ECAI 2012
Luc De Raedt et al. (Eds.)
© 2012 The Author(s).
This article is published online with Open Access by IOS Press and distributed under the terms of the Creative Commons Attribution Non-Commercial License.
doi:10.3233/978-1-61499-098-7-258

A SAT-Based Approach for Discovering Frequent, Closed and Maximal Patterns in a Sequence

Emmanuel Coquery[1] and **Said Jabbour**[2] and **Lakhdar Sais**[2] and **Yakoub Salhi**[2]

Abstract. In this paper we propose a satisfiability-based approach for enumerating all frequent, closed and maximal patterns with wildcards in a given sequence. In this context, since frequency is the most used criterion, we introduce a new polynomial inductive formulation of the cardinality constraint as a Boolean formula. A nogood-based formulation of the anti-monotonicity property is proposed and dynamically used for pruning. This declarative framework allows us to exploit the efficiency of modern SAT solvers and particularly their clause learning component. The experimental evaluation on real world data shows the feasibility of our proposed approach in practice.

1 Introduction

Frequent sequence data mining is the problem of discovering frequent patterns shared across time among an input data-sequence. Pattern discovery in data is widely used in bioinformatics as a way to extract meaningful "knowledge" in large volume of data such as protein motif discovery, gene prediction and sequence alignment. This problem is central to many other applications domains such as database and text mining.

In this paper we consider the pattern discovery problem for a specific class of patterns in a sequence. The data could be represented as a string (a sequence of "solid" characters), while the pattern can be seen as a subsequence with an additional special character called wildcard or joker that match any character [11, 14, 1]. We are particularly interested in enumerating all frequent, closed and maximal patterns in a sequence. The enumeration problem for maximal and closed motifs with wildcards has been investigated recently by several authors [12, 14, 1]. 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. [11], 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 [1] proposed a polynomial space and polynomial delay algorithm MaxMotif for maximal pattern discovery of the class of motifs with wildcards.

Our approach for enumerating all frequent patterns in a sequence differs from all the previous specialized approaches. It follows the constraint programming (CP) based data mining framework proposed recently by Luc De Raedt et al. in [15] 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 [7]. This first study leads to the first CP approach for itemset mining displaying nice declarative opportunities without neglecting efficiency. Encouraged by these promising results our first goal is to heavily exploit these declarative languages (constraint programming and Boolean satisfiability (SAT)) and their associated efficient and generic solving techniques. First, we propose a SAT model for our problem that includes different types of constraints. The first one, called support constraint, allows us to capture the locations of a given candidate pattern. The second one, called frequency constraint, encodes that a pattern must appears at least λ times in the sequence. Using inductive arguments, we also provide an interesting encoding of the frequency constraint to a Boolean formula. Finally, in order to search for closed and maximal motifs, we also provide a Boolean formulation of both closedness and maximality constraints. This leads us to the first SAT-based model for enumerating frequent, closed and maximal patterns in a sequence. In addition to the benefit that can be obtained thanks to the recent progress in satisfiability testing, another important motivation behind the Boolean model rises in the ability of modern SAT solvers to efficiently handle nogoods thanks to their clause learning component. In order to encode the anti-monotonicity property, we show that it can be dynamically maintained using a set of nogoods. Each time a pattern p is proved infrequent, all patterns p' such that $p \subseteq p'$ are eliminated thanks to additional nogoods.

2 Preliminaries

2.1 Frequent patterns mining: definitions and notations

In this section, we give a formal description of the problem of enumerating frequent patterns, possibly interspersed with a wildcard symbol, in a sequence. To this end we first introduce some basic definitions and notations according to [11, 14, 1]. Let Σ be an alphabet, which is a finite set of symbols (called solid characters). A sequence (or simply a string) s is a successive sequence of characters $s_1 \dots s_n \in \Sigma^n$. The length of the string s is denoted by $|s|$. We denote by $\mathcal{O} = \{1 \dots |s|\}$ the set of positions of the characters in s. A wildcard (or don't care) is an additional symbol $\circ$ not belonging to Σ ($\circ \notin \Sigma$) that matches any symbol of the alphabet including itself.

