Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Mining Quantitative Top-k Sequential Patterns under Constraints

Amina Kemmar

kemmar.amina@gmail.com

Oran Graduate School of Economics

Chahira Touati

Université Oran 1 Ahmed Ben Bella

Amel Mounia Djebbar

Advanced DAta Science and Cognitive Applications Laboratory, University of Science and Technology of Oran Mohamed-Boudiaf

Research Article

Keywords:

Posted Date: November 10th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-11458206/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Mining Quantitative Top-k Sequential Patterns under Constraints

Amina Kemmar[1,2*], Chahira Touati[2†] and Amel Mounia Djebbar[1,3†]

[1*]Oran Graduate School of Economics, BP 65 CH 2 Achaba Hnifi - USTO, Oran, 31000, Algeria.
[2]Computer Science, LITIO, University of Oran 1 Ahmed Ben Bella, Oran, 31000, Algeria.
[3]Computer Science, Advanced DAta Science and Cognitive Applications Laboratory, University of Science and Technology of Oran Mohamed-Boudiaf, USTO, Oran, 31000, Algeria.

*Corresponding author(s). E-mail(s): amina.kemmar@ese-oran.dz;
Contributing authors: touati.chahira@univ-oran1.dz;
amel.djebbar@ese-oran.dz;
†These authors contributed equally to this work.

Abstract

Sequential pattern mining with quantitative attributes is essential in many real-world applications where items are associated with numerical values such as quantities, dosages, or utilities. While numerous algorithms have been proposed for frequent or top-k sequential pattern mining, none simultaneously address top-k extraction, quantitative constraints, and flexible user-defined constraints within a unified framework. In this paper, we introduce a novel Constraint Programming (CP) model for mining top-k quantitative sequential patterns under heterogeneous constraints. Our approach relies on a global constraint that enforces both subsequence embedding and frequency filtering through dynamic support raising. We propose two dedicated search strategies that accelerate convergence toward the k most frequent patterns while progressively pruning the search space. Extensive experiments on real-world datasets demonstrate that our approach scales favorably compared to state-of-the-art top-k sequential pattern mining algorithms in terms of scalability and flexibility, while naturally supporting complex structural and quantitative constraints.

Keywords: sequence datasets, sequential patterns, quantitative items, top-k patterns; constraints, constraint programming.

1 Introduction

One of the most essential data mining operations, which has various applications in different domains, is known as Sequential Pattern Mining (SPM). The applications of this operation include market basket analysis, web mining, bioinformatics, intrusion detection, and many others. So far, the main objective of SPM has been to identify the frequently appearing sequences of items in the sequence database. But in some cases, it becomes necessary to represent the corresponding attribute of the items, such as the quantity of the items, and this leads to the Quantitative Sequential Pattern Mining (QSPM).

Although considerable progress has been achieved in QSP mining, some critical challenges remain to be addressed. Firstly, the sheer volume of discovered patterns is potentially overwhelming, and it is not clear to analysts what the most important patterns might be. Secondly, incorporating user-defined constraints, such as pattern length, duration, average quantity, or gap constraints, in a manner that is computationally efficient is a complex problem. Thirdly, the traditional "minimum support" approach to QSP mining is potentially tedious, requiring a trial-and-error approach to determine a suitable threshold that results in a manageable, yet interesting, set of patterns.

To address these above problems, we introduce a new approach for mining top-k quantitative sequential patterns with constraints. The key idea of this approach is to employ Constraint Programming (CP) which is a flexible and a powerful paradigm for solving combinatorial problems. It has proven its efficiency compared to classical approaches in several studies [1] [2] [3] [4] [5] [6].

Our proposal is based on the global constraint proposed in [7] for SPM problem with quantities, its filtering algorithm keep only consistent values allowing to verify simultaneously the subsequence relation and the minimum threshold constraint. In order to returns quickely the first k patterns with the highest support, we propose two search strategies allowing an efficient support raising. The serach process starts with a minimum support equal to 1, each time the value increases, it allows for a quickly pruning for the search space. Since our approach is based on CP, we can impose several constraints without touching the main algorithm. These constraints can be defined either on the pattern (e.g. length, membership, regular xepression) or on the quantities (e.g. sum, average).

Main Contributions

1. We formalize the Top-k Quantitative Sequential Pattern Mining problem under heterogeneous constraints.
2. We adapt a quantitative global CP constraint to support dynamic top-k pruning.
3. We propose two novel initialization strategies that accelerate support raising.

4. We empirically show that our approach scales favorably compared to TKS for large values of k on the considered datasets.
5. We show that CP enables the seamless integration of structural and quantitative constraints without algorithmic redesign.

The paper is organized as follows. In section 2, we explain the topk SPM on a real world example. Section 3 gives the necessary preliminaries to understand our proposal. SPM methods are discussed in Section 4. In section 5, we detail the different algorithms allowing to effitiently extract the topk frequent patterns. Section 7 reports the experimental results. Finally, in section 8, we conclude and draw some perspectives.

2 Illustrative Real-World Example

We illustrate the concept of top-k quantitative sequential pattern mining using a real-world *healthcare monitoring* scenario.

Consider a quantitative sequence database where each sequence represents the chronological administration of treatments to a patient during hospitalization. Each medical action is associated with a dosage quantity. Such data are commonly found in clinical pathways and electronic health records.

An example database is shown in Table 1, where dosages are expressed in milligrams.

Table 1 Example of a quantitative healthcare sequence database

Patient	Treatment Sequence
P_1	⟨ antibiotic(500), analgesic(200), anticoagulant(50) ⟩
P_2	⟨ antibiotic(500), anticoagulant(40) ⟩
P_3	⟨ analgesic(150), anticoagulant(50) ⟩
P_4	⟨ antibiotic(250), analgesic(200), anticoagulant(60) ⟩

In this example, a pattern is considered to be contained in a patient sequence if the same treatments appear in the same order and the dosage specified in the pattern is less than or equal to the dosage actually administered. This allows patterns to represent minimal dosage requirements that are satisfied by higher administered doses.

Table 2 presents all frequent quantitative sequential patterns extracted from the healthcare sequence database using a minimum support threshold of 2. Each pattern is described by its structure, its size (i.e., the number of treatments it contains), and its support, which corresponds to the number of patient sequences in which the pattern occurs.

The mining process starts by identifying frequent single-treatment patterns. The results show that anticoagulant treatments appear in all patient sequences, making them the most frequent single-item patterns. Antibiotics and analgesics also occur frequently, but with different dosage levels, reflecting variations in treatment intensity among patients.

The extraction is then extended to longer sequences by combining frequent treatments while preserving their temporal order. Several frequent two-step treatment

patterns emerge, such as the administration of an anticoagulant following an antibiotic or an analgesic. These patterns indicate common therapeutic progressions observed during patient care.

Finally, the mining process identifies the frequent pattern of three items: an antibiotic, followed by an analgesic, followed by an anticoagulant. This is because it is commonly used in the following order: an antibiotic is first followed by an analgesic, which is then followed by an anticoagulant in anticoagulant therapy.

Overall, the obtained quantitative sequential patterns represent the most recurrent treatments. Finally, the novelty of the database representation through the integration of dosage values represents an exactly accurate representation of the treatments, rather than the necessarily approximate representation offered by the sequential patterns.

