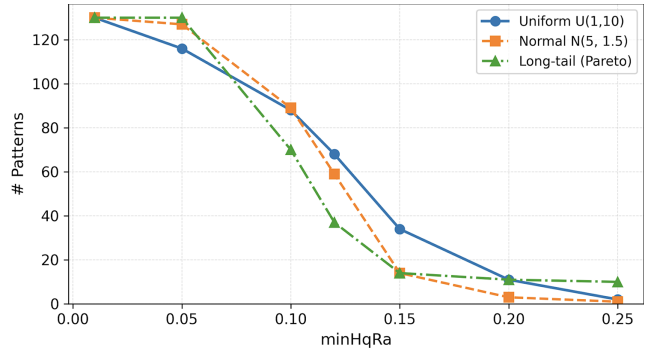

Figure 10: Number of discovered patterns under different *minHqRa* settings for three quantitative value distributions. The results show that the distribution of quantitative values has a clear effect on pattern discovery, with the long-tail setting producing a sharper decline as *minHqRa* increases.

Fig. 11 shows that runtime remains relatively stable across all three distributions and all tested minHqRa values. This suggests that the pruning behavior is not strongly affected by the quantitative value distribution under the tested settings, although the number of discovered patterns is clearly influenced by how quantitative values are distributed.

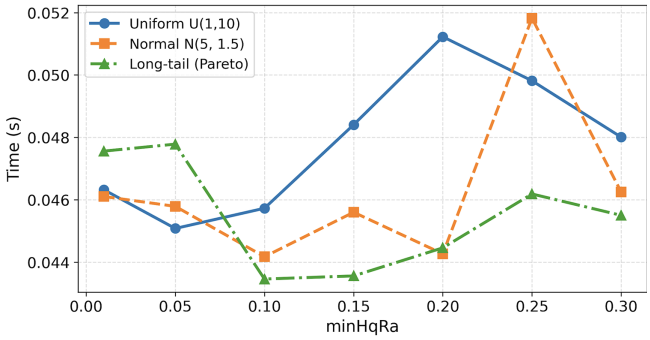

Figure 11: Runtime under different *minHqRa* settings for three quantitative value distributions. The results show that runtime remains relatively stable across the tested distributions, although the number of discovered patterns varies substantially.

These results indicate that MHQPFPS is sensitive to the distribution of quantitative values mainly through its effect on quRa and the resulting number of discovered patterns. Accordingly, the practical choice of minHqRa should take into account the quantitative characteristics of the target dataset.

6 Conclusion

This work investigated the problem of mining High-Quantitative Periodic Frequent Patterns in multi-sequence databases, moving beyond the traditional focus on periodicity within a single sequence. By jointly modeling support ratio, periodic regularity, quantitative contribution, and cross-sequence consistency, we provided a unified definition of HQPFPS that better reflects the characteristics of real-world sequential data. To efficiently solve this problem, we designed the MHQPFPS algorithm, which combines a compact HQPFPS-list structure with a depth-first pattern-growth strategy and an anti-monotonic upper-bound pruning mechanism. This design enables effective candidate evaluation while avoiding repeated scans of the database.

Experimental results on both real-world and synthetic datasets show that the proposed approach is efficient and robust under a wide range of parameter settings. In particular, $minSupRa$ has a dominant influence on memory consumption and overall runtime by controlling the number of maintained candidates, whereas $minSeqRa$ is crucial for reducing the search space through early pruning of patterns that lack sufficient cross-sequence support. These observations highlight the importance of explicitly incorporating sequence-level constraints when mining periodic patterns across multiple sequences.

Several directions remain for future research. One promising extension is to investigate adaptive or data-driven strategies for automatically selecting threshold parameters. In addition, the HQPFPS-list structure could be further optimized to reduce memory overhead, and parallel or distributed implementations could be explored to improve scalability on large-scale datasets. Finally, extending the proposed framework to uncertain, weighted, or multidimensional sequence databases would further broaden the applicability of high-quantitative periodic pattern mining in practical settings.

Acknowledgement: Not applicaple.

Funding Statement: This research was funded by the Startup Foundation for Introducing Talent of NUIST, China and the Natural Science Foundation of Shandong Province, China (Grant no. ZR2022MF298).