Definition 1 (Pattern) *A pattern over Σ is a sequence $p = p_1 \dots p_m \in \Sigma \cup \Sigma.(\Sigma \cup \{\circ\})^*.\Sigma$, where $\cup$ and $.$ are the classi-*

[1] Université de Lyon, F-69622 France. Université de Lyon 1, LIRIS UMR CNRS 5205, France. emmanuel.coquery@liris.cnrs.fr
[2] Université Lille Nord de France, F-59000 Lille, France. UArtois, CRIL UMR CNRS 8188, F-62300 Lens, France. {jabbour, sais, salhi}@cril.univ-artois.fr

cal regular expression operators of union and concatenation respectively. The first and the last position of a pattern must contain a solid character i.e. $p_1 \neq \circ$ *and* $p_m \neq \circ$.

Definition 2 (Inclusion) *A pattern* $p = p_1 \dots p_m$ *appears in a sequence* $s = s_1 \dots s_n$ *at the location* $l \in \mathcal{O}$ *denoted* $p \subseteq_l s$, *if* $\forall i \in \{1 \dots m\}$, $p_i = s_{l+i-1}$ *or* $p_i = \circ$. *We note* $\mathcal{L}_s(p) = \{l \in \mathcal{O} | p \subseteq_l s\}$ *the support of* p *in* s. *We say that* $p \subseteq s$ *iff* $\exists l \in \mathcal{O}$ *such that* $p \subseteq_l s$.

Definition 3 (Motif/Frequent pattern) *Let* s *be a sequence and* p *a pattern. Given a positive number* $\lambda \geq 1$, *called quorum, we say that* p *is a motif (or frequent pattern) in* s *when* $|\mathcal{L}_s(p)| \geq \lambda$. *The set of all motifs of* s *for the quorum* λ *is denoted by* $\mathcal{M}_s^\lambda$.

Example 1 *Let* $s = aababcabcba$ *be a sequence and* $p = ab \circ b$ *be a pattern. As* $p \subseteq_2 s$, *and* $p \subseteq_7 s$, *we have* $\mathcal{L}_s(p) = \{2, 7\}$. *If we set the quorum* λ *to the value* 2, *then the pattern* p *is a motif of* s.

Definition 4 (EMS) *The problem of Enumerating all Motifs in a Sequence (EMS) can be defined as follows. Given a sequence* s *and a quorum* $\lambda \geq 1$, *enumerates all the elements of* $\mathcal{M}_s^\lambda$.

The following property can play an important role in reducing the search space. Indeed, if a pattern is shown to be infrequent, all patterns containing it are also infrequent.

Proposition 1 (Anti-monotonicity) *Let* p_1 *and* p_2 *be two patterns such that* $p_1 \subseteq p_2$. *If* $|\mathcal{L}_s(p_2)| \geq \lambda$ *then* $|\mathcal{L}_s(p_1)| \geq \lambda$.

Definition 5 (Closed Frequent Pattern) *A frequent pattern* p *is closed in a sequence* s *if for any frequent pattern* q *such that* $q \supset p$, *there is no integer* d *such that* $\mathcal{L}_s(q) = \mathcal{L}_s(p)+d$, *where* $\mathcal{L}_s(p)+d = \{l + d | l \in \mathcal{L}_s(p)\}$.

Definition 6 (Maximal Frequent Pattern) *A frequent pattern* p *is maximal in a sequence* s *if for any frequent pattern* q, $q \not\supset p$.

One can easily see that all frequent patterns can be obtained from the closed frequent patterns and also from the maximal ones by replacing solid characters with wildcards. Since the number of the closed frequent patterns (resp. maximal frequent patterns) are smaller or equal to the number of the frequent patterns, enumerating all closed frequent patterns (resp. maximal frequent patterns) allows us to reduce the size of the output.