Table 2 All frequent quantitative sequential patterns (minsup = 2)

ID	Pattern	Size	Support
1	⟨ antibiotic(250) ⟩	1	3
2	⟨ antibiotic(500) ⟩	1	2
3	⟨ analgesic(150) ⟩	1	3
4	⟨ analgesic(200) ⟩	1	2
5	⟨ anticoagulant(40) ⟩	1	4
6	⟨ anticoagulant(50) ⟩	1	3
7	⟨ antibiotic(250) → analgesic(200) ⟩	2	2
8	⟨ antibiotic(250) → anticoagulant(40) ⟩	2	3
9	⟨ antibiotic(500) → anticoagulant(40) ⟩	2	2
10	⟨ analgesic(150) → anticoagulant(50) ⟩	2	3
11	⟨ analgesic(200) → anticoagulant(50) ⟩	2	2
12	⟨ antibiotic(250) → analgesic(200) → anticoagulant(50) ⟩	3	2

Although the exhaustive mining of frequent quantitative sequential patterns could bring a full view of the occurring behaviors within a database, the results could be the generation of numerous patterns, some of which might be redundant to some others. As shown in Table 2, for a small healthcare database and a high minimum support threshold, a number of different size and dosage level patterns could be generated.

Clearly, this large number of patterns can cause analysis to become cumbersome for domain experts, who are mostly interested in discovering the behavior that is more representative or information-bearing. Besides that, the success of the mining process heavily depends on the appropriate specification of the minimum support threshold, which can require several adjustments.

To address these limitations and instead of generating all sequential patterns, we can fix the number of generated patterns to k such that these pattrens forms the best patterns according to a given measure value such as frequency (support). This helps ensure a more interpretable effect as a result of the extraction process while preserving the expressiveness offered by quantitative information and constraints.

In this paper, we propose an effitient CP-based method which enables to extract top-k sequential patterns with different constraint like length, membership and regular expressions.

sid	quantitative Sequence
1	$\langle a(500)b(200)c(50)\rangle$
2	$\langle a(500)c(40)\rangle$
3	$\langle b(150)c(50)\rangle$
4	$\langle a(250)b(200)c(60)\rangle$

Table 3 A quantitative sequence database example QDB_1.

3 Preliminaries and problem statement

First, we provide the basic definitions for sequential pattern mining in the context of sequences of items with quantities. Then, we give an example to illustrate the topk QSPM problem. Finally, we give an overview of constraint programming.

3.1 Theoretical Foundations of Quantitative Sequential Pattern Mining

Let $\mathcal{I}$ be a finite set of distinct *items* and Q a finite set of quantities. We call a quantitative item, each item i with its quantity q denoted as $i(q)$, where $i \in \mathcal{I}, q \in Q$. The quantity values of an item i are represented by $Q_i = \{q_{i1} \dots q_{ik}\}$ in an increasing order. A quantitative sequence s, denoted q-sequence, is an ordered list $\langle i_1(q_1)i_2(q_2)\dots i_n(q_n)\rangle$, where $i_j(q_j)$, $1 \leq j \leq n$, is an extended item with its quantity. n is called the length of the q-sequence s. A quantitative sequence database QDB is a set of tuples (sid, s), where sid is a sequence identifier and s a q-sequence denoted by $QDB[sid]$. Mining sequential patterns is based on the subsequence relation which is defined below taking into account the quantities of items:

Definition 1 (subsequence relation) A q-sequence $\alpha = \langle \alpha_1(q'_1)\dots\alpha_m(q'_m)\rangle$ is a subsequence of $s = \langle i_1(q_1)i_2(q_2)\dots i_n(q_n)\rangle$, denoted by $(\alpha \preceq s)$, if $m \leq n$ and there exist integers $1 \leq j_1 \leq \dots \leq j_m \leq n$, such that $(\alpha_j = i_{k_j})$ and $(q'_j \leq q_j)$ for all $1 \leq j \leq m$. We also say that α is contained in s or s is a super-sequence of α. A tuple (sid, s) contains a q-sequence α, if $\alpha \preceq s$.

For example, the sequence $\langle b(4)a(1)\rangle$ is a sub-sequence of $\langle c(2)b(5)c(3)a(1)\rangle$ but not of $\langle b(2)a(1)c(3)\rangle$.

The subsequence relation allows to define the cover of a q-sequence α which consists in all tuples in QDB containing α. Thus, the support of α is the cardinal of its cover: $\sup(\alpha)_{QDB} =\mid \{(sid, s) \in QDB \mid \alpha \preceq s\} \mid$. We can define now the problem of mining quantitative sequential patterns as follows:

Definition 2 (Quantitative sequential pattern mining (QSPM)) Given a quantitative sequence database QDB and a minimum support threshold *minsup*. The problem of quantitative sequential pattern mining (QSPM) is to find all q-patterns p such that $sup_{QDB}(p) \geq minsup$.

Example 1 Let us consider the healthcare sequence database given in Table 1. After a preprocessing step, this dataset is represented by the sequence database given in Table 3. QDB_1 contains three sequences where the set of items is $\mathcal{I} = \{a, b, c, d\}$ and the set of all quantities is $Q = \{500, 200, 50, 150, 250, 60\}$. The allowed quantities for each item are : $Q_a = \{250, 500\}, Q_b = \{200\}, Q_c = \{40, 50, 60\}$. The sequence $s = \langle a(500)b(200)c(50)\rangle$ has 3 quantitative items, we say that s is 3-length sequence. The q-pattern $p = \langle a(500)c(50)\rangle$ is a subsequence of s: $p \preceq s$. If we consider $minsup = 3$, only one q-sequences is extracted: $p = \langle a(250)b(200)c(50)\rangle$.

In this paper, we address the problem of mining quantitative sequential patterns under constraints using constraint programming (CP). Differently to ad-hoc methods, CP offers an easy way to the user to express many constraints in a declarative way without considering new implementations. In the next section, we give an overview of CP necessary to understand our CP-approach.

Let $f : \mathcal{P} \rightarrow \mathbb{R}^+$ be a scoring function that assigns a numerical value to each quantitative sequential pattern, such as the support, the total quantity, a weighted support, or any utility-based measure.

Definition 3 (Top-k QSPM with constraints) The *Top-k Quantitative Sequential Pattern Mining with constraints (Top-k QSPM)* problem consists of discovering a set

$$\mathcal{P}_k = \{p_1, p_2, \ldots, p_k\}$$

of quantitative sequential patterns such that:

1. Each pattern p_i is a valid quantitative subsequence of at least one sequence in QDB;
2. Each pattern satisfies the user-defined constraints (e.g., size, gap, quantity, membership, regular expression);
3. Patterns are ranked in decreasing order according to the score function f;
4. For any pattern $p \notin \mathcal{P}_k$, it holds that $f(p) \leq f(p_k)$, where p_k denotes the k-th ranked pattern.

In this work, the scoring function f is fixed to support.

3.2 Example

Considering the quantitative sequence database in Table 1, we extracted the top-3 quantitative sequential patterns using support as the scoring function, which are presented in Table 4.

Table 4 Top-3 quantitative sequential treatment patterns

Rank	Pattern	Support
1	⟨ anticoagulant(40) ⟩	4
2	⟨ antibiotic(250) → anticoagulant(40) ⟩	3
3	⟨ analgesic(150) → anticoagulant(50) ⟩	3

Rather than enumerating all frequent treatment sequences, the top-k approach highlights the most common and clinically relevant therapeutic pathways. In this example, anticoagulant administration appears as a key treatment, frequently occurring after antibiotics or analgesics with consistent dosage levels.

