Inteligencia Artificial, 26(71) (2023), 1-12
doi: 10.4114/intartif.vol26iss71pp1-12

INTELIGENCIA ARTIFICIAL

http://journal.iberamia.org/

A CP-based approach for mining sequential patterns with quantities

Amina Kemmar[1,2], Chahira Touati[2] and Yahia Lebbah[2]
[1] Oran Graduate School of Economics, BP 65 CH 2 Achaba Hnifi - USTO, Oran, Algeria
kemmar.amina@gmail.com
[2] LITIO, University of Oran 1 Ahmed Ben Bella, Oran, Algeria
touati.chahira@univ-oran1.dz
ylebbah@gmail.com

Abstract This paper addresses the problem of mining sequential patterns (SPM) from data represented as a set of sequences. In this work, we are interested in sequences of items in which each item is associated with its quantity. To the best of our knowledge, existing approaches don't allow to handle this kind of sequences under constraints. In the other hand, several proposals show the efficiency of constraint programming (CP) to solve SPM problem dealing with several kind of constraints. However, in this paper, we propose the global constraint QSPM which is an extension of the two CP-based approaches proposed in [5] and [7]. Experiments on real-life datasets show the efficiency of our approach allowing to specify many constraints like size, membership and regular expression constraints.

Keywords: Sequential pattern mining, quantitative sequences, constraint programming, constraints.

1 Introduction

In recent years, the amount of collected data in many domains grows more and more considerably, leading to the use of data mining techniques for a better analysis of these data. This paper addresses the problem of mining sequential patterns (SPM) from data represented as a set of sequences. SPM problems are found in many domains like healthcare, education, Web usage mining, bioinformatics and telecommunications. One of the classical applications is the market basket analysis for which the sequence database represents the purchases made by customers in a retail store. Each sequence represents the items purchased by a customer at different times. A sequence is an ordered list of itemsets (sets of items bought together). In this case, the item can be associated with its quantity or its price. For example, a customer can purchase two sweets, then, one pant and finally 4 socks, this sequence is represented by <sweet(2)pant(1)socks(4)>. Adding the quantity information can help to better analyse the behavior of customers. Usually, these information are ignored by the proposed approaches for SPM problems.

The SPM was first proposed in [1]. Since then, many efficient specialized approaches have been proposed: cSpade [18], SPIRIT [4], SMA [16], CloSpan [17] and Gap-BIDE [9] are both extensions of PrefixSpan [14] to mine closed frequent patterns and closed frequent patterns with gap constraints respectively. In order to offer to the user the possibility to add easily constraints on the extracted patterns, other approaches based on constraint programming (CP) were proposed [10] [6] [12] [11]. In [5], the authors have proposed the global constraint Prefix-Projection which performs better comparing to the proposed methods. Aoga et al. [2] have further extended this work by combining ideas from pattern mining as well as from CP. They improve the efficiency of the previous global constraint using (i) last-position lists technique similar to the LAPIN algorithm [19] and (ii) ideas from trailing CP solvers to avoid unnecessary copying. These two works don't allow to directly handle gap constraints. Thus, in [7], the authors proposed the global constraint GAP-SEQ enabling to handle the gap constraint combining with other types of constraints. Note that the cited CP approaches don't consider quantitative datasets.

In this paper, we tackle the SPM problem, under constraints, considering only sequences of items in which each item is associated with its quantity. We propose an extension of the two approaches proposed in [5] and [7] which are based on CP. To the best of our knowledge, there is only one ad-hoc approach [8] handling exactly this problem, in which the authors proposed Apriori-QSP and PrefixSpan-QSP which are respectively extensions of the apriori [1] and PrefixSpan [13] algorithms. The Apriori algorithm is the first algorithm proposed for SPM problem. It is based on a breath first search using two steps "join" and "prune" to reduce the search space. In the othen hand, PrefixSpan performs a depth-first search based on the prefix-projected database principle allowing to extend the current pattern considering only frequent items computing from the current projected database. This principle forms the basis of our contribution, more details are given in Section 3.1.

Since the above method [8] is limited by the fact that the user cannot add as many constraints as he wants to guide the mining process, a new declarative methods are required. For this reason, the objective of this paper is to: (1) propose a global constraint to SPM problem with quantities (denoted QSPM), its filtering algorithm keep only consistent values allowing to verify simultaneously the subsequence relation, minimum threshold and gap constraints, (2) show how our approach allows to specify many constraints like size, membership and regular expression constraints which can be also combined together, (3) perform many experiments on real-life datasets in order to understand the behavior of our approach considering quantities and show its effectiveness when several constraints are added.

The paper is organized as follows. Section 3 recalls preliminaries to understand the QSPM problem and constraint programming. Section 2 shows the interest of adding quantities to items in the sequence database. In section 4, we detail our global constraint by describing its filtering algorithm based on the principle of projected databases. Section 5 reports experiments we performed. Finally, we conclude and draw some perspectives in Section 6.

