Mining Relevant Sequence Patterns with CP-based Framework

Amina Kemmar*‡, Willy Ugarte*, Samir Loudni*, Thierry Charnois†, Yahia Lebbah‡*, Patrice Boizumault*, Bruno Crémilleux*

*GREYC (CNRS UMR 6072) – University of Caen, Campus II Côte de Nacre, 14000 Caen - France

†LIPN (CNRS UMR 7030) – University PARIS 13, 99, avenue Jean-Baptiste Clément 93430 Villetaneuse - France

‡LITIO, University of Oran, 1524 El-M'Naouer & EPSECG d'Oran, BP 65 Ch 2 USTO, Oran, Algeria

***Abstract*—Sequential pattern mining under various constraints is a challenging data mining task. The paper provides a generic framework based on constraint programming to discover sequence patterns defined by constraints on local patterns (e.g., gap, regular expressions) or constraints on patterns involving combination of local patterns such as relevant subgroups and top-k patterns. This framework enables the user to mine in a declarative way both kinds of patterns. The solving step is done by exploiting the machinery of Constraint Programming. For complex patterns involving combination of local patterns, we improve the mining step by using dynamic CSP. Finally, we present two case studies in biomedical information extraction and stylistic analysis in linguistics.**

***Keywords*—Sequential mining, Constraint programming, subgroup patterns.**

I. Introduction

Sequential pattern mining is a well-known data mining technique introduced in [1] that aims at discovering correlations between events through their order of appearance in a database of sequences. For effectiveness and efficiency considerations, many authors [2], [3] have promoted the use of constraint programming to focus on the most promising knowledge by reducing the number of extracted patterns to those of a potential interest given by the final user. In this paper, we propose a generic approach for modelling and mining sequential patterns under various constraints using Constraint Programming (CP).

Sequential pattern mining is one of the most studied and challenging tasks in data mining with a wide range of applications and domains. Accurate processing of texts is crucial for the numerous applications on textual data. In linguistics, textual analysis enables to find interesting characteristic patterns from corpora of a literary author. In biomedicine, more than 20 million publications are currently listed in the PubMed repository. The automatic extraction of gene for rare diseases (RD) relationship, where the data is available in a textual form, is a very motivating application, since about 7,000 RD affect about 30 million people in Europe. A critical challenge is then to discover relevant and useful knowledge spread in such sequential data.

There are many algorithms to extract sequential patterns under the frequency constraint such as `PrefixSpan` [4], `CloSpan` [5], `BIDE` [6], etc. A survey of various constraints such as regular expression, length, aggregates can be found in [2]. But there is no unifying method to mine sequence patterns under various constraints adding and handling simultaneously several types of constraints in a nice and elegant way beyond the few classes of constraints studied is far from trivial. The main problem with these methods is that the user has to adopt a new method each time he wants to extract patterns satisfying a new type of constraints.

The paper addresses this issue by taking benefit of the Constraint Programming (CP) which brought successful results in the data mining area [7], [8]. The key power of CP lies in its generic and declarative approach to problem solving. The user models a problem by specifying constraints and the CP solver provides the complete set of solutions satisfying all the constraints. The great advantage of this modelling is its flexibility. It enables to define and to push new constraints without having to develop new algorithms from scratch.

We propose a unified framework for modelling and mining sequence patterns in a sequence database under a large set of constraints. We address both constraints dealing with local patterns (e.g. frequency, size, gap, regular expressions) and constraints defining more complex patterns such as relevant subgroups and top-k patterns. Moreover, our approach enables to combine simultaneously different types of constraints. To the best of our knowledge, it is the first CP-based model for discovering sequence patterns under various constraints. We show the interest of our approach towards two case studies in biomedical information extraction and stylistic analysis in linguistics.

This paper is organized as follows. Section II gives the preliminaries. In Section III, we state the sequence mining problem and detail the constraints. Section IV introduces the main principles of constraint programming. Section V describes our CP model. We review related work in Section VI. Section VII reports the two case studies. Finally, we conclude and draw some perspectives.

II. Preliminaries

In this section, we fix the notations and give the necessary definitions according to [9], [10].

Let $\mathcal{I}$ be a set of distinct literals called *items*. The language of sequences corresponds to $\mathcal{L}_{\mathcal{I}} = \mathcal{I}^n$ where $n \in \mathbb{N}^+$. A *sequence* of items S is an ordered list of items: $S = s_1 s_2..s_n \in \mathcal{L}_{\mathcal{I}}$ where $n = |S|$ is the length of the sequence. A *sequence database* SDB is a set of sequences, $SDB = \{S_1, S_2, ..., S_m\}$, $S_i \in \mathcal{L}_{\mathcal{I}}$ for all $i \in 1..m$. A

TABLE I: SDB_1: a sequence database example.

Classes	ID	Sequences
D^+	s_1	$DBED$
	s_2	$DBAC$
	s_3	CED
D^-	s_4	$ABAE$
	s_5	$DBCE$

bipartition of SDB over $\mathcal{I}$ is the pair (D^+,D^-) where D^+ and D^- are disjoint sets of sequences whose union is SDB. D^+ (resp. D^-) denotes the positive (resp. negative) class.

Definition 1 (Sequence pattern): Given the set of items $\mathcal{I}$, whose elements are also called solid characters. We introduce a new symbol called *wildcard* (denoted by $\diamond$) which can replace any item in $\mathcal{I}$. Let be $\mathcal{I}_\mathcal{P} = \mathcal{I} \cup \{\diamond\}$. A sequence pattern P is a sequence over $\mathcal{L}_\mathcal{P} = \mathcal{I} \times (\mathcal{I}_\mathcal{P})^{\ell-1}$ where $\ell \in \mathbb{N}^+$. We impose that the sequence pattern starts with a solid character and ends with any character including the *wildcard*.