Such insights can support medical experts in understanding standard care protocols, detecting treatment regularities, and identifying deviations from typical clinical pathways.

3.3 Constraint programming Concepts

Constraint programming (CP). Constraint programming [8] is a declarative paradigm for solving combinatorial search problems by modeling them as Constraint Satisfaction Problem (CSP).A *CSP* consists of a set $\mathcal{X}$ of n variables, a domain $\mathcal{D}$ mapping each variable $X_i \in \mathcal{X}$ to a finite set of values $\mathcal{D}(X_i)$, and a set of constraints $\mathcal{C}$. An assignment σ is a mapping from variables in $\mathcal{X}$ to values in their domains. A constraint $C \in \mathcal{C}$ is a subset of the cartesian product of the domains of the variables that occur in C. The goal is to find an assignment such that all constraints are satisfied.

One of the key strengths of CP lies in the separation between problem modeling and solving. The user specifies what constitutes a solution through constraints, while the CP solver determines how to search for solutions efficiently. This separation allows complex constraints to be combined naturally and makes CP particularly suitable for pattern mining tasks involving multiple heterogeneous constraints.

Constraint Propagation and Search. CP solvers rely on the interplay between constraint propagation and search. Constraint propagation aims to reduce the search space by removing inconsistent values from variable domains. Each constraint is associated with a propagator, which enforces a local consistency property by filtering domains whenever variable assignments change.

The search process incrementally assigns values to variables, typically using backtracking. After each assignment, constraint propagation is triggered to prune infeasible values early. If a variable domain becomes empty, the solver backtracks to explore alternative assignments.

This tight integration of propagation and search enables CP solvers to efficiently explore large combinatorial spaces while avoiding redundant computations.

Global constraints. A global constraint is a high-level constraint that captures a recurring combinatorial structure involving many variables. Unlike simple constraints, global constraints are equipped with specialized propagators that exploit problem-specific properties to achieve stronger filtering.

In the context of sequential pattern mining, global constraints allow the mining process to be expressed declaratively, while the solver handles the complexity of pattern enumeration and pruning.

The design of global constraints for pattern mining problems has been extensively studied. In several works [4] [5] [6], the authors propose efficient global contraints to mine sequential patterns under constraints. Schaus et al. [9] introduced the CoverSize constraint for itemset mining, which exposes the cover size as a variable and uses reversible sparse bitsets for efficient filtering.

4 related works

Our work centers within three main areas of research, which are Quantitative Sequential Pattern Mining, Constraint-Based Pattern Mining, and Top-k Pattern Mining.

4.1 Sequential Pattern Mining and Its Extensions

The Sequential Pattern Mining [10] concept was first introduced by Agrawal and Srikant [11], thus leading to a plethora of research in mining sequential patterns from transactional databases. Most of the traditional sequential pattern mining algorithms such as GSP [12], cSpade [13], and PrefixSpan [14] are based on candidate generation and testing, as well as a pattern growth strategy. Nevertheless, such concepts are viewed as obsolete in addressing complex constraints and quantification.

A comprehensive technical review by Mazumdar and Sarma [15] covers the landscape of SPM algorithms and their applications, highlighting the shift from traditional frequent pattern mining toward constraint-based, quantitative, and top-k variants.

4.2 Quantitative Sequential Pattern Mining

In several real-world applications, it is possible that an item is associated with an attribute such as quantity. Quantitative Sequential Pattern Mining takes into account the quantitative data and makes it possible to mine patterns that are related to some quantities, times, or utility values by using techniques such as utility-based sequential mining, Weighted SPM, etc. These techniques are computationally expensive.

Methods like QSP-Miner [16] introduced techniques to discretize quantitative values into intervals and mine patterns where items are associated with these intervals. Subsequent research focused on efficiency, using methods like projection and indexing, and on different semantics for quantity aggregation (e.g., average, sum, min). However, these methods predominantly rely on the a-priori support-based framework and often lack a native, efficient mechanism for complex constraint integration beyond simple item or length restrictions.

4.3 Constraint-Based Pattern Mining

The use of constraints for mining relevant knowledge has been well explored; constraints have been categorized as anti-monotone (minimum support, max gap), monotone (minimum length), and tough (average, variance). The CP approach for solving mining problems is successful for a number of tasks: itemset mining, sequence mining, and even graph mining due to its expressive power. For instance, a "Decomposition-Based" strategy consists of techniques that represent each potential pattern as a combinatorial structure and use black-box generic CP solvers. However, a better solution for mining tasks is given by the Global Constraint approach that treats the mining process (e.g., the frequency) as a particular constraint for which dedicated algorithms exist. Our work is closely related to and builds upon a previously proposed global constraint for sequential pattern mining with constraints (e.g., given by regular expressions). As opposed to other approaches that use it for qualitative patterns

only, we address both quantitative values and the top-k problem by adapting it for quantitative values as well.

4.4 Top-k Pattern Mining

The first algorithm proposed to tackle the topk sequential pattern problem was called TSP [17]. It exploits the pattern growth approach proposed in PrefixSpan by introducing a two-stage pruning method to eliminate the search space using a database projection technique. Later, an efficient algorithm was proposed by Fournier-Viger et al. in [18], called TKS (Top-K Sequential pattern mining). This method implements the vertical database representation and the candidate generation approach of SPAM [19]. TKS also incorporates an extra list to keep track of the set of patterns that can be extended in the database for candidate generation. This extra list is utilized in the raising of support by the extension of the pattern that holds the highest support.

Pham et al. [20] proposed TKCS (Top-K Closed Sequences), an algorithm that addresses the inefficiency of TSP by using a vertical bitmap database representation and by storing the top-k closed sequences in ascending order of support to accelerate minsup raising.

Beyond frequency-based top-k methods, a recent line of work explores constraint programming for top-k pattern set mining. Douad et al. [21] proposed a compact CP model for extracting a maximally k-diverse set of closed itemsets. This work demonstrates the growing interest in declarative approaches for top-k pattern mining under non-frequency objectives. However, it targets *itemset* mining with *diversity* as the optimization criterion, whereas our work addresses *sequential* patterns with *quantitative* attributes and *support* as the scoring function. The two approaches are therefore complementary: Douad et al. focus on set-level diversity in itemset mining, while QPP-TOPK focuses on sequence-level quantitative semantics in sequential pattern mining.

A recent work by Rizvee et al. [22] proposed a tree-based framework for mining top-K closed sequential patterns. Their approach addresses the challenge of reducing the number of mined patterns while eliminating the dependency on the support threshold. Unlike our approach, which handles quantitative attributes and leverages CP, Rizvee et al. focus on closed patterns and use a tree-based data structure.

Lan et al.proposed in [23] TK-RNSP, which is the first algorithm for mining Top-K Repetitive Negative Sequential Patterns (RNSPs). Their approach addresses the challenge of setting an appropriate support threshold by introducing anti-monotonicity definitions and a bitmap-based Depth-First Backtracking Search (DFBS) strategy. Unlike our approach, which handles quantitative attributes and leverages CP, Lan et al. focus on negative and repetitive patterns with bitwise operations.

Seq2Pat [24] offers a new approach for constraining sequence pattern mining that surpasses other approaches that are based on frequency constraints. Importantly, the approach formulates sequence pattern mining as a solution optimization problem that can be efficiently handled with an A* algorithm. Here, users have the ability to encode various constraints that are non-monotone on item attributes, such as price or category. This ensures that the mined top-k patterns are not just constrained based on frequency, but have a meaning in terms of given constraints.

