Boolean Satisfiability for Sequence Mining

Said Jabbour
jabbour@cril.fr

Lakhdar Sais
sais@cril.fr

Yakoub Salhi
salhi@cril.fr

CRIL - CNRS, University of Artois
F-62307 Lens Cedex
France

ABSTRACT

In this paper, we propose a SAT-based encoding for the problem of discovering frequent, closed and maximal patterns in a sequence of items and a sequence of itemsets. Our encoding can be seen as an improvement of the approach proposed in [8] for the sequences of items. In this case, we show experimentally on real world data that our encoding is significantly better. Then we introduce a new extension of the problem to enumerate patterns in a sequence of itemsets. Thanks to the flexibility and to the declarative aspects of our SAT-based approach, an encoding for the sequences of itemsets is obtained by a very slight modification of that for the sequences of items.

Categories and Subject Descriptors

F.4.1 [**Mathematical logic and formal languages**]: Mathematical Logic—*Logic and constraint programming*; H.2.8 [**Database management**]: Database applications—*Data mining*

Keywords

Data mining; Propositional satisfiability and modeling

1. INTRODUCTION

Frequent sequence data mining is the problem of discovering frequent patterns shared across time among an input data-sequence. Sequence mining is a central task in computational biology, temporal sequence analysis and text mining.

In this paper, we consider the pattern discovery problem for a specific class of patterns with wildcards in a sequence. The data-sequence can be seen as a sequence of items, while the pattern can be seen as a subsequence that might contains wildcards or jokers in the sense that they match any item [18, 20, 2]. At the first sight, allowing wildcards to occur in a pattern can be seen as an even more restrictive type of patterns in general. However as argued in [18] "*studying patterns with wildcards has the merit of capturing one important aspect of biological features that often concerns isolated positions inside a motif that are not part of the biological feature being captured*". The enumeration problem for maximal and closed motifs with wildcards has been investigated recently by several authors [19, 20, 2, 8]. One of the major problem is that the number of motifs can be of exponential size. This combinatorial explosion is tackled using different approaches. For example, in Parida et al. [18], the number of patterns is reduced by introducing the maximal non redundant q-patterns (patterns occurring at least q times in a sequence). Arimura and Uno [2] proposed a polynomial space and polynomial delay algorithm MaxMotif for maximal pattern discovery of the class of motifs with wildcards.

In this work, we follow the constraint programming (CP) based data mining framework proposed recently by Luc De Raedt et al. in [10] for itemset mining. This new framework offers a declarative and flexible representation model. New constraints often require new implementations in specialized approaches, while they can be easily integrated in such a CP framework. It allows data mining problems to benefit from several generic and efficient CP solving techniques. The authors show how some typical constraints (e.g. frequency, maximality, monotonicity) used in itemset mining can be formulated for use in CP [14]. This first study leads to the first CP approach for itemset mining displaying nice declarative opportunities without neglecting efficiency. More recently, Coquery et al. [8] have proposed a SAT-Based approach for Discovering for enumerating frequent, closed and maximal patterns with wildcards in a sequence of items. In this paper, we first propose a new SAT encoding of the problem of enumerating frequent, closed and maximal patterns with wildcards in a sequence of items. Our contribution can be seen as an improvement of the approach proposed in [8]. Indeed, the experimental results clearly show that the new encoding is significantly better than the original SAT encoding proposed in [8].

Encouraged by these promising results, we propose in our second contribution a new variant of the problem of discovering patterns with wildcards in a sequence, by considering a sequence of itemsets instead of a sequence of items. In this extension the emptyset will simply play the same role as the wildcard symbol. Indeed, one can use the emptyset to match any itemset. This new problem admits some similarities and differences with the classical sequential pattern

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2505577 .

mining problem introduced in [1]. Indeed, given an alphabet or a set of items Σ, in both problems we consider a sequence s as an ordered list of itemsets $s_0, \ldots, s_n$ where $s_i \subseteq \Sigma$ for $i = 0, \ldots, n$. However, the first difference resides in the definition of a subsequence. Indeed, in the sequential patterns, we say that s' is a subsequence of s if there exists a one-to-one order-preserving function f that maps (inclusion relation) itemsets in s' with itemsets in s. In our new setting, the notion of subsequence is defined w.r.t. to a given location and by using empty itemsets as wildcards. The other difference is that in the sequential pattern mining we consider a database of sequences of itemsets, while in our setting, we consider only a single sequence of itemsets. These differences leads also to different definitions of the notions of support, closeness and maximality. In summary, our new problem of discovering patterns in a sequence of itemsets can be seen as a simple and natural extension of the same problem in a sequence of items. As we can show later, the SAT encoding can be derived from the one used for the sequences of items with a very slight modification demonstrating its flexibility.

The paper is organized as follows. In the next section, we give a short overview of necessary definitions and notations about Boolean Satisfiability (SAT) and Frequent Pattern mining in a Sequence of items (FPS). The extension to the Frequent Pattern mining in a Sequence of Itemsets (FPSI) is presented in Section 3, followed by a discussion of some related works. Then our new SAT-based approach for FPS is described in Section 4, while the closeness and maximality constraints are discussed in Section 5. In Section 6, we show how the SAT encoding of FPSI, can be obtained by a slight modification of the SAT encoding of FPS. Finally, experimental results are conducted and discussed before concluding.

2. PRELIMINARIES

2.1 Boolean Satisfiability

In this section, we introduce the Boolean satisfiability problem, called SAT. It corresponds to the problem of deciding if a formula of propositional classical logic is consistent or not. It is one of the most studied NP-complete decision problem. In this work, we consider the associated problem of Boolean model enumeration.

We consider the conjunctive normal form (CNF) representation for the propositional formulas. A *CNF formula* Φ is a conjunction of clauses, where a *clause* is a disjunction of literals. A *literal* is a positive (p) or negated ($\neg p$) propositional variable. The two literals p and $\neg p$ are called *complementary*.

A CNF formula can also be seen as a set of clauses, and a clause as a set of literals. Let us recall that any propositional formula can be translated to CNF using linear Tseitin's encoding [22]. We denote by $Var(\Phi)$ the set of propositional variables occurring in Φ.