2 Motivating Example

In order to show the interest of adding quantities to sequences, let us take the market basket analysis application. In this one, we consider the purchases made by customers in a retail store. Each sequence represents the items purchased by a customer at different times (the different items are ordered following their purchase time). The set of all sequences forms a sequence database.

Customer Id	Customer Sequence
1	<sweet(2) pant(1) socks(4)>
2	<dress(1) pant(2) socks(2)>
3	<pant(1) hat(1) socks(3) scarf(2)>

Table 1: Customer purchases represented as sequences with quantities

An example is given in Table 1. In this sequence database, the first customer purchases sweets, then, pant and finally socks, this sequence is represented by <sweet pant socks>. The sequential pattern mining problem consists to extract sequences that appears at least *minsup* time in the database. But, in some situations, we need to store also the quantity of each item (i.e. sweet(2)) in order to analyze the customer's behavior and also to be able to take some decisions. If we fix *minsup* to 3 and imposing quantities greater than 1, only one pattern will be extracted: <socks(2)>.

In order to increase sales, we can be interested in several questions on quantities: (1) Is there customers who buy more than N products in the same time? (2) Is there customers who buy only one item for each product ? (3) Is there customers who buy 3 products with specified quantities? Using classical methods, it will be difficult to get answers without modifying their implementations. However, in this paper, we

sid	quantitative Sequence
1	$\langle b(4)a(1)c(3)\rangle$
2	$\langle b(2)a(2)b(6)d(2)\rangle$
3	$\langle b(3)a(2)c(2)\rangle$

Table 2: A quantitative sequence database example QDB_1.

propose a CP-based approach (detailed in Section 4.1) which allows to extract quantitative sequential patterns verifying different constraints: size, membership, gap, regular expression and other constraints.

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 present the concept of projected-databases, introduced in PrefixSpan algorithm [13]. Finally, we give an overview of constraint programming.

3.1 Sequential patterns background

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 $i_j(q_j)$, $1 \le j \le n$, 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 $m \le n$ and there exist integers $1 \le j_1 \le \dots \le j_m \le n$, such that ($\alpha_j = i_{k_j}$) and ($q'_j \le q_j$) for all $1 \le j \le m$. 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 $\alpha \preceq s$.*

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 $sup_{QDB}(p) \ge minsup$.*

Instead of considering only the minimum support constraint, this paper address the problem of extraction patterns verifying other interesting constraints like size, membership, gap and regular expression constraints.

Example 1 *Let us consider the sequence database QDB_1 given in Table 2. QDB_1 contains three sequences where the set of items is $\mathcal{I} = \{a, b, c, d\}$ and the set of all quantities is $Q = \{1, 2, 3, 4, 6\}$. The allowed quantities for each item are : $Q_a = \{1, 3\}, Q_b = \{2, 3, 4, 6\}, Q_c = \{2, 3\}, Q_d = \{2\}$. The sequence $s = \langle a(1)c(2)\rangle$ has 2 quantitative items, we say that s is 2-length sequence. The q-pattern $p = \langle a(1)c(1)\rangle$ is a subsequence of s: $p{\preceq}s$. If we consider $minsup = 2$, 14 q-squences are extracted, the result of the mining process with details is given in Table3.*

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.

q-sequence	cover	support
$\langle b(2)\rangle$, $\langle b(2)a(1)\rangle$, $\langle b(3)\rangle$, $\langle a(1)\rangle$	$(1, s_1), (2, s_2), (3, s_3)$	3
$\langle b(2)a(1)c(2)\rangle$, $\langle c(2)\rangle$, $\langle a(1)c(2)\rangle$, $\langle b(2)c(2)\rangle$, $\langle b(3)c(2)\rangle$, $\langle b(3)a(1)\rangle$	$(1, s_1), (3, s_3)$	2
$\langle b(2)a(2)\rangle$, $\langle a(2)\rangle$	$(2, s_2), (3, s_3)$	2
$\langle b(4)\rangle$, $\langle a(1)\rangle$	$(1, s_1), (2, s_2)$	2

Table 3: Quantitative sequential patterns extracted from QDB_1 when $minsup = 2$.

3.2 Constraint programming Background

Constraint programming (CP) [15] is a powerful paradigm for solving combinatorial search problems modeled as constraints. It is based on the following principle: (1) the user specifies the problem in a declarative way as a constraint satisfaction problem (CSP); (2) the solver looks for the complete and correct set of solutions to the problem. In this way, the problem specification is separated from the search strategy.

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.