What we propose in this paper is an approach that offers complementary strengths in terms of constraint expressiveness and quantitative semantics. We leverage the global constraint proposed for sequential pattern mining with quantity to generate only the top-k patterns with respect to frequency.

Why not Seq2Pat? Seq2Pat [24] is, to the best of our knowledge, the closest existing approach to ours, as it also addresses top-k sequential pattern mining under non-monotone constraints. However, three fundamental differences prevent a direct methodological comparison and justify our distinct design choices.

First, the solving paradigm differs. Seq2Pat formulates the mining task as a shortest-path optimization problem solved by an A* search algorithm, whereas our approach relies on Constraint Programming with a dedicated global constraint. This distinction matters in practice: A* requires the definition of an admissible heuristic tailored to the constraint set, which becomes non-trivial when heterogeneous structural and quantitative constraints are combined. In contrast, CP propagators compose naturally, allowing new constraints to be added declaratively without redesigning the search procedure.

Second, the role of quantities differs fundamentally. Seq2Pat treats quantitative attributes (e.g., price, category) as constraint attributes used to filter candidate patterns a posteriori, but does not embed them as first-class components of the mined pattern itself. In our framework, quantities are structural components of the pattern: each item is paired with its quantity in the pattern representation, and the subsequence relation explicitly compares quantities (Definition 1). This enables the discovery of patterns whose semantics depend on quantitative values ? such as dosage-dependent therapeutic pathways ? which Seq2Pat cannot express.

Third, the support semantics differ. Seq2Pat does not natively support a quantitative support measure: its objective function is defined over frequency and constraint satisfaction, not over quantitative aggregation. Our approach, by contrast, integrates the quantity directly into the frequency computation through the Q-Prefix-Projection propagator, which simultaneously enforces subsequence embedding and support filtering over quantitative sequences.

For these reasons, Seq2Pat and QPP-TOPK are complementary rather than directly comparable: Seq2Pat excels at top-k mining under complex non-monotone constraints on item attributes, while QPP-TOPK targets scenarios where quantities are integral to the pattern semantics and where declarative constraint composition is paramount. We therefore position QPP-TOPK as an approach that complements existing state-of-the-art methods by extending the expressive power of CP-based pattern mining to the quantitative top-k setting.

Having positioned our work with respect to both top-k and quantitative approaches, we now turn to the CP-based literature that underpins our modeling choices.

4.5 Constraint Programming for Sequential Patterns

Constraint-based data mining involves using user-defined constraints like length, membership, gap, and regular expression constraints. This problem has been solved in early literature using specialized techniques for pruning and itemset-based solvers.

Constraint Programming has, in recent years, proven to be a robust paradigm for declarative pattern mining, as it provides a mechanism for a homogeneous representation of complex constraints [2] [3] [4] [5] [6]. The use of Constraint Programming for mining patterns in a sequence introduced variables for structure in patterns and used global constraints to maintain the subsequence embedding relationship. Nevertheless, using these approaches for mining top-k patterns with quantitative data has not been sufficiently examined.

4.6 Position of Our Work

In [7], the authors propose a global constraint that allows to handle quantitative sequential datasets effitientlly. In this paper, we adopt and significantly adapt this global constraint to handle top-k sequential quantitative sequences by intergrating a dynamic k-th best support bound for top-k search.

Table 5 summarizes the positioning of our work with respect to representative state-of-the-art approaches along four dimensions: support for quantitative attributes, top-k extraction, structural and quantitative constraints and solving paradigm.

Approach	**Quant.**	**Top-k**	**Constraints**	**Paradigm**
GSP [12]	×	×	∼	Apriori
PrefixSpan [14]	×	×	∼	Pattern-growth
cSpade [13]	×	×	∼	Vertical
TKS [18]	×	✓	∼	Vertical/bitmap
QSP-Miner [16]	✓	×	∼	Apriori
Aoga et al. [6]	×	×	✓	**CP**
PP[25]	×	✓	✓	**CP**
Seq2Pat [24]	∼	✓	✓	A*
QPP[7]	✓	×	✓	CP
QPP-TOPK (ours)	✓	✓	✓	**CP**

Table 5 Comparison of representative approaches for sequential pattern mining. ✓ = supported; × = not supported; ∼ = partially supported.

As shown in Table 5, existing methods address at most three of these dimensions simultaneously. Traditional SPM algorithms (GSP, PrefixSpan, cSpade) and top-k extensions (TKS, TSP) do not natively handle quantitative attributes. Quantitative approaches (QSP-Miner) and constraint-based CP approaches (PP, Aoga et al. and QPP) each address one dimension of the problem but not both simultaneously. Seq2Pat is the closest approach in terms of constraints and top-k support, but treats quantities as constraint attributes rather than structural components of the pattern (see the discussion in Section 4.4). QPP-TOPK fills this gap by embedding quantities directly into the pattern representation and by integrating a dynamic top-k support bound into a global CP constraint.

This integration constitutes the core mechanism of our solver and enables effective top-k pruning within a declarative CP framework which allows to handle efficientlly different constraints defined both on patterns and quantities.

5 A CP-based approach for TOPK-QSPM

In this work, we are interested by extracting top-k patterns in term of frequency by leveraging the global constraint, called Q-Prefix-Projection, proposed in [7]. In that paper, the authors propose a CSP model that makes it possible to extract sequential patterns with quantities while guaranteeing the frequency constraint and even other constraints such as size, gap, membership, and regular expression.

In this model, a quantitative pattern is represented using pairs of variables: one variable for the item and one for its associated quantity. Thus, a pattern of maximum length ℓ is encoded by $2*\ell$ variables $P = \langle P_1, P_2, \ldots, P_{2\times\ell}\rangle$, alternating between items and quantities. Empty symbols $\square$ are used to represent the end of a sequence, ensuring variable-length patterns within a fixed-size representation. Logical consistency rules enforce that an item and its quantity are either both present or both absent.

The subsequence relation as well as the minimum support constraint is ensured by the global constraint Q-Prefix-Projection proposed in [7], its definition is given bellow.

Definition 4 (Q-Prefix-Projection global constraint) Let $P = \langle P_1, P_2, \ldots, P_{2\times\ell}\rangle$ be a q-pattern of size ℓ. $\langle d_1, ..., d_{2\times\ell}\rangle \in D(P_1)\times\ldots\times D(P_{2\times\ell})$ is a solution of Q-Prefix-Projection $(P, QDB, minsup)$ iff $sup_{QDB}(\langle d_1(d_2)d_3(d_4)...d_{(2\ell-1)}(d_{2\ell})\rangle) \geq minsup$.

Example 2 Consider the sequence database of Table 3 with $minsup = 2$ and $\ell = 2$. Let $P = \langle P_1, P_2, P_3, P_4\rangle$ with $D(P_1) = \mathcal{I}$, $D(P_2) = Q$, $D(P_3) = D(P_5) = \mathcal{I} \cup \{\square\}$ and $D(P_4) = D(P_6) = Q \cup \{\square\}$. Suppose that $\sigma(P_1) = a$, $\sigma(P_2) = 500$, $\sigma(P_3) = c$ and $\sigma(P_4) = 40$. Since $sup_{QDB}(\langle a(500)c(40)\rangle) = 2$, the Q-Prefix-Projection holds.