Author Contributions: The authors confirm contribution to the paper as follows: study conception and design: Yan Ge, Zhenzhou Zhang, Chien-Ming Chen; analysis and interpretation of results: Yan Ge, Zhenzhou Zhang; draft manuscript preparation: Yan Ge, Zhenzhou Zhang, Chien-Ming Chen. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: Not applicable.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Chen MS, Han J, Yu PS. Data mining: an overview from a database perspective. IEEE Trans Knowl Data Eng. 2002;8(6):866–83. doi:10.1109/69.553155.
2. Bramer M. Principles of data mining. London, UK: Springer; 2007.
3. Han J, Cheng H, Xin D, Yan X. Frequent pattern mining: current status and future directions. Data Min Knowl Discov. 2007;15(1):55–86. doi:10.1007/s10618-006-0059-1.
4. Chen J, Yang S, Gao T, Ying Y, Li T, Li P. Multi-type concept drift detection under a dual-layer variable sliding window in frequent pattern mining with cloud computing. J Cloud Comput. 2024;13(1):40. doi: 10.1186/s13677-023-00566-9.

5. Zhong N, Li Y, Wu ST. Effective pattern discovery for text mining. IEEE Trans Knowl Data Eng. 2010;24(1):30–44. doi:10.1109/tkde.2010.211.
6. Li X, Deng ZH. Mining frequent patterns from network flows for monitoring network. Expert Syst Appl. 2010;37(12):8850–60. doi:10.1016/j.eswa.2010.06.012.
7. Liu YC, Lee LF, Yeh KH. Enhancing UAV security with GPS spoofing and jamming anomaly detection. J Reliab Secur Comput. 2025;1(1):54–67.
8. Henriques R, Madeira SC. BicPAM: pattern-based biclustering for biomedical data analysis. Algorithms Mol Biol. 2014;9(1):27.
9. Hellal A, Romdhane LB. Minimal contrast frequent pattern mining for malware detection. Comput Secur. 2016;62:19–32.
10. Gan W, Chen L, Wan S, Chen J, Chen CM. Anomaly rule detection in sequence data. IEEE Trans Knowl Data Eng. 2023;35(12):12095–108. doi:10.1109/tkde.2021.3139086.
11. Ma C, Vu HQ, Wang J, Trieu VH, Li G. Understanding residents' behavior for smart city management by sequential and periodic pattern mining. IEEE Trans Comput Soc Syst. 2023;11(1):1260–76. doi:10.1109/tcss.2023.3249740.
12. 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; 2002 Jul 23–26; Edmonton, AB, Canada. p. 429–35.
13. Li G, Xiang J, Fang W, Wang J, Shang T. HUPSP-LAL: efficiently mining utility-driven sequential patterns in uncertain sequences. Expert Syst Appl. 2025;270:126536.
14. Tanbeer SK, Ahmed CF, Jeong BS, Lee YK. Discovering periodic-frequent patterns in transactional databases. In: Pacific-Asia Conference on Knowledge Discovery and Data Mining. Berlin/Heidelberg, Germany: Springer; 2009. p. 242–53.
15. Jiang Z, Tang Y, Gu J, Zhang Z, Liu W. Discovering periodic frequent travel patterns of individual metro passengers considering different time granularities and station attributes. Int J Transp Sci Technol. 2024;14(4):12–26. doi:10.1016/j.ijtst.2023.03.003.
16. Fournier-Viger P, Yang P, Li Z, Lin JCW, Kiran RU. Discovering rare correlated periodic patterns in multiple sequences. Data Knowl Eng. 2020;126:101733.
17. Chien-Ming C, Zhang Z, Wu J, Lakshmanna K. High utility periodic frequent pattern mining in multiple sequences. Comput Model Eng Sci. 2023;137(1):733. doi:10.32604/cmes.2023.027463.
18. Hong TP, Kuo CS, Chi SC. Mining association rules from quantitative data. Intell Data Anal. 1999;3(5):363–76. doi:10.3233/ida-1999-3504.
19. Nguyen H, Le N, Bui H, Le T. Mining frequent weighted utility patterns with dynamic weighted items from quantitative databases. Appl Intell. 2023;53(16):19629–46. doi:10.1007/s10489-023-04554-z.