An *interpretation* $\mathcal{B}$ of a propositional formula Φ is a function which associates a value $\mathcal{B}(p) \in \{0, 1\}$ (0 corresponds to $false$ and 1 to $true$) to the variables $p \in Var(\Phi)$. A *model* of a formula Φ is an interpretation $\mathcal{B}$ that satisfies the formula. *SAT problem* consists in deciding if a given formula admits a model or not.

We denote by $\bar{l}$ the complementary literal of l, i.e., if $l = p$ then $\bar{l} = \neg p$ and if $l = \neg p$ then $\bar{l} = p$. For a set of literals L, $\bar{L}$ is defined as $\{\bar{l} \mid l \in L\}$. Moreover, $\overline{\mathcal{B}}$ ($\mathcal{B}$ is an interpretation over $Var(\Phi)$) corresponds to the clause $\bigvee_{p \in Var(\Phi)} f(p)$, where if $\mathcal{B}(p) = 0$ then $f(p) = p$, otherwise $f(p) = \neg p$.

Let us informally describe the most important components of modern SAT solvers. They are based on a reincarnation of the historical Davis, Putnam, Logemann and Loveland procedure, commonly called DPLL [9]. It performs a backtrack search; selecting at each level of the search tree, a decision variable which is set to a Boolean value. This assignment is followed by an inference step that deduces and propagates some forced unit literal assignments. This is recorded in the implication graph, a central data-structure, which encodes the decision literals together with there implications. This branching process is repeated until finding a model or a conflict. In the first case, the formula is answered satisfiable, and the model is reported, whereas in the second case, a conflict clause (called learnt clause) is generated by resolution following a bottom-up traversal of the implication graph [17, 24]. The learning or conflict analysis process stops when a conflict clause containing only one literal from the current decision level is generated. Such a conflict clause asserts that the unique literal with the current level (called asserting literal) is implied at a previous level, called assertion level, identified as the maximum level of the other literals of the clause. The solver backtracks to the assertion level and assigns that asserting literal to *true*. When an empty conflict clause is generated, the literal is implied at level 0, and the original formula can be reported unsatisfiable.

In addition to this basic scheme, modern SAT solvers use other components such as activity based heuristics and restart policies. An extensive overview about propositional Satisfiability can be found in [6, 15].

2.2 Frequent Pattern Mining in a Sequence of Items (FPS)

In this section, we present the frequent pattern mining problem of enumerating frequent, closed and maximal patterns with wildcards in a sequence of items [18, 20, 2]. Let us first give some preliminary definitions and notations.

Sequences of items.

Let Σ be a finite set of items, called alphabet. A *sequence of items* s over Σ is a simple sequence of symbols $s_0 \cdots s_{n-1}$ belonging to Σ. We denote by $|s|$ its length and by $\mathcal{P}_s$ the set $\{0, \ldots |s| - 1\}$ of all the locations of its symbols. *A wildcard* is a new symbol $\circ$ which is not in Σ. This symbol matches any symbol of the alphabet.

Pattern.

A *pattern* over Σ is a sequence $p = p_0 \ldots p_{m-1}$, where $p_0 \in \Sigma$, $p_{m-1} \in \Sigma$ and $p_i \in \Sigma \cup \{\circ\}$ for $i = 1, \ldots, m-2$. We say that p is included in $s = s_0 \ldots s_{n-1}$ at the location $l \in \mathcal{P}_s$, denoted $p \preceq_l s$, if $\forall i \in \{0 \ldots m-1\}$, $p_i = s_{l+i}$ or $p_i = \circ$. We also say that p is included in s, denoted $p \preceq s$, if $\exists l \in \mathcal{P}_s$ such that $p \preceq_l s$. The *cover* of p in s is defined as the set $\mathcal{L}_s(p) = \{l \in \mathcal{P}_s | p \preceq_l s\}$. Moreover, The *support* of p in s is defined as the value $|\mathcal{L}_s(p)|$.

FPS problem.

Let s be a sequence, p a pattern and $\lambda \geq 1$ a minimal support threshold, called also a quorum. We say that p is a *frequent pattern in s w.r.t. λ* if $|\mathcal{L}_s(p)| \geq \lambda$. *The frequent pattern mining problem in a sequence of items (FPS)* consists in computing the set $\mathcal{M}_s^\lambda$ of all the frequent patterns w.r.t. λ. For instance, let us consider the sequence $s = aaccbcabcba$ and then pattern $p = a \circ c$. We have $\mathcal{L}_s(p) = \{0, 1, 6\}$, since $p \preceq_0 s$, $p \preceq_1 s$ and $p \preceq_6 s$. In this case, if we consider that the minimal support threshold is equal to the value 3, then the pattern p is a frequent pattern of s.

Closed and Maximal Patterns.

A frequent pattern p of a sequence s is said to be *closed* if for any frequent pattern q satisfying $q \succ p$, there is no integer d such that $\mathcal{L}_s(q) = \mathcal{L}_s(p) + d$, where $\mathcal{L}_s(p) + d = \{l + d | l \in \mathcal{L}_s(p)\}$. Moreover, it is said to be *maximal* if for any frequent pattern q, $q \not\succ p$. Clearly, the set of closed frequent patterns (resp. maximal frequent patterns) is a condensed representation of the set of frequent patterns. Indeed, the frequent patterns can be obtained from the closed (resp. maximal) ones by replacing items with wildcards.

Note that if p_1 and p_2 are two patterns such that $p_1 \preceq p_2$, then if $|\mathcal{L}_s(p_2)| \geq \lambda$ then $|\mathcal{L}_s(p_1)| \geq \lambda$. This property is called anti-monotonicity.

3. FREQUENT PATTERN MINING IN A SEQUENCE OF ITEMSETS (FPSI)

In this section, we define a new variant of the problem of discovering patterns with wildcards in a sequence, by considering a sequence of itemsets instead of a sequence of items. The role of wildcard symbol is nicely played by the empty itemset as it match any itemset. As mentioned in the introduction, this new problem admits some similarities and differences with the classical sequential pattern mining problem introduced in [1]. The main difference resides in the definition of the notion of subsequence (inclusion), where empty itemsets are used as wildcards, and in the use or not of a single or several sequences

A *sequence of itemsets* s over an alphabet Σ is defined as a sequence $s_0, \ldots, s_{n-1}$, where $s_i \subseteq \Sigma$ for $i = 0, \ldots, n-1$. Similarly to the sequences of items, we denote by $|s|$ its length ($|s| = n$) and by $\mathcal{P}_s$ the set $\{0, \ldots |s| - 1\}$ of the locations.