5.1 The principle algorithm

Generally, the process of extracting top-k patterns takes place in two steps: 1) initializing the list of top-k patterns by the k first generated patterns, and 2) Exploring other best patterns and updating the support during the search. This approach is described in Algorithm 1.

The algorithm 1 takes as input a CSP model encoding the sequential pattern mining problem, a quantitative sequence database QSDB, a minimum pattern length ℓ_{min}, a maximum pattern length ℓ, and an integer k specifying the number of patterns to extract. The output is a set topkList containing the k most frequent sequential patterns whose length is at least ℓ_{min}.

Initially, the algorithm sets the internal minimum support threshold minsup to zero (line 1) and initializes an empty list of top-k patterns (line 2). It then invokes an initialization procedure that explores the CSP model under this relaxed support constraint in order to generate candidate patterns (line 3). During the search, each valid

Algorithm 1: Extract-TopK($csp, QDB, \ell_{min}, \ell, k$)

Data: csp: CSP; QDB: initial quantitative database; $minsup$: internal variable for the minimum support threshold; ℓ_{min}: the minimum size of a pattern; ℓ: the maximum size of a pattern; $k : aninteger \geq 1$.
Result: $topkList$: the set of all top-k sequential patterns of minimal size ℓ_{min}.
begin

```
1  minsup ← 0 ;
2  topkList ← ∅ ;
3  topkList ← initialize-topk(csp, QDB, minsup, ℓ_min, ℓ, k) ;
4  if (|topkList| = k) then
5      explore-topk(csp, QDB, topk − list, ℓ_min, k) ;
6  return topkList
```

pattern discovered by the solver is evaluated and inserted into topkList.We propose two strategies to initialize the topk list (see Algorithm 2). Once the first k patterns are generated, the remaining patterns will be explored to keep only the best ones in terms of frequency (line 5).

5.2 Strategies for Initializing the topk List

When solving a CSP (exploring the search space), the key question is how to reduce the search space to speed up the solving process. For our TOPK-QSPM problem, this is primarily determined by the value of minsup; if minsup is large, it is possible to obtain the top-k patterns faster. This is where our idea comes from, and it is to manipulate the initialization of the top-k list with patterns that may have the highest support. In the algorithm 2, we introduce two strategies of initializing the *topkList*:

1. **Base-Line Strategy.** The pseudo-code of the base-line strategy is provided by the function `initialize-topk-BL` (Algorithm 2). This strategy enables the initialization of the topkList with the first sequential patterns found during the execution of a depth-first search in the search space. As soon as a new sequential pattern is found (lines 2), it is added to the pattern list *topkList*, which is sorted according to support. Once k patterns are generated or there is no pattern left, the variable *minsup* is updated with the minimum support of patterns in *topkList* (lines 5,6). Nevertheless, the base-line strategy has a very bad performance because of the size of the search space. It is therefore crucial to give priority to patterns with a high support value as soon as possible, in order to increase the minsup value more quickly.
2. **Increasing-Length Strategy**. The pseudo-code of the increasing length strategy is provided by the function `initialize-topk-IL` (Algorithm 2).

   To speed up the increase of `minsup`, a more effective strategy consists in mining the most promising patterns first so that the high-support patterns can be discovered early and used to prune the low-support patterns. We introduce a novel approach that starts with patterns of small size and then increases the length of sequential patterns until k patterns are discovered. The motivation of this strategy is that the shorter patterns are more likely to be frequent in the sequence database.

This strategy follows a level-wise exploration scheme, where each iteration generates new patterns by increasing the length of previously discovered ones (line 17). This strategy starts with the minimum pattern length ℓ_{min} (line 7) and iteratively searches patterns of increasing length until k patterns are discovered. At each iteration, new patterns are added to the top-k list (line 11), and the minimum support threshold (minsup) is dynamically changed to the support of the least frequent pattern in the current top-k list (lines 15,16). At the end of the initialization, ℓ_{min} is updated to $length+1$ (line 18). This ensures that the next function `explore-topk` (Algorithm 1) will explore patterns that were not already examined during the initialization phase. This early identification of high-support patterns enables effective pruning of low-support patterns and provides a strong initialization for subsequent top-k mining procedures.

Algorithm 2: Strategies for initializing *topkList* with the k first generated patterns

```
   Function initialize-topk-BL (csp, QDB, minsup, ℓ_min, ℓ, k) ;
   begin
1      for i ← 1 to k do
2          if (P = getNextSolution(csp, QDB, minsup, ℓ_min, ℓ) then
3              add P to topkList ;
4      5   else minsup ← minimum support of patterns in topkList ;
           return topkList ;
           ;
6      minsup ←minimum support of patterns in topkList ;
       return topkList
   Function initialize-topk-IL (csp, QDB, minsup, ℓ_min, ℓ, k) ;
   begin
7      length = ℓ_min ;
8      while (|topkList| < k do
9          while (P = getNextSolution(csp, QDB, minsup, length, length)) do
10             if (|topkList| < k) then
11                 add P to topkList ;
12             else
13                 if support(P) > minsup then
14                     add P to topkList ;
15                     minsup ← support(P);
16                 minsup ← support(P) ;
17         length ← lenght + 1 ;
18     ℓ_min ← length ;
       return topkList
```

Algorithm 3: Exploring the best patterns in terms of frequency

```
   Procedure explore-topk (Φ, QDB, topkList, ℓ_min, k) ;
   begin
1      Set minsup to the lowest support of patterns in topkList ;
2      while (P = solverNext(csp, QDB, minsup) ∧ |P| ≥ ℓ_min) do
3          add (P,topkList) ;
4          topk-update (topkList, P, minsup, k) ;
5          Set minsup to the lowest support of patterns in topkList;
   Procedure topk-update (topkList, P, minsup, k) ;
   begin
6      if (sup(P) > minsup) then
7          while ((|topkList| > k) and (∃ P' ∈ topkList ∧ sup(P') = minsup)) do
8              remove (P',topkList) ;
```

5.3 Explore other patterns

Once k patterns have been collected in the initialization step (`initialize-topk` function), the algorithm dynamically updates the support threshold minsup to the support of the current k-th most frequent pattern. This updated threshold is added as a constraint to the CSP model, allowing the solver to prune all future patterns whose support cannot exceed the current top-k boundary. As a result, the search space is progressively reduced while guaranteeing that only patterns potentially belonging to the top-k set are explored.

This process continues until the CSP solver has exhausted the search space under the current constraints. At termination, topkList contains all the k most frequent quantitative sequential patterns satisfying the minimum length constraint. When several patterns share the support of the $k - th$ pattern, they will all be saved in an additional list. The use of constraint programming allows the algorithm to naturally integrate frequency, length, and quantity constraints, while benefiting from constraint propagation to efficiently prune non-promising candidates.

Theorem 1 (Completeness) *QPP-TOPK-IL returns exactly the top-k quantitative sequential patterns with the highest support that satisfy the given constraints, provided that the search space is fully explored.*

Proof (sketch): By induction on the dynamic minimum support bound (minsup). At each update, no pattern whose support exceeds minsup is discarded, since the propagator only filters out values that are inconsistent with the current minsup value. Termination is guaranteed by the monotonically decreasing nature of minsup.

6 Modeling Constraints

In this section, we show how to model several constraints on the extracted patterns. We can distinguish two kinds of constraints:

1. Item constraints which are imposed on the pattern structure.
2. Quantities constraints which are imposed on the quantity variables.