Definition 2 ($\preceq$, Occurrence): For any two characters $\sigma_1, \sigma_2 \in \mathcal{I}_\mathcal{P}$, we have $\sigma_1 \preceq \sigma_2$ if $\sigma_1 = \diamond$ or $\sigma_1 = \sigma_2$. A sequence pattern $P = p_1..p_\ell \in \mathcal{L}_\mathcal{P}$ is a sub-sequence of $S = s_1..s_n \in SDB$, denoted by $P \preceq S$, iff there exists an integer $i \geq 1$, such that $occur(P, S, i)$ holds (equivalently, we say that P occurs at position i in S), where $occur(P, S, i) \equiv (i + \ell \leq n)$ and $(p_j \preceq s_{j+i-1}, \forall j \in 1..\ell)$.

Definition 3 (Cover, Support): The cover of a sequence pattern P in SDB is defined as the set $cover_{SDB}(P) = \{S | S \in SDB, P \preceq S\}$. The support of P, denoted by [1] $sup_{SDB}(P)$, is defined as the number of sequences in SDB which contain P, $sup_{SDB}(P) = |cover_{SDB}(P)|$. A sequence pattern P is said *frequent* in a database SDB if, given a *minimum support threshold* $minsup$, $sup_{SDB}(P) \geq minisup$.

Table I represents a sequence database of five sequences where the set of items is $\mathcal{I} = \{A, B, C, D, E\}$. Let us consider the pattern $P = B\diamond E$. We have $cover_{SDB_1}(P) = \{s_4, s_5\}$, since $occur(P, s_4, 2) = true$ and $occur(P, s_5, 2) = true$. If we consider $minsup = 2$, the pattern $P = B\diamond E$ is a frequent sequence pattern because $sup(P) \geq 2$.

III. Mining Sequence Patterns with Constraints

In this section, we define the problem of mining sequence patterns in a sequence database of items satisfying user-defined constraints. Then we present the modelling of several sequence pattern constraints. These constraints address *local* patterns (e.g., frequency, gap, regular expressions) or patterns involving several local patterns such as relevant subgroups and top-k patterns. Our framework does not require that constraints satisfy an (anti-)monotonic property.

Problem statement. Given a constraint $C(P, SDB)$ on sequence patterns P and a sequence database SDB, the problem of constraint-based sequence pattern mining is to find the *complete* set of sequence patterns satisfying $C(P, SDB)$.

The following sections present different types of constraints that we explicit in the context of the sequence patterns. All these constraints are handled by our framework.

[1] When there is no ambiguity, $sup_{SDB}(P)$ will be simply noted $sup(P)$.

A. Constraints on local patterns

Local patterns are regularities that hold for a particular part of the data. Here, locality refers to the fact that checking whether a pattern satisfies or not a constraint can be performed independently of the other patterns holding in the data. We review some of the most important constraints on local patterns [2].

Frequency constraint. Let P be a sequence pattern and $minsup$ a *minimum support threshold*. We say that P is a *frequent sequence pattern* in SDB w.r.t. $minsup$ if $sup(P) \geq minsup$.

Size constraint. This constraint, noted $size_{min,\ell}(P, SDB)$, restricts: (i) the length of a sequence pattern P to be equal to ℓ and (ii) the number of solid characters in P to be greater than or equal to min: $size_{min,\ell}(P, SDB) \equiv (|P| = \ell \wedge \ |\{i \,|\, p_i \neq \diamond\}| \geq min)$. For instance, if we impose the constraint $sup(P) \geq 2 \wedge size_{2,4}(P, SDB_1)$, only one pattern is mined from Table I: $P_1 = DB\diamond\diamond$.

Item constraint. An item constraint specifies a subset of items that should or should not be present in the sequence patterns, $item_t(P, SDB) \equiv (\exists i \in 1..|P|, p_i = t)$. For instance, if we impose the constraint $item_A(P, SDB_1) \wedge item_B(P, SDB_1)$, we can consider only the two sequences from Table I: s_2 and s_4. Then, if we impose the constraint $size_{2,3}(P, SDB_1)$ and $sup(P) \geq 2$, one sequence pattern is mined: $P_2 = BA\diamond$.

Gap constraint. Another widespread constraint is the gap constraint. A pattern satisfying a gap constraint $gap_{(m,n)}(P, SDB)$ is a pattern such that at least m wildcards and at most n wildcards are allowed between every two solid neighbor items in the pattern. For instance, let the following three constraints $gap_{(1,2)}(P, SDB_1)$, $size_{2,3}(P, SDB_1)$, and $sup(P) \geq 2$, defined over sequences of Table I, pattern $P_3 = B\diamond E$ satisfies these three constraints.

Regular expression constraint. A regular expression constraint $reg_{exp}(P, SDB)$ is a constraint specified as a regular expression exp over the set of items of SDB. A pattern satisfies $reg_{exp}(P, SDB)$ iff the pattern is accepted by its equivalent deterministic finite automata [11]. For example, the sequence pattern DBC is extracted from SDB_1 (Table I) since it satisfies the regular expression constraint $reg_{exp}(P, SDB_1)$, where $exp = D\{CE|AE|BC\}$.

B. Constraints on sets of local patterns

In practice, the data analyst is often interested in discovering richer patterns than local patterns and he/she is looking for patterns that reveal more complex characteristics from the database. The definitions relevant to such patterns rely on properties involving several local patterns [12], [13]. In the following, we give the modelling of several such patterns.

Closedness constraint. A sequence pattern P is closed in a sequence database SDB if there exists no sequence pattern P' such that $P \preceq P'$ and $sup(P)$=$sup(P')$. The closedness constraint allows to get a condensed representation of the complete set of extracted sequence patterns.

$$closedness(P, SDB) \equiv \begin{cases} true \text{ if } \forall P' \in \mathcal{L}_\mathcal{P} \text{ such that } P \preceq P', \\ freq(P) > freq(P') \\ false \text{ otherwise} \end{cases}$$

Top-k constraint. top-k patterns are the k patterns optimizing an interestingness measure m. In this work, we consider the problem of mining top-k frequent sequence patterns having at least min solid characters. In this problem, the minimum support threshold $minsup$ is not known.

Definition 4: Let k and min be strictly positive integers. A sequence pattern P is a top-k frequent pattern of at least min solid characters if there exists no more than $(k-1)$ sequence patterns having at least min solid characters and whose support is higher than that of P.