Example 2 *Let be the following CSP:*

$$\begin{cases} \mathcal{X} = \{X_1, X_2, X_3\} \\ \mathcal{D}(X_1) = \mathcal{D}(X_2) = \mathcal{D}(X_3) = \{1, 2, 3\} \\ \mathcal{C} = \{C_1(X_1, X_2), C_2(X_1, X_3), C_3(X_2, X_3)\}, where, \end{cases}$$

$$\begin{array}{l} C_1(X_1, X_2) \equiv (X_1 \neq X_2) \\ C_1(X_1, X_3) \equiv (X_1 \neq X_3) \\ C_1(X_2, X_3) \equiv (X_2 \neq X_3) \end{array}$$

The above CSP admits three solutions: ($X_1 = 1, X_2 = 2, X_3 =$3), ($X_1 = 3, X_2 = 1, X_3 = 2$) and ($X_1 = 2, X_2 = 3, X_3 = 1$).

In CP, the resolution process consists in combining iteratively search and propagation phases. The search phase consists in enumerating all possible partial instantiations of variables until finding a solution or proving that no solution exists. The constraint propagation phase allows to reduce search space by filtering values from variable domains which can not participate in any solution for the CSP. Thus, each constraint is associated with a propagator (i.e. a filtering algorithm) for deleting all values from variable domains which do not satisfy this constraint. Since a variable can participate in several constraints, modifications on domains are propagated by activating the propagators of these constraints. This propagation process is repeated on all constraints until no filtering is possible or a variable domain becomes empty.

4 A Global Constraint for QSPM

The first CP-approach allowing to handle the sequential pattern mining problem with gap constraint was proposed in [7]. In this paper, we propose an extension of this work combined with the approach proposed in [5], in order to consider item quantities in the mining process ensuring the gap constraint, which should be implemented in the filtering algorithm. In this section, we first present our CSP modeleing considering quantities and then, we detail the filtering algorithm of our global constraint.

4.1 A CSP modeling for QSPM: variable and domains

For the SPM problem without quantities, the pattern P of length ℓ to be extracted is modeled with ℓ variables $\langle P_1, P_2, \ldots, P_\ell\rangle$ s.t. $\forall i \in [1 \ldots \ell], D(P_i) = \mathcal{I} \cup \{\Box\}$. For QSPM problem, since each item has a quantity, we modeled the unknown quantitative pattern with $2 \times \ell$ variables $\langle P_1, P_2, P_3, P_4, \ldots, P_{2\times\ell}\rangle$.

For each item in position i (an even position), we associate its quantity in the next one $i+1$ (an odd position). Both items and quantities are encoded as integers.

The symbol $\square$ ($\square \notin \mathcal{I}$) stands for an empty item (or an empty quantity) and denotes the end of the sequence. Let $FreqI$ and $FreqQ$ be the set of frequent items and frequent quantities respectivelly in the initial database. The domains of variables are defined as follows:

1. $D(P_1) = FreqI$ and $D(P_2) = FreqQ$ to avoid the empty sequence,
2. $\forall i \in \{3 \ldots 2 \times \ell\}$:
    - $D(P_i) = FreqI \cup \{\square\}$ if i is even,
    - $D(P_i) = FreqQ \cup \{\square\}$ if i is odd.

When the length of the unknown pattern is k ($k < \ell$), the last variables from the position $2 \times k + 1$ are filled with the symbol $\square$ as follows: $\forall j \in [2 \times k + 1 \ldots 2 \times \ell], (P_j = \square)$.

When an item variable is assigned to an empty symbol $\square$, its corresponding quantity variable is also assigned to $\square$. Otherwise, if the item variable is not empty, then its corresponding quantity variable can not be empty. We obtain the following two rules (i corresponds to the position of an item variable):

1. $P_i = \square \Rightarrow P_{i+1} = \square$,
2. $P_i \neq \square \Rightarrow P_{i+1} \neq \square$.

In the following, we give the definition of our global constraint called Q-Prefix-Projection, which is an extension of the global constraint proposed in [5] without considering gap constraint.

Definition 3 (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 $sup_{QDB}(\langle d_1(d_2)d_3(d_4)...d_{(2\times\ell)-1}(d_{2\times\ell})\rangle) \geq minsup$.*

Example 3 *Consider the sequence database of Table 2 with $minsup = 2$ and $\ell = 3$. Let $P = \langle P_1, P_2, P_3, P_4, P_5, P_6\rangle$ 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 $\sigma(P_1) = b$, $\sigma(P_2) = 3$, $\sigma(P_3) = a$, $\sigma(P_4) = 1$, $\sigma(P_5) = c$ and $\sigma(P_5) = 2$. Since $sup_{QDB}(\langle b(3)a(1)c(2)\rangle) = 2$, the* Q-Prefix-Projection *holds.*