A *pattern* $p = p_0, \ldots, p_{m-1}$ over Σ is also defined as a sequence of itemsets where the first and the last elements are different from the empty itemset. In this context, let us mention that we do not need the wildcard symbol. Indeed, one can use the empty itemset to match any itemset. Furthermore, we say that p is included in $s = s_0 \ldots s_{n-1}$, denoted $p \preceq_l s$, at the location $l \in \mathcal{P}_s$ if $\forall i \in \{0 \ldots m-1\}$, $p_i \subseteq s_{l+i}$. The relation $\preceq$ and the set $\mathcal{L}_s(p)$ are defined in the same way as in the case of the sequences of items. The *cover* (resp. *support*) of p in s is defined as the set $\mathcal{L}_s(p)$ (resp. as the value $|\mathcal{L}_s(p)|$).

The frequent, closed and maximal patterns are also defined in the same way. For instance, a frequent pattern p of a sequence s is said to be *closed* if for any frequent pattern q satisfying $q \succ p$, there is no integer d such that $\mathcal{L}_s(q) = \mathcal{L}_s(p) + d$, where $\mathcal{L}_s(p) + d = \{l + d | l \in \mathcal{L}_s(p)\}$. The frequent patterns can be obtained from the closed (resp. maximal) ones by replacing itemsets with their subsets.

For example, let us consider the sequence of itemsets $s = \{a, b\}, \{a, b\}, \{c, d\}, \{c, e\}, \{f\}, \{g\}, \{d\}, \{a, b, d\}, \{f\}, \{c\}$ and the pattern $p = \{a, b\}, \{\}, \{c\}$. If we set the minimal support threshold to 3, then p is a frequent pattern in s, since $\mathcal{L}_s(p) = \{0, 1, 7\}$. The pattern p is also a closed frequent pattern, but $p' = \{a\}, \{\}, \{c\}$ is not closed, since $p \prec p'$.

The pattern mining task that we consider in the sequences of itemsets allows to exhibit a high degree of self similarity for better understandings of large volumes of data. For instance, a sequence of itemsets can be seen as a record of the articles bought by a customer over a period of time. In such a case, a frequent pattern could be "the customer bought acetylsalicylic acid two days after buying beer and wine in 20% of the days from 2008 to 2012".

3.1 Related Works and Motivations

SAT-based encodings for enumerating frequent, closed and maximal patterns in the sequences of items have been proposed in [8]. They follows the constraint programing (CP) based approach proposed recently by Luc De Raedt et al. in [10] for itemset mining. The SAT and CP based approaches in data mining are proposed in order to offer declarative and flexible frameworks. Indeed, new constraints require often new implementations in specialized approaches, while they can be easily integrated in such frameworks.

In this paper, we propose a SAT-based encodings for enumerating frequent, closed and maximal patterns in the sequences of items and the sequences of itemsets. The choice of SAT comes from our desire to exploit the efficiency of modern SAT solvers [6]. In this context, our encodings can be seen as an improvement and an extension of the encodings proposed in [8]. Indeed, we show experimentally on real world data that our encodings are better than those in [8]. Furthermore, we show that encodings in the case of the sequences of itemsets are obtained by a very slight modification of that for the sequences of items.

4. A NEW SAT-BASED APPROACH FOR FPS

We describe here our new Boolean encoding for the problem of enumerating the frequent patterns in a sequence of items FPS. The base idea consists in using a propositional variable to represent the location of an element of the alphabet in the candidate pattern. Moreover, we use the well-known cardinality constraint to reason about the support of the candidate pattern.

Let $\Sigma = \{a_1, \ldots, a_m\}$ be an alphabet, s a sequence over Σ of length n and λ a minimal support threshold. We associate to each character a appearing in s a set of k_a propositional variables $p_{a,0}, \ldots, p_{a,(k_a-1)}$ such that $k_a = min(max(\mathcal{L}_s(a)) + 1, n - \lambda + 1)$. The variable $p_{a,i}$ means that a is in the candidate pattern at the location i. In fact, that explains why we associate only $min(max(\mathcal{L}_s(a)) + 1, n - \lambda + 1)$ variables to each character a, because $\{0, \ldots, min(max(\mathcal{L}_s(a)), n - \lambda)\}$ corresponds to the set of all possible locations of a in the candidate patterns.

We first need to encode that the first symbol must be a solid character (different from the wildcard symbol). This property is expressed by the following simple clause:

$$\bigvee_{a \in \Sigma} p_{a,0} \tag{1}$$

The following constraint composed of binary clauses allows us to capture the locations where the candidate pattern does not appear:

$$\bigwedge_{a \in \Sigma, 0 \leq l \leq n-1, 0 \leq i \leq k_a - 1} (p_{a,i} \wedge s_{l+i} \neq a) \rightarrow b_l \tag{2}$$

where $b_0, \ldots, b_{n-1}$ are n new propositional variables. In the previous formula $b_j = 1$ if the candidate pattern does not appear in s at the location j. Let us recall that, in classical propositional logic, we have $A \rightarrow B := \neg A \vee B$, and that explains why the previous formula can be seen as a set of binary clauses (the expressions of the for $s_{l+i} \neq a$ are constants, i.e. $s_{l+i} \neq a \in \{0, 1\}$).

In the problem of enumerating all the frequent patterns in s w.r.t. λ, we need to express that the candidate pattern occurs at least λ times. This property is obtained by the following *cardinality constraint*:

$$\sum_{l=0}^{n-1} b_l \leq n - \lambda \tag{3}$$

Indeed, if this constraint is not satisfied, then we know that there exist at least $n - \lambda + 1$ locations where the candidate pattern does not appear. This is equivalent to say that there exist at most $\lambda - 1$ locations where the candidate pattern appears, i.e. it is not frequent. Otherwise, there exist at least λ locations of the candidate pattern , i.e. it is frequent. Hence this constraint allows us to reason about the support of the considered candidate pattern an to decide whether it is greater or equal to the minimal support threshold or not.