From definition 4, we can formulate the top-k constraint as follows:

$top_{k,min}(P, SDB) \equiv |\{P' \in \mathcal{L}_\mathcal{P} | P' \neq P \wedge size_{min,\ell}(P') \wedge m(P') > m(P)\}| < k$

where ℓ is the length of the sequence pattern and m is the frequency measure or any other measure. For instance, if we look for Top-2 sequence patterns with $min = 2$ and $\ell = 3$, three sequence patterns are mined from Table I: $DB\diamond$ with support 3, $BA\diamond$ and $B\diamond E$ with support 2.

Relevant subgroup constraint. Let the pair (D^+, D^-) be a bipartition of SDB. Mining relevant patterns consists in finding patterns that discriminate the positive dataset (D^+) from the negative one (D^-) [14]. This constraint can be formulated as follows:

$subG_{minsup}(P, SDB) \equiv$

$$\begin{cases} true \text{ if} \\ sup_{D^+}(P) \geq minsup \ \wedge \nexists P' \in \mathcal{L}_\mathcal{P} : \\ cover_{D^+}(P) \subseteq cover_{D^+}(P') \wedge \\ cover_{D^-}(P) \supseteq cover_{D^-}(P') \wedge \\ (cover_{SDB}(P) = cover_{SDB}(P') \Rightarrow P' \preceq P) \\ false \text{ otherwise} \end{cases}$$

The last condition states that if two patterns cover exactly the same set of sequences in SDB, the one that includes the other is considered. Let us consider the database SDB_1 from Table I. We classify its sequences on two classes: D^+ and D^-. If we impose the constraint $size_{1,3}(P, SDB_1) \wedge subG_{minsup}(P)$ with $minsup = 2$, only one sequence pattern is considered as relevant, which is $P_4 = DB\diamond$.

IV. Constraint Programming

Constraint programming (CP) is a generic framework for solving combinatorial problems modelled as Constraint Satisfaction Problems (CSP).

Constraint Satisfaction Problems. A *Constraint Satisfaction Problem* (CSP) consists of a finite set of variables $X = \{X_1, \ldots, X_n\}$ with finite domains $\mathcal{D} = \{D_1, \ldots, D_n\}$ such that each D_i is the set of values that can be assigned to X_i, together with a finite set of constraints $\mathcal{C}$, each on a subset of X. A constraint $C \in \mathcal{C}$ is a subset of the cartesian product of the domains of the variables that are in C. The goal is to find an assignment $(X_i = d_i)$ with $d_i \in D_i$ for $i = 1, \ldots, n$, such that all constraints are satisfied.

Dynamic CSPs. A Dynamic CSP [15] is a sequence $\mathcal{P}_1, \mathcal{P}_2, ..., \mathcal{P}_n$ of CSPs, each one resulting from some changes in the definition of the previous one. These changes may affect every component in the problem definition: variables, domains and constraints. *For our approach, changes are only performed by adding new constraints.* Solving such dynamic CSP involves solving a single CSP with additional constraints posted during search. Each time a new solution is found, new constraints are imposed. Such constraints will survive backtracking and state that next solutions should verify both the current set of constraints and the added ones.

Global constraints. An important modelling technique from CP are the *global constraints* that provide shorthands to often-used combinatorial substructures. We present briefly two global constraints, `Among` and `Regular`, allowing to model constraints described in Section III-A.

Among Constraint. This constraint restricts the number of variables using values from a given set (see [16] for more details).

Regular Constraint. Given a deterministic finite automaton M describing a regular language, constraint $\texttt{Regular}(X, M)$ ensures that every sequence of values taken by the variables of X have to be a member of the regular language recognised by M [17].

V. Our CP Model for Sequence Patterns

We present our CP model for mining sequence patterns with constraints. We start by introducing the model of a sequence pattern that we defined within the CP paradigm (Section V-A). Thanks to the nice declarative side of the CP, our approach enables us to express in a straightforward way constraints on local patterns (Section V-B). We describe how more complex sequence patterns such as top-k and relevant subgroups can be modelled and extracted using Dynamic CSP [15] (Section V-C). To the best of our knowledge, our approach is the first generic one for mining sequence patterns in a sequence database under constraints on local patterns or on patterns involving combination of local patterns.

A. Sequence pattern encoding

In the remainder of this section, let SDB be a dataset where $\mathcal{I}$ is the set of its n items and $\mathcal{S}$ the set of its m sequences, and let d be the matrix representing the SDB where $S \in \mathcal{S}$ and for $i \in \{1, \ldots, |S|\}$, $d_{S,i} = s_i$.

Variables. Let P be the unknown sequence pattern of size ℓ we are looking for. First, ℓ variables $\{p_1, p_2, \ldots, p_\ell\}$ are introduced to represent P. The domains of variables p_i are defined as follows: $D_1 = \mathcal{I}$ and $\forall i \in [2..\ell], D_i = \mathcal{I}_\mathcal{P}$. This expresses that the first item of P must be different from the *wildcard* symbol, while the other items of P may be wildcards.

Second, to encode that "$P \preceq S$", the following boolean variables are introduced: $POS_{S,j}$ such that $(POS_{S,j} = 1) \Leftrightarrow (occur(P, S, j) = true)$. Variables $POS_{S,j}$ enable us to capture the positions where the candidate sequence pattern P appears. As the last possible position of a pattern P in S is $(|S| + 1 - \ell)$, then $j \in [1 \ldots |S| + 1 - \ell]$.

Finally, m variables $\{T_1, \ldots, T_m\}$, having the domain $\{0, 1\}$, are used such that (T_S = 1) iff the sequence S contains the sequence pattern P:

$$\forall S \in \mathcal{S}, (T_S = 1) \Leftrightarrow (P \preceq S) \quad (1)$$

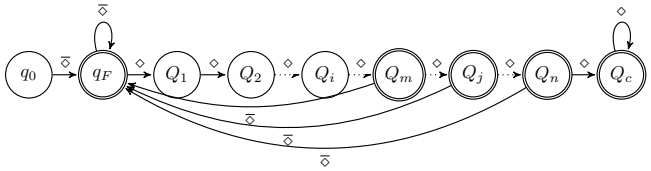

Fig. 1: Automaton $A_{m,n}$. $\diamond$ represents the wildcard symbol and $\overline{\diamond}$ represents any symbol of $\mathcal{I}$.