4.2 The filtering algorithm for Q-Prefix-Projection global constraint

Algorithm 2 describes the pseudo-code of the filtering algorithm of Q-Prefix-Projection global constraint. Since each item is associated with its quantity, we have to consider two types of filtering: (1) filtering the domains of item variables according to frequent items, (2) filtering the domains of quantity variables keeping only frequent quantities. These new domains allow to extend the current assignment (pattern) to form a new frequent pattern. The algorithm takes as input the quantitative database QDB, the minimum support threshold *minsup* and the current assignment σ. It uses three internal data structures: (1) $\mathcal{PSDB}$ to store projected databases, (2) $\mathcal{FI}$ to store the set of frequent items, and (3) $\mathcal{FQ}$ to store frequent quantities for each frequent item according to the current assignment σ. Each time a variable is assigned, the filtering algorithm is lunched, it starts by detecting the first variable not assigned P_i according to the lexicographic order. If the first quantity variable (position $i+1$) is not assigned, its domain is initialized with frequent quantities corresponding to the first item variable (lines 1-2). Since the length of the pattern is at least 1, from position 3, a variable can be instanciated to the empty symbol $\square$ which indicates the end of the sequence, in this case, all next variables are assigned in the same way to $\square$ (lignes 3-6). Otherwise, the projected database $\mathcal{PSDB}_i$ (the pseudo-code is given in Algorithm 2) is computed incrementally from an old one (line 7). Computing The size of $\mathcal{PSDB}_i$ (line 7) allows to determine if the current assignment can be extended to another one, or not (return false). In the first case, the locally frequent items ($\mathcal{FI}$), the internal structure $FreqQItems$ and the locally frequent quantities ($\mathcal{FQ}$) are computed (lines 10-11). Then, we have two cases: (1) $P_{(i+1)}$ corresponds to an item variable, which lead to filter domains of all next item variable according to $\mathcal{FI}$ (lines 14-16) and filter domains of all next quantity variable according to $\mathcal{FQ}$ (lines 17-19), (2) $P_{(i+1)}$ corresponds to a quantity variable, in this case, its domain is reduced according to frequent quantities obtained from the item $\sigma(P_i)$ (lines 20-22).

Algorithm 1: FILTER-QPP(QDB, σ, i, P, $minsup$)

```
Data: $QDB$: initial database; $\sigma$: current prefix $\langle\sigma(P_1),\dots,\sigma(P_i)\rangle$; $minsup$: the minimum support threshold; $\mathcal{PQDB}$:
      internal data structure of Q-PREFIX-PROJECTION for storing pseudo-projected databases; $FreqQItems$: internal
      data structures using a hash table storing frequent quantities for each frequent item; $\mathcal{FQ}$: locally frequent
      quantities; $\mathcal{FI}$: locally Frequent items.
begin
1     if $(i == 1)$ then
          /*The first quantity variable is not assigned*/ ;
2         $D(P_{i+1}) \leftarrow FreqQItems[\sigma(P_i)]$;
3     else if $(i \geq 3 \wedge \sigma(P_i) = \square)$ then
4         for $j \leftarrow i+1$ to $2 \times \ell$ do
5             $P_j \leftarrow \square$;
6         return True;
      else
7         $\mathcal{PSDB}_i \leftarrow$ PROJECTSDB$(QDB, \mathcal{PSDB}_{i-1}, \langle\sigma(P_i)\rangle)$;
8         if $(\#\mathcal{PSDB}_i < minsup)$ then
9             return False ;
          else
10            $\mathcal{FI} \leftarrow$ GETFREQITEMS$(QDB, \mathcal{PSDB}_i, minsup)$ ;
11            $FreqQItems \leftarrow$ GETFREQQFORITEMS$(QDB, \mathcal{PSDB}_i, minsup)$ ;
12            $\mathcal{FQ} \leftarrow$ GETFREQQ$(FreqQItems)$ ;
13            if $(i\%2 == 0)$ then
                  /*In this case, the last not assigned variable corresponds to an item variable: Filtering domain of the
                    next item variables */ ;
                  $j \leftarrow i+1$ ;
14                while $j <= 2 \times \ell$ do
15                    $D(P_j) \leftarrow (D(P_j) \cap \mathcal{FI}) \cup \{\square\}$;
16                    $j \leftarrow j+2$;
                  /* Filtering the domain of the next quantity variables */ ;
                  $j \leftarrow i+2$ ;
17                while $j <= 2 \times \ell$ do
18                    $D(P_j) \leftarrow (D(P_j) \cap \mathcal{FQ}) \cup \{\square\}$;
19                    $j \leftarrow j+2$;
20            else if $((i+1)\%2 == 0)$ then
                  /* Filtering the domain of the next quantity variable */ ;
21                foreach $a \in D(P_{i+1})$ $s.t.(a \neq \square \wedge a \notin FreqQItems[\sigma(P_i)])$ do
22                    $D(P_{i+1}) \leftarrow D(P_{i+1}) - \{a\}$;
23            return True;
```

A running example. Let us take the quantitative sequence database given in Table 2 with $minsup = 2$. Let $P = \langle P_1, P_2, P_3, P_4\rangle$ with $D(P_1) = \mathcal{I}$, $D(P_2) = Q$, $D(P_3) = \mathcal{I} \cup \{\square\}$ and $D(P_4) = Q \cup \{\square\}$.