The previous constraint involves the well known cardinality constraint (0/1 linear inequality). Several polynomial encoding of this kind of constraints into a CNF formula have been proposed in the literature. The first linear encoding of general linear inequalities to CNF have been proposed by J. P. Warners [23]. Recently, efficient encodings of the cardinality constraint to CNF have been proposed, most of them try to improve the efficiency of constraint propagation (e.g. [4, 21, 3, 16]).

PROPOSITION 1. *The problem of enumerating all frequent patterns in a given sequence s is expressed by the constraints (1), (2) and (3).*

PROOF. We first prove that if $p = a_0, \ldots, a_{k-1}$ is a frequent pattern, then there exists an extension of its corresponding Boolean interpretation $\mathcal{B}_p$ which is a model of (1), (2) and (3). Note that $\mathcal{B}_p$ is defined as follows: for all $a \in \Sigma$ and for all $i \in \{0, \ldots, k_a\}$, if $a_i = a$ then $\mathcal{B}_p(p_{a,i}) = 1$. One can easily see that the constraint (1) is satisfied by $\mathcal{B}_p$, since $\mathcal{B}_p(p_{a_0,0}) = 1$. Let us now extend the Boolean Interpretation $\mathcal{B}_p$ to the variables $b_0, \ldots, b_{n-1}$. This extension is obtained as follows: for all $0 \leq i \leq n-1$, if $p \not\preceq_i s$ then $\mathcal{B}_p(b_i) = 1$. Clearly this extension corresponds to a Boolean interpretation that satisfies (2). Finally, it also satisfies (3), since p is a frequent pattern, i.e. its support is greater or equal to the minimal support threshold λ.

Conversely, we have to prove that if a Boolean interpretation $\mathcal{B}$ is a model of (1), (2) and (3), then there exists a unique pattern $p_{\mathcal{B}}$ corresponding to $\mathcal{B}$ which is frequent. Note that, using the constraints (2) and (3), we have, for all $a, a' \in \Sigma$ and for all $i \in \{0, \ldots, k_a - 1\}$, if $\mathcal{B}(p_{a,i}) = \mathcal{B}(p_{a',i}) = 1$, then $a = a'$. Indeed, if there exists $a \neq a'$ such that $\mathcal{B}(p_{a,i}) = \mathcal{B}(p_{a',i}) = 1$, then we get $\sum_{l=0}^{n-1} b_l = n$ and this is in contradiction with $\sum_{l=0}^{n-1} b_l \leq n - \lambda$ ($\lambda \neq 0$). Furthermore, using the constraint (1), we know that the first symbol of p is different from $\circ$. Therefore, we deduce that there exists a unique pattern associated to $\mathcal{B}$. This pattern corresponds to $p_{\mathcal{B}} = a_0 \cdots a_{k-1}$ such that, for all $i \in \{0, \ldots, k-1\}$ with $a_i \neq \circ$, $\mathcal{B}(p_{a_i,i}) = 1$, and $\mathcal{B}(p_{a,i}) = 0$ for all $a \in \Sigma$ with $a \neq a_i$. Moreover, using the constraint (2), we know that if $\mathcal{B}(b_i) = 1$, then $p \not\preceq_i s$. Hence, using the cardinality constraint (3), we deduce that the support of p is greater or equal to the support threshold λ. □

Example. Consider the frequent pattern mining problem in the case of the sequence *aabb* with 2 as minimal support threshold. Our encoding corresponds to the following formulae:

$p_{a,0} \vee p_{b_0}$
$p_{a,0} \rightarrow (b_2 \wedge b_3)$
$p_{a,1} \rightarrow (b_1 \wedge b_2 \wedge b_3)$
$p_{a,2} \rightarrow (b_0 \wedge b_1 \wedge b_2 \wedge b_3)$
$p_{b,0} \rightarrow (b_0 \wedge b_1)$
$p_{b,1} \rightarrow (b_0 \wedge b_3)$
$p_{b,2} \rightarrow (b_2 \wedge b_3)$
$b_0 + b_1 + b_2 + b_3 \leq 2$

Note that, for all Boolean interpretation $\mathcal{B}$, if $\mathcal{B}(p_{a,i}) = \mathcal{B}(p_{b,i})$, then $b_0 + b_1 + b_2 + b_3 = 4$. Hence we cannot have different solid characters at the same position. Moreover, using the last constraint, for all Boolean interpretation $\mathcal{B}$ which is a model of the encoding, we must have $\mathcal{B}(p_{a,1}) = \mathcal{B}(p_{a,2}) = 0$ and $\mathcal{B}(p_{a,0}) \neq \mathcal{B}(p_{b,1})$. If we describe each Boolean model of the formula by a subset of $\{p_{a,0}, p_{a,1}, p_{a,2}, p_{b,0}, p_{b,1}, p_{b,2}\}$, then we obtain as models $\{p_{a,0}\}$, $\{p_{b,0}\}$ and $\{p_{a,0}, p_{b,2}\}$. These Boolean models correspond to the patterns a, b and $a \circ b$.

Note that in order to consider the frequent patterns with at least *min* solid characters, we just have to add the following constraint:

$$\sum_{a \in \Sigma, 0 \leq i \leq k_a - 1} p_{a,i} \geq min \tag{4}$$

Conversely, in order to only consider the frequent patterns with at most *max* solid characters, we add:

$$\sum_{a \in \Sigma, 0 \leq i \leq k_a - 1} p_{a,i} \geq max \tag{5}$$

Moreover, the combination of the two previous constraints allows us to only consider with the number of solid characters between *min* and *max*. Let us mention that such extensions show that our approach is flexible.

5. ENUMERATING CLOSED AND MAXIMAL MOTIFS

5.1 Enumerating Closed Motifs (CPS)

In order to provide constraints allowing to enumerate the closed frequent patterns, we associate to each symbol a a set of $k_a + (n - min(\mathcal{L}_s(a)) - 1)$ propositional variables:

$$p_{a,-k'_a}, \ldots, p_{a,k_a-1}$$

where $k'_a = n - min(\mathcal{L}_s(a)) - 1$. Similarly to our previous encoding, the propositional variables $p_{a,0}, \ldots, p_{a,k_a-1}$ allow us to reason about the possible locations of a in the candidate pattern. The variables with negative indices are used to force the candidate pattern to be closed. Our encoding of the problem of enumerating the closed frequent patterns in a sequence of items CPS is obtained by extending the previous one with new constraints.