Size constraints. Our CP modeling allow to control straightforwardly the minimum and maximum length of the sequence as well as the length of elements, using the predefined solver constraints.

$$minSize(P, \ell_{min}) \equiv \bigwedge_{i=1}^{i=2\times\ell_{min}} (P_{i,1} \neq \square)$$

$$maxSize(P, \ell_{max}) \equiv \bigwedge_{i=2\times\ell_{max}+1}^{i=2\times\ell} (P_{i,1} = \square)$$

Membership constraint. This constraint can be imposed either on the pattern or the quantities. We can use the among global contraints as follow :

Let V be a subset of items, l and u two integers s.t. $0 \leq l \leq u \leq \ell$.

$$item(P, V) \equiv \bigwedge_{v \in V} \texttt{Among}(P, \{v\}, l, u)$$

enforces that items of V should occur at least l times and at most u times in P. To forbid items of V to occur in P, l and u must be set to 0.

Our CP modeling allows also to impose this constraint on some item of position i in the pattern with a specified value v, since the variable encoding allows to capture the position of each item in the sequential pattern:

$$item(P_i, v) \equiv \texttt{Among}(P_i, v, 1, 1)$$

Regular expression constraint.

In general, this constraint is imposed on the extracted pattern and not on the quantities. let A_{reg} be the deterministic finite automaton encoding the regular expression exp. Let P' be the variables allowing to capture all item variables (i.e. $P' = \langle P_1, P_3, P_5, \ldots, P_{2*\ell-1} \rangle$).

$$reg(P', exp) \equiv \texttt{Regular}(P', A_{reg})$$

To the best of our knowledge, the more efficient ad hoc method to handle regular expressions is SMA [26]. Unfortunately, this method doesn't allow such constraint on sequences with quantities.

Other constraints on quantities. We can impose different constraints on variable quantities using mathematical constraints. In order to give examples, we have to take a real database like datasets extracted from an e-commerce. We can analyze these kind of datasets by asking various questions for a given *minsup* value.

1. Is there customers who buy less than M products in the same time ?
2. Is there customers who buy only one item for each product ?
3. Is there customers who buy only 3 products with the quantities 1, 2, 3 respectively ?

In practice, CP enables us to easily answer these questions, as it allows constraints to be directly expressed and incorporated into the model without requiring additional implementations.

The above constraints can be expressed as follows:

1. $const.1 \equiv \sum_{i=2}^{i=2\times\ell}(i\%2 = 0)(P_i < M)$
2. $const.2 \equiv \bigwedge_{i=2}^{i=2\times\ell}(i\%2 = 0)(P_i = 1)$
3. $$const.3 \equiv (\bigwedge_{i=7}^{i=2\times\ell} P_i = \square) \wedge (P_2 = 1, P_4 = 2, P_6 = 3)$$

7 Experiments

Our approach was carried out using the `gecode` solver[1]. All experiments were conducted on a processor Intel core i5-8365U with 16 GB of memory. A time limit of 1800 seconds has been set. If an approach is not able to complete the extraction within the time limit, it will be reported as $(-)$. ℓ was set to the length of the longest sequence of QDB. The datasets [27] considered in these experiments have different characteristics and represent different type of data (see Table 6). Since these datasets contain items without quantities, we add a random quantity to each item between 1 and 10.

dataset	$\mid QDB \mid$	$\mid \mathcal{I} \mid$	avg $\mid s \mid$	$\max_{s \in QDB} \mid s \mid$	type of data
Sign	730	267	51.99	94	sign language utterances
Leviathen	5834	9025	33.81	100	book
FIFA	20450	2990	34.74	100	web click stream
MSNBC	31790	17	13.33	100	web click stream
BIBLE	36369	13905	21.64	100	bible
Kosarak	69999	21144	7.97	796	web click stream

Table 6 Dataset Characteristics.

In order to evaluate our proposed approach `QPP-TOPK-IL`, we performed the following experimentation:

1. We compare our three approaches: `QPP-TOPK-IL`, `QPP-TOPK-BL` and `QPP-TOPK-NAIVE`,
2. We study the influence of item quantities on `QPP-TOPK-IL`,
3. We compare `QPP-TOPK` (when all quantities are equal to 1) with the state-of-the-art algorithms for top-k SPM, `TKS` [2], for different values of k,
4. We show the ability of our approach to handle different kinds of constraints.

7.1 Comparing the perfomance of our approches: `QPP-TOPK-IL` vs. `QPP-TOPK-BL` and `QPP-TOPK-NAIVE`

In this first experiment, we compare our three approaches:

[1] http://www.gecode.org
[2] http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/

1. QPP-TOPK-IL: this approach performs an iterative increasing length to generate the first k pattrens, then a depth-first traversal to explore the rest of the search space,
2. QPP-TOPK-BL: this approache perform a depth- first traversal of the search space to generate the top-k patterns,
3. QPP-TOPK-NAIVE: this is the naive approche which explore the complete set of patterns, and then, generate the top-k in the post-processing step.

Results are shown in Table 7 in terms of CPU times. First, the comparative analysis of the three methods shows that QPP-TOPK-IL is consistently the most efficient and the most robust: it is the only one that delivers results for all configurations (6 datasets, k ranging from 100 to 3000), with the shortest execution times. the QPP-TOPK-BL method is approximately 1.5 to 3 times slower and exhibits significant scalability limitations (failing on several datasets starting at k = 1000 or 2000). Finally, the QPP-TOPK-NAIVE method is completely non-functional (no results produced). Therefore, QPP-TOPK-IL clearly stands out as the preferred solution due to its speed and scalability. It is not only faster than QPP-TOPK-BL in all cases, but it is also capable of handling all the tested configurations, unlike the other two methods, which exhibit significant scalability limitations.

Table 7 Comparing QPP-TOPK-IL with QPP-TOPK-BL and QPP-TOPK-NAIVE in terms of CPU time.

Dataset	k parameter	CPU times (s)		
		QPP-TOPK-IL	QPP-TOPK-BL	QPP-TOPK-NAIVE
SIGN	100	1.11	1.75	–
	1000	6.62	9.24	–
	2000	11.63	17.58	–
	3000	20.93	26.52	–
LEVIATHAN	100	7.90	25.89	–
	1000	56.10	–	–
	2000	96.51	–	–
	3000	133	–	–
FIFA	100	17.95	–	–
	1000	160.78	–	–
	2000	310.45	–	–
	3000	443.34	–	–
MSNBC	100	9.51	15.87	–
	1000	51.89	88.80	–
	2000	82.49	161.01	–
	3000	112.97	–	–
BIBLE	100	19.38	43.99	–
	1000	100.06	314.10	–
	2000	160.39	–	–
	3000	215.09	–	–
KOSARAK	100	14.26	20	–
	1000	91.33	128.27	–
	2000	144.26	204.66	–
	3000	191.74	–	–

7.2 Influence of item quantities on QPP-TOPK-IL

The objective of this second experiment is to analyze the behavior of our approach when the quantity interval varies. So, we have fixed this interval to $[1, 10]$ for all the experimented datasets. Results are reported in Figure 1.

Based on these results, we observe that the QPP-TOPK-IL method exhibits an overall linear increase in execution time as the quantity parameter (ranging from 1 to 10) grows, although the slopes vary significantly depending on the dataset.