Figure 1 depicts the search tree explored w.r.t. the filtering achieved by Q-PREFIX-PROJECTION global constraint (In each node, we have the frequent item with all its frequent quantities). We adopt a variable selection strategy based on the lexicographic ordering of variables: P_1 is assigned first, then P_2, then P_3 and finally P_4. This is the best strategy since our filtering algorithm is based on the first assigned variables which represent the prefix. For value selection strategy, the smallest value in the domain (w.r.t. its lexicographic order) is selected first. Initially, a preprocessing step is established in order to eliminate all infrequent items and quantities from variable domains. Hence, the first variable P_1 will be assigned to a, b and then c. Previously, we imposed that the two first variables must be assigned so that we avoid the empty sequence. Suppose that $\sigma(P_1) = a$, since the item a has two frequent quantities 1 and 2, so the domain of P_2 is restricted to these two values. Now, suppose that $\sigma(P_2) = 2$, the algorithm computes the projected database of $a(2)$: $QDB_1|_{\langle a(2)\rangle} = \{\langle b(6), d(2)\rangle, \langle c(2)\rangle\}$. According to this result, no frequent item is computed, so $\mathcal{FI} = \{\square\}$ which leads to instanciate P_3 and P_4 to the empty symbol $\square$ indicating the end of the sequence.

4.3 Handling the gap constraint

Before explaining how we integrate this constraint in the filtering algorithm, we give bellow its definition in the case of quantitative datasets.

Definition 4 (Gap constraint) *A quantitative sequential pattern with gap constraint $gap[M, N]$ is a pattern such that at least M and at most N elements are allowed between every two adjacent items, in the original q-sequences. Formally, a q-sequence $p = \langle p_1(q_1) \dots p_m(q_m)\rangle$ is a subsequence of $s =$*

Algorithm 2: ProjectQDB(QDB, $ProjSDB$, α)

```
Data: $QDB$: initial database; $ProjQDB$: projected q-sequences; $\alpha$: prefix
begin
1   $QDB\mid_\alpha \leftarrow \emptyset$ ;
2   for each pair $(sid, start) \in ProjSDB$ do
3       $s \leftarrow QDB[sid]$ ;
4       $pos_\alpha \leftarrow 1$; $pos_s \leftarrow start$ ;
5       while $(pos_\alpha \leq \#\alpha \wedge pos_s \leq \#s)$ do
            $s[pos_s]$ contains the pair (item, quantity), $s[pos_s].item$ returns the item and $s[pos_s].quantity$ returns its
              quantity.;
6           if $(\alpha[pos_\alpha] = s[pos_s].item)$ and $(\alpha[pos_\alpha + 1] \leq s[pos_s].quantity)$ then
7               $pos_\alpha \leftarrow pos_\alpha + 2$ ;
8           $pos_s \leftarrow pos_s + 1$ ;
9       if $(pos_\alpha = \#\alpha + 1)$ then
10          $QDB\mid_\alpha \leftarrow QDB\mid_\alpha \cup \{(sid, pos_s)\}$
11  return $QDB\mid_\alpha$ ;
```

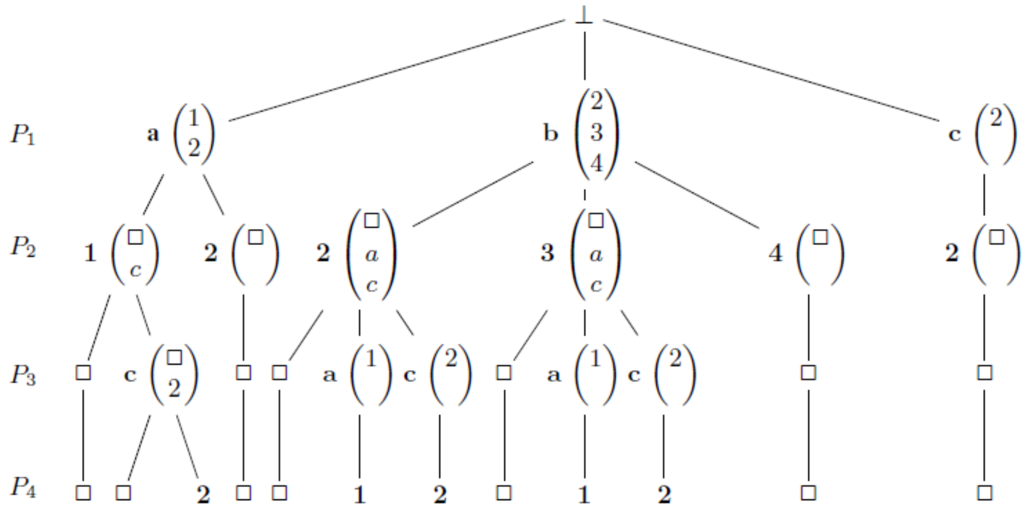

Figure 1: The serach tree associated to the running example.