We first have to capture all the locations where the candidates pattern appears. This is obtained by the following constraint:

$$\bigwedge_{l=0}^{n-1}(b_l \rightarrow \bigvee_{a\in\Sigma, 0\leq i\leq k_a-1}(p_{a,i} \wedge s_{l+i} \neq a)) \quad (6)$$

Indeed, the previous constraint combined to the constraint (2) allows us to obtain that, if the Boolean interpretation $\mathcal{B}$ is a model of the constraints (1), (2) and (6), then the candidate pattern that corresponds to $\mathcal{B}$ appears only in the locations $\{0 \leq l \leq n-1 | \mathcal{B}(b_l) = 0\}$.

Now, we introduce a necessary, but not sufficient, constraint, w.r.t. the previous constraints, for obtaining a closed frequent pattern:

$$\bigwedge_{a\in\Sigma, 0\leq i\leq k_a-1}(\bigwedge_{l=0}^{n-1}\overline{b_l} \rightarrow s_{l+i} = a) \rightarrow p_{a,i} \quad (7)$$

Intuitively, the previous constraint maximizes the number of the symbols different from wildcard on the right side of the symbol represented by the propositional variable having 0 as index.

We now define a constraint with the propositional variables having the negative indices. Conversely to the previous constraint, the following constraint allows us to to maximize the number of the symbols different from wildcard on the left side:

$$\bigwedge_{a\in\Sigma, 1\leq i\leq k'_a}(\bigwedge_{l=0}^{n-1}\overline{b_l} \rightarrow s_{l-i} = a) \leftrightarrow p_{a,-i} \quad (8)$$

Let us note that if the Boolean interpretation $\mathcal{B}$ is a model of $(1) \wedge (2) \wedge (3) \wedge (6) \wedge (7) \wedge (8)$ and $\mathcal{B}(p_{a,i}) = 1$, then, for all $b \in \Sigma$ such that $a \neq b$ and $p_{b,i}$ exists, $\mathcal{B}(p_{b,i}) = 0$ holds. This property is mainly obtained from the constraints (2) and (8) (see arguments used in the proof of Proposition 1). A closed motif is obtained from a model by using the propositional variables associated to the elements of Σ and evaluated to 1 by this model. Let $p_{a_0,i_0}, p_{a_1,i_1}, \ldots, p_{a_{k-1},i_{k-1}}$ be these variables. In this case, the closed motif is:

$$a_0 \overbrace{\circ \cdots \circ}^{i_1-i_0-1} a_1 \cdots a_{k-1}$$

We now provide another encoding without using propositional variables with negative indices. The idea consists in excluding each interpretation whenever its corresponding pattern is not closed. This encoding is obtained by replacing the constraint (8) with the following constraint:

$$\bigwedge_{a\in\Sigma, 1\leq i\leq k'_a} \neg(\bigwedge_{l=0}^{n-1}\overline{b_l} \rightarrow s_{l-i} = a) \quad (9)$$

By the previous constraint, we simply force the candidate pattern to be closed.

PROPOSITION 2. *The problem of enumerating all the closed frequent patterns in a given sequence s is expressed by the constraints (1), (2), (3), (6), (7) and (9).*

PROOF. Let $p = a_0, \ldots, a_{k-1}$ be a closed frequent pattern. We define its corresponding Boolean interpretation $\mathcal{B}_p$ as follows: for all $a \in \Sigma$ and for all $i \in \{0, \ldots, k_a\}$, if $a_i = a$ then $\mathcal{B}_p(p_{a,i}) = 1$. We extend $\mathcal{B}_p$ to the propositional variables $\{b_0, \ldots, b_{n-1}\}$ as follows: $p \not\preceq_i s$ iff $\mathcal{B}_p(b_i) = 1$, for $i = 0, \ldots, n-1$. By using similar arguments as in our proof of Proposition 1, we know that $\mathcal{B}_p$ satisfies the constraints (1), (2) and (3). It also satisfies (6), since if $\mathcal{B}_p(b_i) = 1$ then $p \not\preceq_i s$. In this context, the support of p is equal to $|\{b_i \mid \mathcal{B}_p(b_i) = 0\}|$. This allows us to deduce that $\mathcal{B}_p$ satisfies (7) and (9), since p is closed. Indeed, if (7) or (9) are not satisfied by $\mathcal{B}_p$, then there exists a pattern p' such that $p \prec p'$ with a support greater or equal to that of p and we get a contradiction because that means that p is not a closed pattern.

Conversely, consider $\mathcal{B}$ a model of (1), (2), (3), (6), (7) and (9). Using the constraints (2) and (3), we have, for all $a, a' \in \Sigma$ and for all $i \in \{0, \ldots, k_a - 1\}$, if $\mathcal{B}(p_{a,i}) = \mathcal{B}(p_{a',i}) = 1$, then $a = a'$. Hence, there exists a unique pattern associated to $\mathcal{B}$ that corresponds to $p_{\mathcal{B}} = a_0 \cdots a_{k-1}$ such that, for all $i \in \{0, \ldots, k-1\}$ with $a_i \neq \circ$, $\mathcal{B}(p_{a_i,i}) = 1$, and $\mathcal{B}(p_{a,i}) = 0$ for all $a \in \Sigma$ with $a \neq a_i$. Using Prposition 1, we know that the pattern $p_{\mathcal{B}}$ is frequent. The constraint (6) allows us to obtain that the support of p is equal to $|\{b_i \mid \mathcal{B}(b_i) = 0\}|$. Using the constraints (7) and (9), we now that there is no frequent pattern q having the same support as p such that $p \prec q$. Therefore, we deduce that p is a closed frequent pattern. □

Note that in order to enumerate all frequent patterns without any condition on their supports, we only have to remove the constraint 3.

5.2 Enumerating Maximal Motifs (MPS)

In our encoding of the problem of enumerating the maximal frequent patterns in a sequence of items MPS, we only use the propositional variables associated to the elements of Σ with positive indices, i.e. we associate to each symbol a a set of k_a propositional variables $p_{a,0}, \ldots, p_{a,(k_a-1)}$. Our encoding of MPS is obtained by extending the one of FPS in a similar way as our encoding of CPS.