Occurrence constraint. From equation (1) we have: $(T_s = 1) \Leftrightarrow \exists j \in [1 \dots |S| + 1 - \ell]$ such that $occur(P, S, j) = true$. This property is reformulated by the following constraint:

$$\forall S \in \mathcal{S}, (T_S = 1) \Leftrightarrow (\Sigma_{j \in [1 \dots |S|+1-\ell]} POS_{S,j} \geq 1) \quad (2)$$

The relationship between each candidate pattern P and a sequence S of the SDB where P appears is expressed by the following constraint, stating that, for all $i \in [1..\ell]$, item of rank $(j + i - 1)$ in S corresponds to the $i - th$ item of P, and the first item of P must be different from $\diamond$.

$$(POS_{S,j} = 1) \Leftrightarrow (P_1 = d_{S,j}) \wedge (\bigwedge_{i \in [2..\ell]} (p_i = \diamond \vee p_i = d_{S,j+i-1})) \quad (3)$$

B. Reformulating constraints on local patterns

This section shows how our model allows us to express in a straightforward way constraints presented in Section III-A using the constraints provided by CP solvers.

Frequency constraint: $sup(P) \geq minsup \equiv \sum_{S \in \mathcal{S}} T_S \geq minsup$

Item constraint: $item_V(P, SDB) \equiv \bigwedge_{t \in V} \texttt{Among}(P, \{t\}, [l, u])$. V is a subset of items, l and u are two integers s.t. $0 \leq l \leq u \leq \ell$. It enforces that items of V should be present at least l times and at most u times in P. To forbid items of V to appear in P, it suffices to set l and u to 0.

Size constraint: $size_{min,\ell}(P, SDB) \equiv \texttt{Among}(P, \{\diamond\}, [1, \ell - min])$.

Regular expression constraint: $reg_{exp}(P, SDB) \equiv \texttt{Regular}(P, A_{reg})$. A_{reg} is the deterministic finite automaton encoding the regular expression exp over the set of items.

Gap constraint: $gap_{(m,n)}(P, SDB) \equiv \texttt{Regular}(P, A_{m,n})$.

$A_{m,n}$ is defined by a 5-tuple $\{Q, \Sigma, \delta, q_0, F\}$ where Q is a finite set of states, Σ=$\mathcal{I}_\mathcal{P}$ the emission alphabet, $\delta : Q \times \Sigma \rightarrow Q$ the transition function, q_0 is the initial state and $F \subseteq Q$ the set of final states. The idea is to record in a state Q_i the last i consecutive wildcards encountered in the sequence pattern P. As at most n wildcards are allowed, we create n such states. We also add a final state q_F. The first item of pattern must be different from wildcard, we define a transition from a state q_0 to state q_F on value $\overline{\diamond}$. To satisfy the minimum gap constraint, we must add the following transitions: $Q_1 = \delta(q_F, \diamond)$ and $Q_{i+1} = \delta(Q_i, \diamond)$ for all $i \in [1, m-1]$. In addition, the states Q_i, $m \leq i \leq n$ are linked to allow sequence patterns with a number of wildcards between m and n. So, all these states are accepting states. Consequently, a transition on a non-wildcard value is added from each final state Q_i, $m \leq i \leq n$ to state q_F. The loop transition defined on q_F is only considered when $m = 0$ so that patterns without gap will be accepted. Figure 1 illustrates the automaton that would be built for the gap constraint $gap_{(m,n)}$. Finally, to allow wildcards at the end of patterns, we add the state Q_c with the following transitions: $Q_c = \delta(Q_n, \diamond)$ and $Q_c = \delta(Q_c, \diamond)$.

C. Dynamic CSP for Mining richer Sequence Patterns

This section describes how sequence patterns under constraints involving combination of local patterns can be modelled and extracted using Dynamic CSP [15]. We start by giving the key ideas of our method and then we show how it performs in the case of the top-k patterns and relevant subgroups.

A) Principles of the method. Basically, the main idea is to exploit a preference relation (noted $\rhd$) between patterns to produce a continuous refinement on the extracted patterns thanks to constraints dynamically posted during the mining process. Each dynamic constraint will impose that none of the candidate patterns already extracted is better (w.r.t $\rhd$) than the next pattern (which is searched). This process stops when no better solution can be found. We define a preference relation $\rhd$ between patterns as follows:

Definition 5 (Preference relation): A preference $\rhd$ is a strict partial order relation on $\mathcal{L}_\mathcal{P}$. Let P and P' be two sequence patterns, $P \rhd P'$ means that P is strictly preferred than P'.

Let us consider the sequence $\mathcal{P}_1, ..., \mathcal{P}_n$ of CSP where each $\mathcal{P}_i = (\{P\}, \mathcal{L}_\mathcal{P}, \mathcal{C}_i(P))$ and:

$$\mathcal{C}_{i+1}(P) = \begin{cases} \mathcal{C}(P) & \text{if } i = 0 \\ \mathcal{C}_i(P) \wedge \phi(P_i, P) & \text{if } i > 0 \end{cases} \quad (4)$$

$\mathcal{C}(P)$ is the initial constraint system. It expresses some property that must be satisfied by the unknown pattern P, like minimum frequency. Each time the first solution P_i is found to $\mathcal{C}_i(P)$, we dynamically post a new constraint $\phi(P_i, P)$. Constraints $\phi(P_i, P)$ for $i \in [1..n]$ successively impose that all patterns P_i found may not be better (according to $\rhd$) than the searched pattern P. Thus, at step $(i+1)$ we add the constraint $\phi(P_i, P)$ defined by : $\phi(P_i, P) = \neg(P_i \rhd P)$.