$\langle s_1(q'_1)\dots s_n(q'_n)\rangle$, *under the gap constraint* $gap[M,N]$, *denoted by* $(p\preceq^{[M,N]}s)$, *if* $m \leq n$ *and, for all* $1 \leq i \leq m$, *there exist integers* $1 \leq j_1 \leq \dots \leq j_m \leq n$, *such that* $p_i = s_{j_i}$ *and* $q_i <= q'_{j_i}$, *and* $\forall k \in \{1,...,m-1\}, M \leq j_{k+1} - j_k - 1 \leq N$.

Example 4 *Let us take the q-sequence* $s_1 = \langle b(4)a(1)c(3)\rangle$. *The q-sequence* $p_1 = \langle b(4)a(1)\rangle$ *is a subsequence of* s_1 *under* $gap[0,0]$ *and* $p_2 = \langle b(3)c(1)\rangle$ *is a subsequence of* s_1 *under* $gap[1,1]$.

In order to consider the gap constraint, the necessary modifications affect the way to compute the the frequent items ($\mathcal{FI}$), the internal structure $FreqQItems$ and the frequent quantities ($\mathcal{FQ}$) (lines 10-11) in the filtering algorithm 1. Instead of scanning all the items of each sequence of the projected database, we scan only the items located in the interval [M,N] (i.e items located after M positions, going to position N of the current sequence). For example, if we consider the dataset QDB_1 with $minusp = 2$ and $gap[1,1]$, the locally frequent items allowing to extend the pattern $\langle b(2)\rangle$, are computed from the following projected database under gap constraint: $\{(1,\langle c(3)\rangle),(2,\langle b(6)\rangle),(3,\langle c(2)\rangle)\}$. We can observe that only $c(2)$ is frequent and thus the pattern $\langle b(2)c(2)\rangle$ is frequent under $gap[1,1]$.

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
BIBLE	36369	13905	21.64	100	bible
MSNBC	31790	17	13.33	100	web click stream
Kosarak	69999	21144	7.97	796	web click stream
BMS	59601	497	2.42	162	e-commerce

Table 4: Dataset Characteristics.

5 Experiments

Our approach was carried out using the `gecode` solver[1]. All experiments were conducted on a processor Intel core i5-3210M with 8 GB of memory. A time limit of 1 hour 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 [3] considered in these experiments have different characteristics and represent different type of data (see Table4). Since these datasets contain items without quantities, we add a random quantity to each item between 1 and 10.

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

1. We compare `QPP` (when all quantities are equal to 1) with `PP`, the global constraint proposed for sequential pattern mining without quantities, `PrefixSpan`[2] and `cSpade`[3], the state-of-the-art specialized methods for SPM,

2. We evaluate our approach `QPP` when the interval of quantities increases,

3. We give results of `QPP` when the gap constraint is enables, and we compare it with (1) GAP-SEQ [7] which is a CP-based approach to handle the gap constraint, and (2) `cSpade` which is the best ad-hoc method allowing to handle this constraint,

4. We show the ability of our approach to handle different kinds of constraints.

5.1 `QPP` vs. `PP`, `PrefixSpan` and `cSpade`

In this first experiment, we compare our approach `QPP` with `PP` and the state- of-the-art methods for SPM `cSpade` and `PrefixSpan`. Since these last three methods don't consider quantities, we fixed the quantity interval to only one value equal to 1 ($Q = \{1\}$). Results are shown in Table 5 in terms of CPU times. First, `cSpade` obtains the best times on all datasets. The second best method is `PP` which behaves better than `PrefixSpan`. More results comparing these there methods are given in [5]. Now, if we consider our new approach `QPP`, it is 4 times less faster than `PP`: this result explains the cost of adding quantities for items since `QPP` is an extension of `PP`. Despit the slowness of our method, it remains competitive with `PrefixSpan` and it behaves better in some cases.

5.2 The behaviour of `QPP` varying the quantity interval

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 the datasets FIFA, Kosarak and Leviathan. Results are reported in Figure 2. First, we observe that the CPU time and the number of patterns increase in a monotonic manner. Second, when the quantity increases, the number of extracted patterns increases as well, except for the FIFA dataset, when $Q = [1, 4]$, the CPU time decreases while the number of patterns increases, but, this difference is not significant.