In order to enumerate the maximal frequent patterns, we

need to capture all the locations where the candidates pattern appears. To this end, similarly to our encodings of CPS, we use the constraint (6). Indeed, the combination of the constraints (2) and (6) allows us to obtain, if $\mathcal{B}$ is a Boolean model of these two constraints, then $\{0 \leq l \leq n-1 | \mathcal{B}(b_l) = 0\}$ corresponds to the set of the locations where the candidate pattern appears.

We now provide the constraint allowing to maximize the number of symbols different from wildcard on the right side of the symbol represented by the propositional variable having 0 as index:

$$\bigwedge_{a \in \Sigma, 1 \leq i \leq k_a - 1} (\sum_{l=0}^{n-1} \overline{b_l} \wedge s_{l+i} = a \geq \lambda) \rightarrow p_{a,i} \quad (10)$$

Intuitively, the constraint means that if $p = a_0 \cdots a_{k-1}$ is the patter candidate and there exists $a \in \Sigma$ such that then pattern $a_0 \cdots a_{k-1} \circ \cdots \circ a$ have the same support as p, then p is not a maximal frequent pattern.

Coversely to the previous constraint, we finally introduce the constraint allowing to maximize the number of symbols different from wildcard on the left side of the symbol represented by the propositional variable having 0 as index:

$$\bigwedge_{a \in \Sigma, 1 \leq i \leq k'_a} \neg(\sum_{l=0}^{n-1} \overline{b_l} \wedge s_{l-i} = a \geq \lambda) \quad (11)$$

One can easily see that it is equivalent to the following constraint:

$$\bigwedge_{a \in \Sigma, 1 \leq i \leq k'_a} \sum_{l=0}^{n-1} \overline{b_l} \wedge s_{l-i} = a \leq \lambda - 1 \quad (12)$$

Indeed, the constraint $\neg(\sum_{l=0}^{n-1} x \geq \lambda)$ is equivalent to cardinality constraint $\sum_{l=0}^{n-1} x \leq \lambda - 1$.

PROPOSITION 3. *The problem of enumerating all the maximal frequent patterns in a given sequence s is expressed by the constraints (1), (2), (3), (6), (10) and (12).*

PROOF. Similar to our proof of Proposition 2. □

Let us mention that we can also use the constraints (4) and (5) to reason about the number of the solid characters in the considered patterns in the cases of CPS and MPS. Furthermore, we can use a constraint in order to only consider the closed and maximal patterns with support between λ and λ'. This constraint is the following:

$$\sum_{l=0}^{n-1} b_l \geq n - \lambda' \quad (13)$$

6. SAT-BASED ENCODINGS AND SEQUENCE OF ITEMSETS

In this section, we extend our SAT-based approach for discovering frequent, closed and maximal patterns in a sequence of itemsets. We will show that our encodings in this case can be obtained from the previous ones with a very slight modification.

Our encoding of the problem of enumerating the frequent patterns in a sequence of itemsets FPSI can be easily obtained from the one of FPS. We only have to replace the equalities of the form $s_{l+i} \neq a$ with $a \notin s_{l+i}$:

$$\bigvee_{a \in \Sigma} p_{a,0} \quad (14)$$

$$\bigwedge_{a \in \Sigma, 0 \leq l \leq n-1, 0 \leq i \leq k_a - 1} (p_{a,i} \wedge a \notin s_{l+i}) \rightarrow b_l \quad (15)$$

$$\sum_{l=0}^{n-1} b_l \leq n - \lambda \quad (16)$$

In this case, the variable $p_{a,i}$ means that the symbol a is in the candidate pattern in the itemset at the location i. Let us recall that we use the empty itemset as wildcard.

We denote by CPSI (resp. MPSI) the problem of enumerating the closed (resp. maximal) frequent patterns in a sequence of itemsets. Similarly to FPSI, Boolean encodings of CPSI and MPSI can be directly obtained from the ones of respectively CPS and MPS by replacing the expressions of the form $s_{l+i} \neq a$ (resp. $s_{l+i} = a$) with $a \notin s_{l+i}$ (resp. $a \in s_{l+i}$).

Constraints of closeness:

$$\bigwedge_{l=0}^{n-1} (b_l \rightarrow \bigvee_{a \in \Sigma, 0 \leq i \leq k_a - 1} (p_{a,i} \wedge a \notin s_{l+i})) \quad (17)$$

$$\bigwedge_{a \in \Sigma, 0 \leq i \leq k_a - 1} (\bigwedge_{l=0}^{n-1} \overline{b_l} \rightarrow a \in s_{l+i}) \rightarrow p_{a,i} \quad (18)$$

$$\bigwedge_{a \in \Sigma, 1 \leq i \leq k'_a} \neg(\bigwedge_{l=0}^{n-1} \overline{b_l} \rightarrow a \in s_{l-i}) \quad (19)$$

Constraints of maximality: to express the maximality, we add the following two constraints to (17):

$$\bigwedge_{a \in \Sigma, 1 \leq i \leq k_a - 1} (\sum_{l=0}^{n-1} \overline{b_l} \wedge a \in s_{l+i} \geq \lambda) \rightarrow p_{a,i} \quad (20)$$

$$\bigwedge_{a \in \Sigma, 1 \leq i \leq k'_a} \sum_{l=0}^{n-1} \overline{b_l} \wedge a \in s_{l-i} \leq \lambda - 1 \quad (21)$$

The slight modification of our encodings in the case of the sequences of items in order to obtain encodings for the sequences of itemsets clearly shows the high flexibility of our proposed framework.

7. IMPLEMENTATION AND EXPERIMENTS

In our study, we carried out a preliminary experimental evaluation of our proposed approaches using two different datasets.

1. *Bioinformatics*: proteinic data encoded as a sequence of items, where an item is an amino-acid [1].

[1] http://www.biomedcentral.com/1471-2105/11/175/additional/

2. *Synthetic datasets:* we use the well-known IBM itemset data generator[2] to derive datasets with different features. We used this generator to get sequences of itemsets, that can be seen as an ordered set of transactions.