Small datasets such as SIGN and LEVIATHAN maintain very low execution times (under 5 seconds) even at quantity = 10, whereas larger datasets such as FIFA, BIBLE, and KOSARAK show a more pronounced growth, reaching approximately 80, 60, and 50 seconds respectively at quantity = 10.

The MSNBC dataset displays intermediate behavior, with a moderate progression up to about 25 seconds.

This hierarchy of execution times directly reflects the size and complexity of the different datasets, confirming that the computational cost of QPP-TOPK-IL increases with the requested workload, but in a fully predictable manner and without any combinatorial explosion.

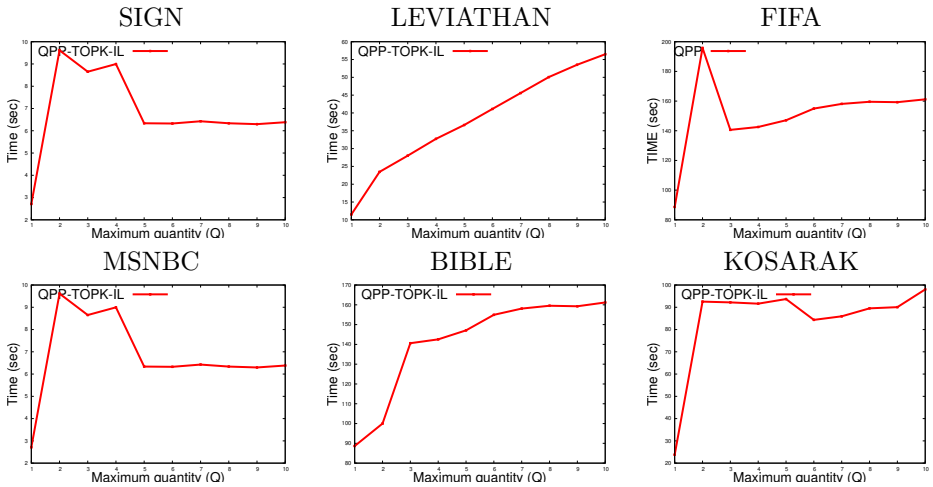


Fig. 1 CPU times when varying the quantity interval.

7.3 Performance comparison of QPP-TOPK-IL with TKS

In this section, we compare QPP-TOPK-IL with the state-of-the-art algorithm for top-k SPM, TKS [3], for different values of k. Since TKS does not take the quantity parameter into account, the quantity have been fixed to 1 for all datasets.

[3] http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/

The comparison of CPU times between QPP-TOPK-IL and TKS reveals a consistent advantage of QPP-TOPK-IL in terms of efficiency on the considered datasets.

For all datasets and for every value of k (from 100 to 3000), QPP-TOPK-IL is consistently faster on the tested datasets, with the performance gap widening as k increases. For example, on the FIFA dataset (the most computationally expensive one), QPP-TOPK-IL increases from 18 seconds ($k = 100$) to 443 seconds ($k = 3000$), whereas TKS literally explodes from 400 seconds to more than 3000 seconds.

On BIBLE and KOSARAK, TKS becomes prohibitive as early as $k = 2000$ (around 1500?2000 seconds), while QPP-TOPK-IL remains capped at approximately 200 seconds. Even on small datasets such as SIGN or LEVIATHAN, QPP-TOPK-IL is still 2 to 5 times faster.

These results indicates that QPP-TOPK-IL improved scalability on the tested configurations and represents a much more practical alternative for large values of k.

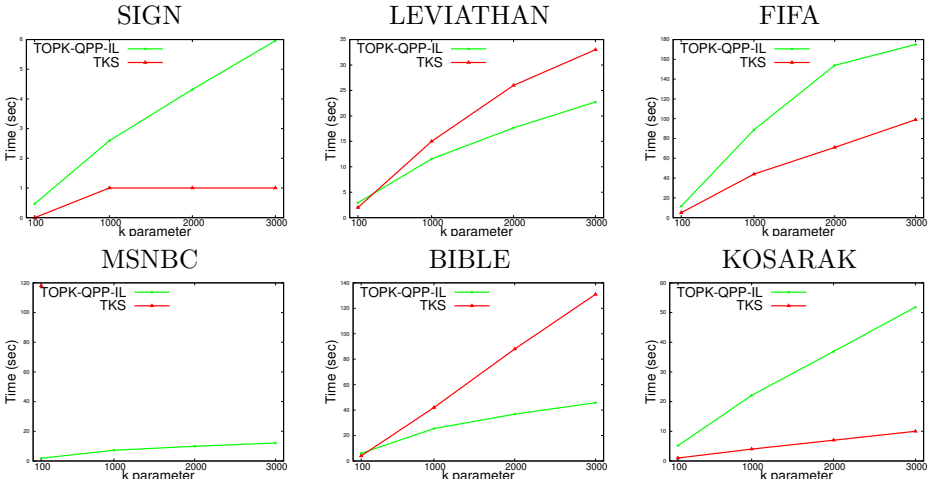


Fig. 2 QPP-TOPK-IL vs. TKS in terms of CPU times.

7.4 Extracting relevant top-k quantitative patterns under various constraints

In order to show the interest of our approach based on CP, we have considered the dataset FIFA containing clickstream data extracted from an e-commerce. In this case, we can consider that each item represents an item that was purchased by a client, following by its quantity.

Analysing such dataset, several questions can be formulated: What is are the 10 best frequent purchasing patterns where (1) some of items are fixed, (2) the number of items is fixed to N, (3) the total quantity of items exceeds (or lower than) M units? (4) the quantities of items follow a periodic expression, for example (1, 2, 3)*.

Effectively, we can obtain answers to these questions easily using CP which allows to express and to add constraints directly in the CP model without considering new implementations.

The constraints can be expressed as follows:

constraint 1.

The three first items are fixed to: 1,2,3.

$$P_1 = 1, P_3 = 2, P_4 = 3$$

constraint 2.

The minimum and maximum length contraints are modeled in Section 6. In order to satisfy this constraint we have to set: $\ell_{min} = \ell_{max} = 5$.

constraint 3.

a) $\sum_{i=1}^{i=\ell} P_{2*i} > M$***, we fix M to*** 10

b) $\sum_{i=1}^{i=\ell} P_{2*i} < M$***, we fix M to*** 10

constraint 4.

we can formulate the regular expression on the quantities as follows:

$$RE \equiv (1 + 2 + 3)^*$$

Constraint	Final minsup	CPU times (s)
No constraint	7827	5.41
Const1.	3026	2,57
Const.1 + const.2	1070	9.63
Const.1 + const.3a	1594	4.96
Const.1 + const.3b	3026	2.50
Const.1 + const.2 + const.3a	645	19.8
Const.1 + const.2 + const.3b	1070	4.33
Const.4	3509	31.19

Table 8 Extracting the top-1 pattern under constraints from the FIFA dataset.

Results are given in Table 8. We can do these observations: (1) when all constraints are ennable, the top-10 patterns are extratced quickely, (2) integrating constraints on pattren make the serach process harder, (3) the execution time when constraints are combined is variable; we cannot say that if such a constraint is imposed, the search process becomes faster. Finally, these results show that extracting top-k patterns remains computationally challenging. While constraints reduce the solution space semantically, they may restrict early support raising and therefore impact pruning efficiency.