[1]`http://www.gecode.org`
[2]`http://illimine.cs.uiuc.edu/software/prefixspan-mining-sequential-patterns-efficiently-prefix-projected-pattern-growth/`
[3]`http://www.cs.rpi.edu/~zaki/www-new/pmwiki.php/Software/`

Dataset	minsup (%)	# PATTERNS	CPU times (s)			
			`QPP`	`PP`	`cSpade`	`PrefixSpan`
	10	40642	417.89	132.80	121.765	314.74
	20	938	34.98	6.00	6.41	18.46
FIFA	30	47	2.22	1.50	0.84	5.69
	40	5	1.12	0.38	0.04	4.39
	50	0	1.05	0.32	0.003	4.38
	10	651	3.00	0.98	0.6	5.75
	20	101	0.99	0.32	0.14	3.76
LEVIATHAN	30	32	0.57	0.19	0.16	3.21
	40	12	0.28	0.14	0.13	3.08
	50	8	0.25	0.11	0.12	2.97
	10	338	5.91	1.79	06	–
	20	73	2.88	0.78	0.37	244
MSNBC	30	27	1.72	0.48	0.3	54
	40	13	1.06	0.27	0.1	9.5
	50	5	0.70	0.18	0.04	2.77
	10	110417	57.37	21.14	4.96	22.49
	20	9718	12.53	3.14	1.03	2.8
SIGN	30	1928	4.26	1.07	0.48	0.83
	40	518	1.39	0.38	0.23	0.34
	50	173	0.34	0.09	0.18	0.19

Table 5: `QPP` vs. `PP`, `cSpade` and `PrefixSpan` for $Q = \{1\}$ in terms of CPU time.

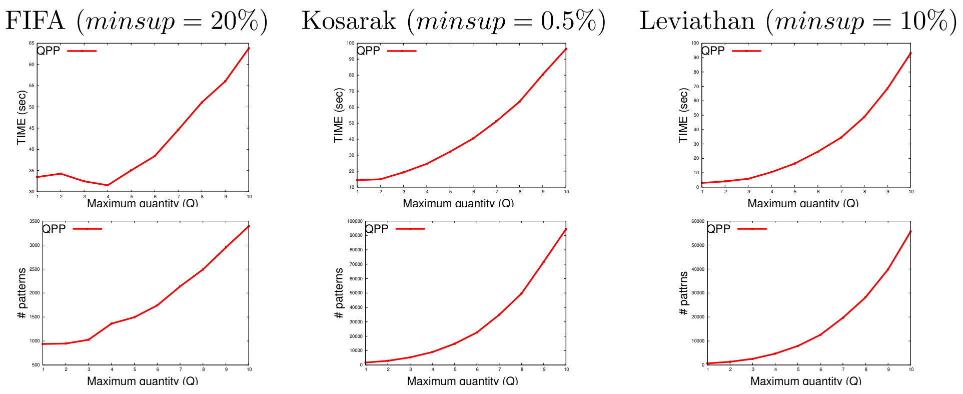

Figure 2: Results of `QPP` varying the quantity interval: CPU times and number of patterns.

5.3 Handling the gap constraint

Since our approach allows to handle the gap constraint, which is considered in the implementation of the filtering algorithm of `QPP`, we compare our approach with GAP-SEQ and `cSpade`. Results are shows in Figure 3. Again, we have fixed the quantity interval to $[1, 1]$ since GAP-SEQ and `cSpade` don't consider quantities. First, GAP-SEQ is the faster method, compared to our proposed method `QPP`, this is explained by the cost of adding quantities. Second, in some cases, `QPP` behaves better than `cSpade` but sometimes it reaches the timeout without extracting all sequential patterns which is the case of FIFA and BIBLE datasets.

5.4 Extracting interesting quantitative patterns under different constraints

In order to show the interest of our approach based on CP, we have considered the dataset BMS containing clickstream data 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 more than N products in the same time? (2) Is there customers who buy only one item for each product ? (3) Is there customers who buy 3 products with specified quantities? Effectively, we can get answers to these questions easily using

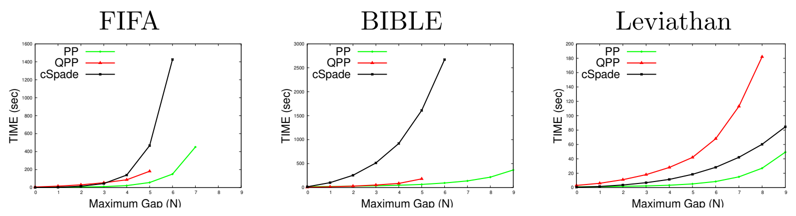

Figure 3: Comparing `QPP` with `PP` and `cSpade` varying the maximum gap N ($M = 0$): CPU times.

constraints	minsup %	# patterns	CPU times (s)
Const.1	3	0	0.51
	2	15	0.66
	1	236	1.25
	0.5	3622	3.50
Const.2	5	1	0.46
	1	163	0.96
	0.5	817	2.71
	0.1	46587	46.87
Const.3	1	6	0.74
	0.5	75	1.33
	0.1	2102	7.30
	0.05	5809	13.34

Table 6: Results of `QPP` under constraints on BMS dataset.