To make fair our comparison with the approach proposed in [8], we adopted the similar choices in our implementations. First, as we deal with the problem of enumerating all the models of a given CNF formula encoding our sequence mining problem, we implemented a model enumeration solver based on the CDCL-based solver MiniSAT 2.2 [3]. To enumerate all the models, each time a model is found, we add only the negation of the sub-model restricted to the literals encoding the pattern to the formula and we restart the search. Secondly, as our SAT encodings include cardinality constraints, we also use the BDD encoding [5] using BoolVar/PB open source java library [4].

In the first experiment, we compare our new SAT encoding (noted CPS1) against the SAT encoding proposed in [8] (noted CPS2), on bioinformatics datasets (sequences of items). This first evaluation concerns the enumeration of frequent closed patterns with wild cards in a sequence of items. We consider a sequence of fixed length and we measure the evolution of computation time with respect to the minimal support threshold (quorum) λ. The quorum evolves linearly ($\lambda_0 = 5$ and $\lambda_i = \lambda_{i-1} + 5$). Several datasets have been considered, their evaluation shows similar behavior. The results obtained on a representative dataset are depicted in Figure 1. This experiment confirms that in the case of sequences of items, our new SAT-based sequence mining approach outperforms (in terms of CPU time) the approach proposed recently in [8]. We also obtain significant improvement w.r.t. the size of the encoding. For instance, if we consider the most difficult dataset of the Figure 1 (quorum $\lambda = 5$), the obtained CNF formula using our encoding contains about 19 millions of clauses and 3 millions of variables, whereas with the encoding proposed in [8] the formula contains about 30 millions of clauses and 7.5 millions of variables.

The second experiment concerns the new extension of the problem to the case of sequence of itemsets. Our goal is show the feasibility of our proposed extension and its associated encodings. For our evaluation, we considered synthetic datasets, generated using the approach outlined in [1] and also used by several authors (e.g. [12]). In our context, we only consider a single sequence of itemsets with different features (size of the sequence, number of items, average size of the itemsets). In Figure 2, we illustrate the results obtained on two datasets: dataset1 (size of the sequence = 100, avg. size of itemsets = 15, number of items = 40) and dataset2 obtained from dataset1 by cutting the sequence at 50th position. The quorum is also varied linearly as in the first experiment. The main observation that can be made, is that the hardness of the enumeration problem increase as the quorum decrease. Indeed, for smaller values of λ, the number of frequent closed patterns is huge, leading to even harder problems. However for higher values of λ, the enumeration problem becomes easy as the number of interesting patterns decreases.

[2]http://sourceforge.net/projects/ibmquestdatagen/
[3]MiniSAT: http://minisat.se/
[4]BoolVAR/PB : http://boolvar.sourceforge.net/

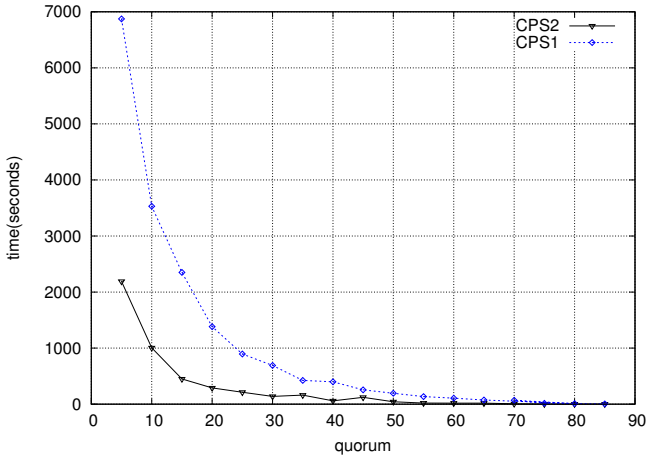

Figure 1: Bioinfo: time Vs quorum (Sequence of Items - Frequent Closed Patterns)

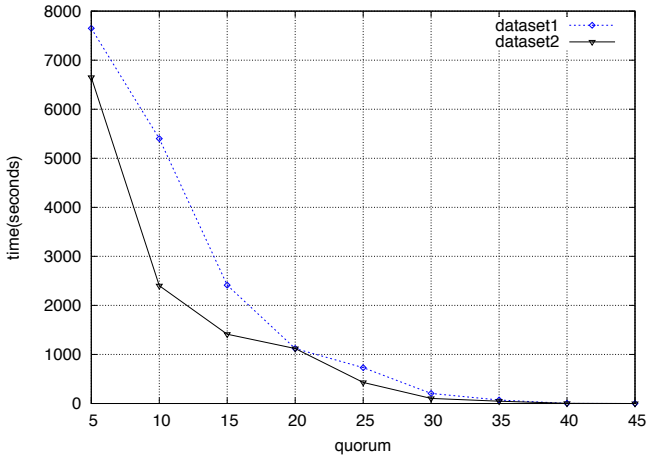

Figure 2: Synthetic datasets: time Vs quorum (Sequence of Itemsets - Frequent Closed Patterns)

As a summary, in the above experiments, we have shown that our encoding significantly improve the one proposed in [8, 7]. For comparison purposes, we used the same encoding of the cardinality constraint and also the same algorithm for enumerating all models of a CNF formula. We think that several room for future improvements can be obtained by using the state-of-the-art encoding of the cardinality [3], and by using more efficient model enumeration algorithm [13, 11, 25]. Obviously, our SAT based approach is less efficient than dedicated approaches such as the state-of-the-art algorithm proposed by Arimura et al. in [2]. However, our SAT model is declarative and highly flexible. Indeed, the SAT encoding for a sequence of itemsets is obtained with a very slight modification of the SAT encoding of a sequence of items. Also, one can easily combine several kind of constraints. Finally, SAT-based data mining benefits from the continuous progress of SAT community.

8. AKNOWLEDGMENTS

We thank the reviewers for their helpful comments. This work has been supported in part by the CNRS and the French ANR project *"DAG: Declarative Approaches for Enumerating Interesting Patterns"* under the Défis program 2009.