Consequently, none of the patterns P_1, P_2, ..., P_i obtained is better than P (immediate proof by induction). The new constraints $\phi(P_i, P)$ added dynamically enable to reduce the search space. This process stops when no better pattern can be found (i.e. there exists n such that $\mathcal{C}_n(P)$ has no solution). Finally, the completeness of our approach is ensured by the completeness of the CP solver. But, the n extracted patterns P_1, P_2, ..., P_n are not necessarily all final solutions as the considered preference relation $\rhd$ is a partial order. Some of them can only be "intermediate" patterns simply used to improve the pruning of the search space. A post processing step must be performed to filter all candidate patterns P_i for which there exists P_j $(1 \leq i < j \leq n)$ s.t. $P_j \rhd P_i$. So, the mining process is achieved in a two-steps: (1) Compute the set $\{P_1, P_2, \dots, P_n\}$ of candidates using Dynamic CSP; (2) Remove all "intermediate" patterns P_i.

The following sections describe how the examples in Section III-B can be modelled and solved using Dynamic CSP. For two of them, we give the initial constraint system and the constraints $\phi(P_i, P)$ added dynamically.

B) Mining the top-k sequence patterns. Initially, we have to set the minimum support $minsup$ to 1 to ensure that all the top-k patterns will be found. Moreover, we impose that the searched pattern P must have at least min solid characters. Thus, $\mathcal{C}(P) \equiv sup(P) \geq 1 \wedge size_{min,\ell}(P, SDB)$. During the search, a list of candidate patterns $\mathcal{C}and$ is maintained and ordered according to their frequency. At the beginning, all the mined patterns are added until $\mathcal{C}and$ has k candidate patterns. Thus, $\phi(P_i, P)$=$true$. Once k patterns are found[2], we impose that the new searched pattern P must have a frequency greater than the minimum frequency of the k patterns. This is done by adding the dynamic constraint $\phi(P_i, P)$ and by removing from $\mathcal{C}and$ the pattern with the smallest value of frequency i.e $\mathcal{C}and \leftarrow \mathcal{C}and \setminus \{\arg\min_{P_j \in \mathcal{C}_{and}} sup(P_j)\}$. Thus, $\phi(P_i, P)$ is defined as follows:

$$\phi(P_i, P) = \begin{cases} sup(P) > \min\limits_{P_j \in \mathcal{C}_{and}} sup(P_j) & if \quad i \geq k \\ true & otherwise \end{cases}$$

This process is repeated until no pattern is generated. The great interest of this iterative process is that the added constraints will refine the pruning condition leading to more and more powerful pruning of the search space. Let us note that for the top-k patterns, no post-processing step is required as the extracted patterns are ordered according to a total order relation on the frequency measure.

C) Mining the relevant subgroups. From the relevant subgroup constraint defined in Section III-B, the initial constraint system and the constraints $\phi(P_i, P)$ to be added dynamically are defined as follows: $\mathcal{C}(P) \equiv sup(P) \geq minsup$ and

$$\phi(P_i, P) \equiv \begin{cases} cover_{D^+}(P) \nsubseteq cover_{D^+}(P_i) \vee \\ cover_{D^-}(P) \not\supseteq cover_{D^-}(P_i) \vee \\ (cover_{SDB}(P_i) = cover_{SDB}(P) \wedge P_i \not\preceq P) \end{cases}$$

Finally, contrary to the top-k sequence patterns, the relevant subgroup sequence patterns are ordered by a partial order relation. Thus, a post-processing step is required to eliminate the irrelevant ones.

VI. Related Work

Computing Sequential Patterns. In the context of constraint-based sequential pattern mining, several algorithms have been proposed [2], [18], [19], [20] All these algorithms exploit properties of the constraints to perform effective pruning. For constraints that do not fit in these categories, they are handled by relaxing them to achieve some nice property (like anti-monotonicity) facilitating the pruning. Such a method, though interesting, makes tricky the integration of such constraints in a nice and elegant way. So, unlike these algorithms, our approach enables to address in a unified framework a broader set of constraints defined on local patterns or on patterns involving the combination of local patterns.

[2]Since there could be more than one sequence pattern having the same support in a sequence database, the number of the top-k patterns may be greater than k. To generate all top-k patterns, we need just to replace the strict inequality in $\phi(P_i, P)$ by an inequality ($\geq$). For our experiments, we have considered the strict inequality.

CP for Pattern Mining. In the context of local patterns, an approach using CP for itemset mining has been proposed in [8]. This approach addresses in a unified framework a large set of local patterns. To deal with richer patterns satisfying properties involving several local patterns, different extensions have been proposed, such as pattern sets [21], n-ary patterns [7], dominance algebra [22] or skypatterns [23]. Our approach also benefits from the recent progress on cross-fertilization between data mining and CP for itemset mining, but it addresses a different problem with a different modelling.

CP for Sequence Mining. More recently, [9], [24] have proposed a SAT-Based approach for discovering frequent, closed and maximal sequential patterns with wildcards in only a single sequence of items or itemsets. However, unlike [9], [24], our approach considers a sequence database of items. Moreover, our approach allows to consider a broader set of constraints that are not handled in [9] (e.g. gap, regular expression, top-k and relevant subgroup constraints). Finally, [25] have proposed a CP-based approach for mining sequential patterns in a sequence database. Each sequence is encoded by an automaton capturing all subsequences that can be found inside this sequence. However, our sequence pattern encoding is very different and much more efficient. Moreover, we are able to address constraints defined on top-k patterns and on relevant subgroups patterns.

VII. Experimentations

This section evaluates the performance of our approach. For this purpose, we selected two case studies. The first one focuses on the extraction of sequence patterns under various constraints (frequency, gap, item, top-k) from biomedical texts. The second one aims at discovering subgroup patterns for the stylistics analysis of literary texts. All experiments are done on a 8-cores of 2.00GHZ Intel Xeon, running the Linux operating system at UCI-Cerist and UCI-UOran. The implementation was carried out in Gecode[3]. A timeout of 24 hours has been used. When the extraction process cannot be completed within the time limit, it will be indicated by the symbol $(-)$ in the table.

A. Mining Sequence Patterns from Biomedical Texts

Problem. The goal of this application is to discover relations between genes and diseases from biomedical texts. The details of this application is given in [26]. In this section, we focus on the extraction of sequence patterns, using our CP approach.