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 (for the first, the second and the third question respectively):

1. $const.1 \equiv \sum_{i=2}^{i=2\times\ell}(i\%2 = 0)(P_i >= N)$
2. $const.2 \equiv \bigwedge_{i=2}^{i=2\times\ell}(i\%2 = 0)(P_i = 1)$
3. For example, we chose to fix the values of quantities to 1,2 and 3 for the three products respectively, in this case, we impose a maximum size and a membership constraints as follows:

$$const.3 \equiv (\bigwedge_{i=7}^{i=2\times\ell} P_i = \Box) \wedge (P_2 = 1, P_4 = 2, P_6 = 3)$$

Results are given in Table 6. It is clear that having an answer for the above questions is achieved in few seconds only. According to the extracted patterns, the expert can analyze the behavior of customers. He can also add other constraints (like regular expression constraints) for a more advanced study.

6 Conclusion

In this paper, we address the problem of mining quantitative sequential patterns. In the sequence database, each item is associated with its quantity. Since existing approaches don't allow to handle this kind of sequences under constraints, we have proposed the global constraint QSPM which is an extension of the two CP-based approaches proposed in [5] and [7]. Its filtering algorithm keep only consistent values allowing to verify simultaneously the subsequence relation, minimum threshold and gap constraints. In the experimental study, we have shown (1) the behaviour of QSPM varying the quantity interval, (2) that QSPM, under gap constraint, performs well comparing with existing methods, (2) how our approach allows to specify many constraints like size, membership and regular expression constraints which can be also combined together. As future work, we intend to propose a global constraint for mining fuzzy sequential patterns with constraints.

References

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In Philip S. Yu and Arbee L. P. Chen, editors, *ICDE*, pages 3–14. IEEE Computer Society, 1995.

[2] John O. R. Aoga, Tias Guns, and Pierre Schaus. 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*, pages 315–330, 2016.

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

[4] Minos N. Garofalakis, R. Rastogi, and K. Shim. Mining sequential patterns with regular expression constraints. *IEEE Trans. Knowl. Data Eng.*, 14(3):530–552, 2002.

[5] A. Kemmar, S. Loudni, Y. Lebbah, P. Boizumault, and T. Charnois. PREFIX-PROJECTION global constraint for sequential pattern mining. In *Principles and Practice of Constraint Programming - 21st International Conference, CP 2015, Cork, Ireland, August 31 - September 4, 2015, Proceedings*, pages 226–243, 2015.

[6] A. Kemmar, W. Ugarte, S. Loudni, T. Charnois, Yahia Lebbah, P. Boizumault, and B. Crémilleux. Mining relevant sequence patterns with cp-based framework. In *ICTAI*, pages 552–559, 2014.

[7] Amina Kemmar, Samir Loudni, Yahia Lebbah, Patrice Boizumault, and Thierry Charnois. A global constraint for mining sequential patterns with GAP constraint. In *Integration of AI and OR Techniques in Constraint Programming - 13th International Conference, CPAIOR 2016, Banff, AB, Canada, May 29 - June 1, 2016, Proceedings*, pages 198–215, 2016.

[8] Chulyun Kim, Jong-Hwa Lim, Raymond T. Ng, and Kyuseok Shim. SQUIRE: sequential pattern mining with quantities. In Z. Meral Özsoyoglu and Stanley B. Zdonik, editors, *Proceedings of the 20th International Conference on Data Engineering, ICDE 2004, 30 March - 2 April 2004, Boston, MA, USA*, page 827. IEEE Computer Society, 2004.

[9] Chun Li, Qingyan Yang, Jianyong Wang, and Ming Li. Efficient mining of gap-constrained subsequences and its various applications. *Trans. Knowl. Discov. Data*, 6(1):2:1–2:39, March 2012.

[10] J.-P. Métivier, S. Loudni, and T. Charnois. 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.

[11] B. Négrevergne and T. Guns. Constraint-based sequence mining using constraint programming. In *CPAIOR'15*, pages 288–305, 2015.

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

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

[14] Jian Pei, Jiawei Han, and Wei Wang. Mining sequential patterns with constraints in large databases. In *CIKM'02*, pages 18–25. ACM, 2002.

[15] Francesca Rossi, Peter van Beek, and Toby Walsh, editors. *Handbook of Constraint Programming*, volume 2 of *Foundations of Artificial Intelligence*. Elsevier, 2006.

[16] R. Trasarti, F. Bonchi, and B. Goethals. Sequence mining automata: A new technique for mining frequent sequences under regular expressions. In *ICDM'08*, pages 1061–1066, 2008.

[17] X. Yan, J. Han, and R. Afshar. CloSpan: Mining closed sequential patterns in large databases. In Daniel Barbará and Chandrika Kamath, editors, *SDM*. SIAM, 2003.

[18] M. J. Zaki. Sequence mining in categorical domains: Incorporating constraints. In *CIKM'00*, pages 422–429, 2000.

[19] Zhenglu Yang and M. Kitsuregawa. Lapin-spam: An improved algorithm for mining sequential pattern. In *21st International Conference on Data Engineering Workshops (ICDEW'05)*, pages 1222–1222, April 2005.