8 Conclusion

In this paper, we addressed the problem of mining top-k quantitative sequential patterns under heterogeneous constraints. We proposed QPP-TOPK, a CP-based approach that adapts the Q-Prefix-Projection global constraint [11] to integrate a dynamic k-th best support bound, enabling effective top-k pruning during search. We introduced two initialization strategies (BL and IL), the latter exploiting an increasing-length exploration to raise the support threshold early. Extensive experiments on six real-world datasets demonstrated that (i) QPP-TOPK-IL consistently outperforms QPP-TOPK-BL and QPP-TOPK-NAIVE, (ii) QPP-TOPK-IL scales significantly better than TKS for large k, and (iii) CP enables the declarative expression of size, membership, quantity, and regular-expression constraints without algorithmic redesign. As future work, we plan to (a) extend QPP-TOPK to fuzzy and utility-based quantitative patterns, (b) investigate parallel CP search strategies, and (c) provide a public benchmark and reference implementation to foster reproducibility.

References

[1] Métivier, J.-P., Loudni, S., Charnois, T.: A constraint programming approach for mining sequential patterns in a sequence database. In: ECML/PKDD Workshop on Languages for Data Mining and Machine Learning (2013)

[2] Kemmar, A., Ugarte, W., Loudni, S., Charnois, T., Lebbah, Y., Boizumault, P., Crémilleux, B.: Mining relevant sequence patterns with cp-based framework. In: ICTAI, pp. 552–559 (2014)

[3] Novak, P.K., Lavrac, N., Webb, G.I.: Supervised descriptive rule discovery: A unifying survey of contrast set, emerging pattern and subgroup mining. Journal of Machine Learning Research **10** (2009)

[4] Négrevergne, B., Guns, T.: Constraint-based sequence mining using constraint programming. In: CPAIOR'15, pp. 288–305 (2015)

[5] Kemmar, A., Loudni, S., Lebbah, Y., Boizumault, P., Charnois, T.: A global constraint for mining sequential patterns with GAP constraint. CoRR **abs/1511.08350** (2015)

[6] Aoga, J.O.R., Guns, T., Schaus, P.: An efficient algorithm for mining frequent sequence with constraint programming. In: Machine Learning and Knowledge Discovery in Databases - European Conference, ECML PKDD 2016, Riva del Garda, Italy, September 19-23, 2016, Proceedings, Part II, pp. 315–330 (2016). https://doi.org/10.1007/978-3-319-46227-1_20 . https://doi.org/10.1007/978-3-319-46227-1_20

[7] Kemmar, A., Touati, C., Lebbah, Y.: A cp-based approach for mining sequential patterns with quantities. Inteligencia Artif. **26**(71), 1–12 (2023) https://doi.org/10.4114/INTARTIF.VOL26ISS71PP1-12

[8] Rossi, F., Beek, P., Walsh, T. (eds.): Handbook of Constraint Programming. Foundations of Artificial Intelligence, vol. 2. Elsevier, ??? (2006). https://www.sciencedirect.com/science/bookseries/15746526/2

[9] Schaus, P., Aoga, J.O.R., Guns, T.: Coversize: A global constraint for frequency-based itemset mining. In: Principles and Practice of Constraint Programming (CP 2017). Lecture Notes in Computer Science, vol. 10416, pp. 529–546. Springer, ??? (2017). https://doi.org/10.1007/978-3-319-66158-2_34

[10] Li, Y., Han, M., Ma, C., Xue, Y., Ding, J.: A survey of sequence pattern mining: From precise data to uncertain data. Neurocomputing **683**, 133529 (2026) https://doi.org/10.1016/j.neucom.2026.133529

[11] Agrawal, R., Srikant, R.: Mining sequential patterns. In: Yu, P.S., Chen, A.L.P. (eds.) ICDE, pp. 3–14. IEEE Computer Society, ??? (1995)

[12] Srikant, R., Agrawal, R.: Mining sequential patterns: Generalizations and performance improvements. In: EDBT, pp. 3–17 (1996)

[13] Zaki, M.J.: Sequence mining in categorical domains: Incorporating constraints. In: CIKM'00, pp. 422–429 (2000)

[14] Pei, J., Han, J., Mortazavi-Asl, B., Pinto, H., Chen, Q., Dayal, U., Hsu, M.: PrefixSpan: Mining sequential patterns by prefix-projected growth. In: ICDE, pp. 215–224. IEEE Computer Society, ??? (2001)

[15] Mazumdar, N., Sarma, P.K.D.: Sequential pattern mining algorithms and their applications: a technical review. International Journal of Data Science and Analytics **20**(3), 1683–1726 (2024) https://doi.org/10.1007/s41060-024-00659-x

[16] Fournier-Viger, P., Others: Qsp-miner: An algorithm for mining quantitative sequential patterns. Expert Systems with Applications **89**, 123–135 (2017)

[17] Tzvetkov, P., Yan, X., Han, J.: TSP: Mining Top-k closed sequential patterns. In: ICDM, pp. 347–354 (2003)

[18] Fournier-Viger, P., Gomariz, A., Gueniche, T., Mwamikazi, E., Thomas, R.: TKS: Efficient mining of Top-k sequential patterns. In: ADMA (1)'13, pp. 109–120 (2013)

[19] 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, pp. 429–435 (2002). https://doi.org/10.1145/775107.775109 . ACM

[20] Pham, T.-T., Do, T., Nguyen, A., Vo, B., Hong, T.-P.: An efficient method for mining top-k closed sequential patterns. IEEE Access **8**, 118156–118163 (2020)

https://doi.org/10.1109/ACCESS.2020.3004528

[21] Douad, M.E.A., Aribi, N., Loudni, S., Hien, A., Lebbah, Y.: Maximizing diversity in k-pattern set mining through constraint programming and entropy. Applied Intelligence **55**, 597 (2025) https://doi.org/10.1007/s10489-025-06482-6

[22] Rizvee, R.A., Ahmed, C.F., Leung, C.K.: A tree-based framework to mine top-k closed sequential patterns. Applied Intelligence **55**(3), 221 (2025) https://doi.org/10.1007/s10489-024-06137-y

[23] Lan, D., Sun, C., Dong, X., Qiu, P., Gong, Y., Liu, X., Fournier-Viger, P., Zhang, C.: Tk-rnsp: Efficient top-k repetitive negative sequential pattern mining. Information Processing and Management **62**(3), 104077 (2025) https://doi.org/10.1016/j.ipm.2025.104077

[24] Wang, X., Hosseininasab, A., Colunga, P., Kad?o?lu, S., Hoeve, W.-J.: Seq2pat: Sequence-to-pattern generation for constraint-based sequential pattern mining. Proceedings of the AAAI Conference on Artificial Intelligence **36**, 12665–12671 (2022) https://doi.org/10.1609/aaai.v36i11.21542

[25] Kemmar, A., Lebbah, Y., Loudni, S., Boizumault, P., Charnois, T.: Prefix-projection global constraint and top-k approach for sequential pattern mining. Constraints An Int. J. **22**(2), 265–306 (2017) https://doi.org/10.1007/S10601-016-9252-Z

[26] Trasarti, R., Bonchi, F., Goethals, B.: Sequence mining automata: A new technique for mining frequent sequences under regular expressions. In: ICDM'08, pp. 1061–1066 (2008)

[27] Fournier-Viger: SPMF: A java open-source data mining library (available on line at http://www.philippe-fournier-viger.com/spmf/). (2013)

Supplementary Files

This is a list of supplementary files associated with this preprint. Click to download.

- miningTOPK.zip