Settings. We created a corpus from the PubMed database using HUGO[4] and Orphanet dictionaries to query the database to get sentences having these two kinds of entities. $17,527$ sentences of size up to $2,000$ words have been extracted in this way and we labelled the gene and rare disease (RD) names thanks to the two dictionaries. For instance, the sentence "*<**disease**>**Muir-Torre syndrome**<**disease**> is usually inherited in an autosomal dominant fashion and associated with mutations in the mismatch repair genes, predominantly in <**gene**>**MLH1**<**gene**> and <**gene**>**MSH2**<**gene**>*

[3]www.gecode.org

[4]www.genenames.org

genes." contains one recognized RD, and two recognized genes. These $17,527$ sentences are the training corpus from which we experiment the sequence pattern extraction.

Sequences of the SDB are the sentences of the training corpus: an item corresponds to a word of the sentence. We carry out a POS tagging of the sentences thanks to the TreeTagger tool [27]. In the sentences, each word is replaced by its lemma, except for gene names (respectively disease names) which are replaced by the generic item $GENE$ (respectively $DISEASE$). Note that unlike machine learning based approaches relations (e.g. gene-disease relations) are not annotated, but are discovered.

In order to discover sequence patterns, we impose usual constraints such as *the minimal frequency* and *the minimal size* constraints and other useful constraints expressing some *linguistic knowledge* such as the *item constraint*. The goal is to retain sequence patterns which convey linguistic regularities (e.g., gene – rare disease relationships). Our method offers a natural way to simultaneously combine in a same framework these constraints coming from various origins. We briefly sketch these constraints.

- *The minimal frequency constraint.* Three values of minimal frequency have been experimented: $0,5\%$, 1% and 2%.
- *The size constraint.* The aim of this constraint is to remove sequence patterns that are too small with respect to the number of items (number of words) to be relevant linguistic patterns and to limit the length of extracted patterns. We tested this constraint with min set to 3 and ℓ set to 20.
- *The item constraint.* This constraint enables to filter out sequence patterns that do not contain some selected items. For example, we express that the extracted patterns must contain at least three items (expressing the linguistic relation): $GENE$, $DISEASE$ and noun or verb[5].

Experimental protocol. We performed experiments on several subsets of the PubMed dataset with different sizes ranging from 500 to $4,000$ sentences. Such a choice of values is explained by the fact that under 500, we seem to get results within reasonable runtime. In fact, studying the behavior of our CP model on datasets having more than 500 sequences becomes interesting as the size of instances are reasonably large: up to $120,020$ variables (with $120,000$ boolean variables), domain size ranging from 2 to $4,197$, and up to $6,848,002$ constraints. To study the influence of the gap constraint on the extraction process, we considered two cases: with $gap_{(0,9)}$ and without gap constraint. In the last case, the number of wildcards is limited by the length of patterns. We conducted two kinds of experiments. In the first set of experiments, we aim at evaluating the feasibility of our approach for mining sequence patterns under constraints on local patterns described previously, while in the second one we analyze the behavior of our Dynamic CSP for mining the top-k sequence patterns.

Quantitative results. Figure 2 reports the number of extracted sequence patterns and the CPU-times to extract them (in seconds) with and without gap for each dataset we considered. First, as expected, the lower $minsup$ is, the larger the number of extracted sequence patterns. For example, when $minsup = 1\%$, we extract all frequent patterns before reaching the timeout. Second, CPU-times vary according to the size of the datasets. When the number of frequent patterns is reasonably small our CP approach succeeds to complete the extraction of all frequent patterns whatever the size of the dataset. However, when the number of solutions becomes huge (i.e. case of low frequency thresholds), the CP approach does not succeed to complete the extraction of all frequent sequence patterns within the timeout. Finally, the gap constraint decreases the CPU-times as well as the number of extracted patterns. Indeed, without gap constraint, more patterns are extracted (i.e., we obtain all patterns with a number of wildcards limited only by the length of patterns) which substantially increases the CPU-times.

[5]For each word (i.e. item), its grammatical category is stored in a base.

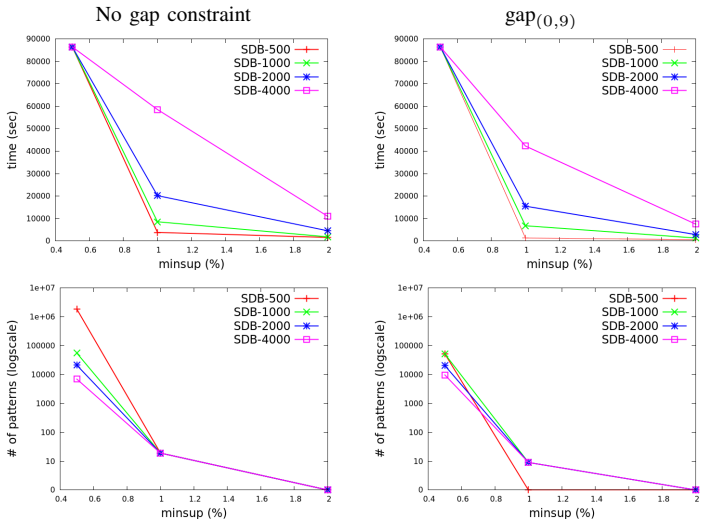

Fig. 2: Number of patterns (bottom) and CPU-times (top) according to the gap and $minsup$ constraints.

Qualitative results. Our approach allowed to extract relevant *linguistic patterns* which are useful to extract gene - RD relationships from biomedical texts (see [26]). In addition, and unlike statistical methods (*e.g.* Support Vector Machines) which are popular in text mining, the patterns are readable and manageable by an expert, see for instance these three patterns:

1) $\langle (DISEASE)\ (be)\ (cause)\ (by)\ (mutation)\ (in)\ (the)\ (GENE) \rangle$
2) $\langle (GENE)\ (occur)\ (in)\ (DISEASE) \rangle$
3) $\langle (DISEASE)\ (be)\ (an)\ (mutation)\ (in)\ (GENE) \rangle$

Mining top-k sequence patterns. In this second set of experiments, we considered the problem of mining top-k frequent sequence patterns having at least min solid items. We set the value of min to 3, ℓ to 20, gap to $(0,9)$ and we varied the value of k from 1 to $10,000$. The results are shown in Figure 3.