9. CONCLUSION AND PERSPECTIVES

The contributions of this paper are twofolds. First, we proposed an interesting improvement of the SAT-based encodings introduced in [8] for enumerating frequent, closed and maximal patterns with wildcards in a sequence of items. Secondly, we introduced a new and natural extension of the problem to deal with the sequences of itemsets. Interestingly, its encoding to SAT is obtained with a slight modification of the SAT encoding of the problem dealing with the sequences of items. This clearly shows the high flexibility of our proposed framework and opens several issues for future research. We first plan to investigate other variants of the problem such as sequences of sequences of items or itemsets. It would be interesting to extend our encoding with constraints on the form of the enumerated patterns (restriction on the number of consecutive wildcards, regular expressions, etc). Finally, on the Boolean satisfiability side, the design of efficient model generation procedures is an important issue for SAT-based data mining framework in general and to other important application domains.

10. REFERENCES

[1] R. Agrawal and R. Srikant. Mining sequential patterns. In A. L. P. C. Philip S. Yu, editor, *Proceedings of the Eleventh International Conference on Data Engineering (ICDE'1995)*, pages 3–14. IEEE Computer Society, 1995.

[2] H. Arimura and T. Uno. An efficient polynomial space and polynomial delay algorithm for enumeration of maximal motifs in a sequence. *Journal of Combinatorial Optimization*, 13, 2007.

[3] R. Asin, R. Nieuwenhuis, A. Oliveras, and E. Rodriguez-Carbonell. Cardinality networks: a theoretical and empirical study. *Constraints*, 16(2):195–221, 2011.

[4] O. Bailleux and Y. Boufkhad. Efficient CNF Encoding of Boolean Cardinality Constraints. In *9th International Conference on Principles and Practice of Constraint Programming - CP 2003*, pages 108–122, 2003.

[5] O. Bailleux, Y. Boufkhad, and O. Roussel. A translation of pseudo boolean constraints to sat. *Journal on Satisfiability, Boolean Modeling and Computation (JSAT)*, 2(1-4), 2006.

[6] A. Biere, M. J. H. Heule, H. van Maaren, and T. Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in AI and Applications*. IOS Press, 2009.

[7] E. Coquery, S. Jabbour, and L. Sais. A constraint programming approach for enumerating motifs in a sequence. In M. Spiliopoulou, H. Wang, D. J. Cook, J. Pei, W. Wang, O. R. Zaïane, and X. Wu, editors, *ICDM Workshops*, pages 1091–1097. IEEE, 2011.

[8] E. Coquery, S. Jabbour, L. Saïs, and Y. Salhi. A SAT-Based Approach for Discovering Frequent, Closed and Maximal Patterns in a Sequence. In *20th European Conference on Artificial Intelligence ECAI*, pages 258–263, 2012.

[9] M. Davis, G. Logemann, and D. W. Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[10] L. De Raedt, T. Guns, and S. Nijssen. Constraint Programming for Itemset Mining. In *Proceedings of the 14th International Conference on Knowledge Discovery and Data Mining (ACM SIGKDD'2008)*, pages 204–212, Las Vegas, Nevada, USA, August 24-27 2008.

[11] M. Gebser, B. Kaufmann, A. Neumann, and T. Schaub. *clasp* : A conflict-driven answer set solver. In *9th International Conference Logic Programming and Nonmonotonic Reasoning (LPNMR'2007)*, volume 4483 of *Lecture Notes in Computer Science*, pages 260–265. Springer, 2007.

[12] K. Gouda, M. Hassaan, and M. J. Zaki. Prism: A primal-encoding approach for frequent sequence mining. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM 2007)*, pages 487–492, Omaha, Nebraska, USA, October 28-31 2007. IEEE Computer Society.

[13] O. Grumberg, A. Schuster, and A. Yadgar. Memory efficient all-solutions sat solver and its application for reachability analysis. In *In Proceedings of the 5th International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, pages 275–289. Springer, 2004.

[14] T. Guns, S. Nijssen, and L. D. Raedt. Itemset mining: A constraint programming perspective. *Artificial Intelligence*, 175(12-13):1951–1983, 2011.

[15] Y. Hamadi, S. Jabbour, and L. Sais. Learning from conflicts in propositional satisfiability. *4OR*, 10(1):15–32, 2012.

[16] S. Jabbour, L. Sais, and Y. Salhi. A pigeon-hole based encoding of cardinality constraints. In *In proceedings of the 29th International Conference on Logic Programming (ICLP'2013)*, August 24-29 2013.

[17] J. P. Marques-Silva and K. A. Sakallah. GRASP - A New Search Algorithm for Satisfiability. In *Proceedings of IEEE/ACM CAD*, pages 220–227, 1996.

[18] L. Parida, I. Rigoutsos, A. Floratos, D. Platt, and Y. Gao. Pattern Discovery on Character Sets and Real-valued Data: Linear Bound on Irredundant Motifs and an Efficient Polynomial Time Algorithm. In *ACM-SIAM Symposium on Discrete Algorithms*, pages 297–308, 2000.

[19] L. Parida, I. Rigoutsos, and D. Platt. An output-sensitive flexible pattern discovery algorithm. In *In proceedings of the 12th Annual Symposium on Combinatorial Pattern Matching (CPM'2001)*, volume 2089 of *Lecture Notes in Computer Science*, pages 131–142. Springer, 2001.

[20] N. Pisanti, M. Crochemore, R. Grossi, and M.-F. Sagot. Bases of motifs for generating repeated patterns with wild cards. *IEEE/ACM Transactions on Computational Biology and Bioinformatics (TCBB'2005)*, 2(1):40–50, 2005.

[21] C. Sinz. Towards an Optimal CNF Encoding of Boolean Cardinality Constraints. In *11th International*

Conference on Principles and Practice of Constraint Programming - CP 2005, pages 827–831, 2005.

[22] G. Tseitin. On the complexity of derivations in the propositional calculus. In H. Slesenko, editor, *Structures in Constructives Mathematics and Mathematical Logic, Part II*, pages 115–125, 1968.

[23] J. P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Information Processing Letters*, 68(2):63–69, 1998.

[24] L. Zhang, C. F. Madigan, M. W. Moskewicz, and S. Malik. Efficient conflict driven learning in Boolean satisfiability solver. In *IEEE/ACM CAD'2001*, pages 279–285, 2001.

[25] W. Zhao and W. Wu. Asig: An all-solution sat solver for cnf formulas. In *11th International Conference on Computer-Aided Design and Computer Graphics, CAD/Graphics 2009, Huangshan, China, August 19-21 (CAD/Graphics)*, pages 508–513. IEEE, 2009.