As expected, the CPU-time needed for computing the top-k patterns increases with k. Our CP approach fails to compute the top-k patterns for higher value of $k \geq 2,000$ in the time limit. This figure clearly shows that finding the top-k sequence patterns can be computed efficiently for small values of k (≤ 500) and for datasets with size up to $2,000$ sequences. Thus, the top-k constraint enables to control the size of the patterns

extracted and to get the most relevant ones.

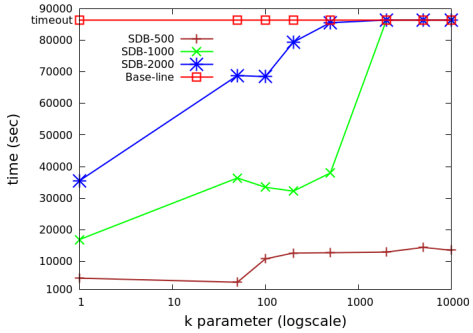

Fig. 3: Results of top-k patterns extraction.

Effectiveness of our Dynamic CSP approach. In order to evaluate the effectiveness of our Dynamic CSP approach, we compared with a `base-line` method to extract top-k patterns. It proceeds in two steps: first, all candidates satisfying the local constraints are generated. Then, a filtering step is applied enabling to remove all useless patterns (that are not top-k patterns w.r.t. the frequency measure). Figure 3 compares the performance of the two methods according to the value of k. Reported CPU-times include the two steps. We can see clearly that our approach clearly outperforms the `base-line` method on all the datasets. Thus, dynamic CSPs enable to prune significantly the search space, but as additional constraints are posted during search, this may increase the CPU time for solving the resulting dynamic CSP. Let us note however, that the time spent for solving the different dynamic CSPs is not constant and can vary greatly.

TABLE II: Comparing CPU times: `SPM-REG` Vs `SPM-REL`.

# seq.	50		100		150		200		250	
$Minsup$	SPM-REG	SPM-REL	SPM-REG	SPM-REL	SPM-REG	SPM-REL	SPM-REG	SPM-REL	SPM-REG	SPM-REL
10%	237	5	22	10	97	14	277	27	676	36

Comparing with the CP-based approach of [25]. To compare our method (`SPM-REL`) with the one proposed in [25] (`SPM-REG`), we have adapted the code of `SPM-REG`, provided by the authors, to generate similar patterns as those generated with our framework. For `SPM-REG`, we set the gap constraint to $Gap = (0, 0)$. Results are shown in Table II. CPU times are measured in seconds. `SPM-REG` succeeds only when $minsup = 10\%$ and only on small instances going from 50 to 250 sequences. When the number of instances is greater than 250 or minsup lower than 10%, `SPM-REG` cannot process these instances due to an excessive memory usage. For the few instances where `SPM-REG` finishes, `SPM-REL` behaves strictly better. This result can be explained by the fact that the main sub-sequence constraint in `SPM-REG` is encoded with an automaton who increases hugely with the size of sequences. Our approach uses basic linear constraints which is efficient for significant sequences size.

TABLE III: Results on datasets extracted from Zola and Roman according to the gap constraint.

	No gap constraint			$gap_{(0,0)}$		
Datasets	# of Candidates	CPU-Time (s.)	# of patterns	# of Candidates	CPU-Time (s.)	# of patterns
Z50-R50	18,957	**293**	681	91	**10**	58
Z100-R100	44,405	**1,584**	2271	198	**29**	133
Z200-R200	64,779	**3,761**	4268	291	**57**	196
Z500-R500	172,535	**36,798**	16811	526	**285**	382
Z1000-R1000	190,272	–	–	1,221	**1,263**	918
Z5000-R5000	24,203	–	–	6,068	**45,700**	4,118

B. Subgroups for Textual Analysis

Problem. The goal of this application is to provide to the linguist experts some prominent and relevant patterns which can be characteristic of a specific type of text so that these experts can carry out a stylistic analysis based on those patterns (see [28] for details).

Settings. For *corpus* setting, we use two corpora in order to discover some specific and emerging patterns of a type of texts compared to the second one. The first one is the one to be analyzed. It is compound of the fictions, "The Rougon-Macquart" a set of novels from Emile Zola (we call this corpus "ZOLA"). It contains 198913 sentences which are the sequences of the first data class.The second corpus is a set of fictions from several authors of the 1800-1900 era (we call this corpus "ROMANS"). ROMANS contains 189646 sentences using as sequences of the second data class.

We carry out a POS tagging of the sentences thanks to the TreeTagger tool [27]. From a linguistic point of view, because we want to focus on syntactical patterns, we replace each non-grammatical word (verb, noun and adjective) by its respective category (V for verb, NC for common name, NP for proper name and ADJ for adjective), and for each other category we keep the lemma of the word.

Experimental protocol. We extracted from the two classes ZOLA and ROMANS several subsets with different sizes ranging from 50 to 5,000 sentences. Zn (resp. Rn) will denote the subset from ZOLA (resp. ROMANS) with a size n. The length was fixed to 20 and the minimum one to 3 to eliminate uninteresting patterns. To study the influence of the gap constraint, we considered two cases: with and without gap. Table III reports, for each dataset: (i) the number of candidates, (ii) the required CPU-times to complete the extraction and (iii) the number of relevant subgroup patterns extracted. Reported CPU-times include the two steps[6]. When the gap constraint is disabled, our approach succeeds to extract all relevant patterns until $n = 500$. Table III also shows that the gap constraint has a great influence on the number of the extracted patterns. For instance, with $gap_{(0,0)}$, our approach performs very well and enables to extract all relevant patterns until $n = 5,000$.

We have also compared our Dynamic CSP approach with the `base-line` method. Results are given in Figure 4. Our

[6]The time of the post-processing step remains negligible compared to the first step.

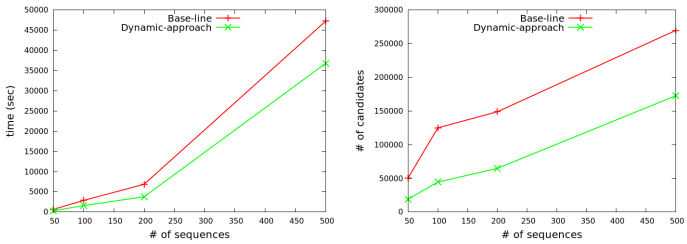

Fig. 4: Comparing the two approaches according to the number of sequences.

approach clearly dominates the `base-line` method in terms of CPU-times. Moreover, the number of candidates generated by our approach remains small compared to the number of candidate patterns computed by the `base-line` method. This confirms the interest of adding constraint dynamically during the solving process, thus allowing to reduce drastically the search space as well as the CPU-times.

If we consider the application from a linguistic point of view, several extracted patterns are relevant because they are characteristic of a type of text. For example, the pattern *de_le NC de le NC*[7] is frequent in Zola's fiction, not in other authors of the same period, thus interesting for stylistics study.

VIII. Conclusion

We have proposed a unified framework for modelling and mining sequence patterns in a sequence database under a large set of constraints. We addressed both constraints dealing with local patterns (e.g. frequency, size, gap, regular expressions) and constraints defining more complex patterns such as relevant subgroups and top-k patterns. Our approach enables to combine simultaneously different types of constraints. To the best of our knowledge, it is the first CP-based model for discovering sequence patterns under various constraints. Experiments performed on two case studies, biomedical information extraction and stylistic analysis in linguistics, showed the interest of our approach. We intend to extend our framework to handle other mining tasks such as skypatterns. We also intend to compare our approach with the state-of-the-art methodes.

Acknowledgments. This work is partly supported by the ANR (French Research National Agency) funded projects FiCoLoFo ANR-10-BLA-0214 and Hybride ANR-11-BS002-002.

References

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

[2] Dong, G., Pei, J.: Sequence Data Mining. Volume 33 of Advances in Database Systems. Kluwer (2007)

[3] Zaki, M.J.: Sequence mining in categorical domains: Incorporating constraints. In: CIKM, ACM (2000) 422–429

[4] 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, IEEE Computer Society (2001) 215–224

[5] Yan, X., Han, J., Afshar, R.: Clospan: Mining closed sequential patterns in large databases. In Barbará, D., Kamath, C., eds.: SDM, SIAM (2003)

[6] Wang, J., Han, J.: Bide: Efficient mining of frequent closed sequences. In: ICDE. (2004) 79–90

[7] Khiari, M., Boizumault, P., Crémilleux, B.: Constraint programming for mining n-ary patterns. In: CP. (2010) 552–567

[8] Raedt, L.D., Guns, T., Nijssen, S.: Constraint programming for itemset mining. In Li, Y., Liu, B., Sarawagi, S., eds.: KDD'08, ACM (2008)

[9] Jabbour, S., Sais, L., Salhi, Y.: Boolean satisfiability for sequence mining. In: CIKM. (2013) 649–658

[10] Pisanti, N., Crochemore, M., Grossi, R., Sagot, M.F.: Bases of motifs for generating repeated patterns with wild cards. IEEE/ACM Trans. Comput. Biology Bioinform. **2**(1) (2005) 40–50

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

[12] Raedt, L.D., Zimmermann, A.: Constraint-based pattern set mining. In: Proceedings of the Seventh SIAM International Conference on Data Mining. (2007)

[13] Crémilleux, B., Soulet, A.: Discovering knowledge from local patterns with global constraints. In: ICCSA (2). (2008) 1242–1257

[14] 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)

[15] Verfaillie, G., Jussien, N.: Constraint solving in uncertain and dynamic environments: A survey. Constraints **10**(3) (2005) 253–281

[16] Beldiceanu, N., Contejean, E.: Introducing global constraints in CHIP. Journal of Mathematical and Computer Modelling **20**(12) (1994) 97–123

[17] Pesant, G.: A regular language membership constraint for finite sequences of variables. In Wallace, M., ed.: CP'04. Volume 2239 of LNCS., Springer (2004) 482–495

[18] Tzvetkov, P., Yan, X., Han, J.: Tsp: Mining top-k closed sequential patterns. In: ICDM. (2003) 347–354

[19] Ji, X., Bailey, J., Dong, G.: Mining minimal distinguishing subsequence patterns with gap constraints. Knowl. Inf. Syst. **11**(3) (2007) 259–286

[20] Albert-Lorincz, H., Boulicaut, J.F.: Mining frequent sequential patterns under regular expressions: A highly adaptive strategy for pushing contraints. In: Third SIAM Intern.Conf. on Data Mining, SIAM (2003)

[21] Guns, T., Nijssen, S., Raedt, L.D.: k-pattern set mining under constraints. IEEE Trans. Knowl. Data Eng. **25**(2) (2013) 402–418

[22] Negrevergne, B., Dries, A., Guns, T., Nijssen, S.: Dominance programming for itemset mining. In: ICDM. (2013)

[23] Ugarte, W., Boizumault, P., Loudni, S., Crémilleux, B., Lepailleur, A.: Mining (soft-) skypatterns using dynamic CSP. In: CPAIOR. (2014)

[24] Coquery, E., Jabbour, S., Saïs, L., Salhi, Y.: A sat-based approach for discovering frequent, closed and maximal patterns in a sequence. In: ECAI. (2012) 258–263

[25] 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)

[26] Béchet, N., Cellier, P., Charnois, T., Crémilleux, B.: Sequential pattern mining to discover relations between genes and rare diseases. In: CBMS. (2012)

[27] Schmid, H.: Probabilistic part-of-speech tagging using decision trees. In: Intern. Conf. on New Methods in Language Processing. (1994)

[28] Quiniou, S., Cellier, P., Charnois, T., Legallois, D.: What About Sequential Data Mining Techniques to Identify Linguistic Patterns for Stylistics? In: CiCLing. (2012)

[7] This pattern is extracted from sentences such as " venu des profondeurs de la nuit (come from the depth of the night) ", " prise du vertige de la faim (feel the hunger dizziness) "