2.2 Boolean Satisfiability

Let $\mathcal{P}$ be a propositional language of formulas $\mathcal{F}_\mathcal{P}$ built in the standard way, using usual logical connectives ($\vee$, $\wedge$, $\neg$, $\Rightarrow$, $\Leftrightarrow$) and a set of propositional variables. Using linear Tseitin encoding, any Boolean formula $f \in \mathcal{F}_\mathcal{P}$ can be translated to $CNF(f)$, a formula in conjonctive normal form. A *CNF formula* Φ is a set (interpreted as a conjunction) of *clauses*, where a clause is a set (interpreted as a disjunction) of *literals*. A literal is a positive or negated propositional variable. A *unit* clause is a clause containing only one literal (called *unit literal*). An *empty clause*, noted $\bot$, is interpreted as false (unsatisfiable), whereas an *empty CNF formula*, noted $\top$, is interpreted as true (satisfiable). We define $\Phi|_x = \{c \mid c \in \Phi, \{x, \neg x\} \cap c = \emptyset\} \cup \{c \backslash \{\neg x\} \mid c \in \Phi, \neg x \in c\}$ as the formula obtained from Φ by assigning the value *true*to x. Φ^* denotes the formula Φ *closed under unit propagation*, defined recursively as follows: (1) $\Phi^* = \Phi$ if Φ does not contain any unit clause, (2) $\Phi^* = \bot$ if Φ contains two unit clauses $\{x\}$ and $\{\neg x\}$, (3) otherwise, $\Phi^* = (\Phi|_x)^*$ where x is the literal of a unit clause of Φ. An *assignment* $\mathcal{I}$ of Φ is a function which associates a value $\mathcal{I}(x) \in \{false, true\}$ to some of the variables of Φ. $\mathcal{I}$ is *complete* if it assigns a value to every variable of Φ, and *partial* otherwise. A *model* of a formula Φ is an assignment that makes the formula $true$. *The SAT problem consists in determining if a Boolean formula expressed in CNF admits a model or not.*

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 [6]. 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 [9, 19]. 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 can be found in [5].

3 A SAT-based Approach for EMS

In this section, we show how the EMS problem (Definition 4) can be encoded as a Boolean formula in CNF. In addition to the encoding of the enumeration of motifs (frequent pattern) in a sequence, we also provide a Boolean encoding of the closed and maximal motifs enumeration problems.

3.1 Motivations

Our Boolean model is motivated by several important features of both the EMS problem and SAT solvers. First, all the constraints defining the EMS problem are defined on boolean variables leading naturally to a SAT-based model (Boolean formula in CNF). Our second motivation comes from our desire to benefit from the (1) *impressive progress in Boolean Satisfiability checking* [5]. The scalability of modern SAT solvers can be related to both algorithmic improvements, and to (2) *their ability to exploit the hidden structures of the problem instance*. By structure, we understand the *dependencies between variables*, which can often appear through Boolean functions. One particular example being the well known notion of *strong backdoors* [18] that can be informally defined as subset of variables such that any assignments of these variables leads to a tractable subformula.

Another important motivation behind our Boolean model rises in the ability of modern SAT solvers to (3) *efficiently handle nogoods thanks to their clause learning component*. Indeed, clause learning play a critical role in the success of modern complete SAT solvers. The main idea is that when a current branch leads to a conflict, clause learning aims to derive a clause that succinctly express the causes of

conflict. Such learned clause is then used to prune the search space. Clause learning also known in the literature as Conflict Driven Clause Learning (CDCL) refers now to the most known and used First UIP learning scheme, first integrated in the SAT solver Grasp [9] and efficiently implemented in zChaff [10]. Most of the SAT solvers, usually called modern SAT solvers, integrate this strong learning scheme. Theoretically, by integrating clause learning to DPLL-like procedure [6], the obtained SAT solver corresponds to a proof system that is as powerful as general resolution [13].

3.2 Boolean Encoding

We recall that n and m represent the size of the sequence and the maximal size of motifs respectively. To enumerate all motifs of arbitrary length, we first need to set the length of the motif to the upper bound $m = n - \lambda + 1$. Secondly, we suppose that the sequence is completed on the right hand side with $m - 1$ wildcards. Let us take again the example 1. The maximum length of the motifs is set to $m = 9$. Indeed, as $\lambda = 3$ and the size of the sequence is $n = 11$, then the patterns of size greater than 9 are infrequent. The number of wildcards added at the end of the sequence is $m - 1 = 8$. This completion allows us to search for all motifs of different sizes. Indeed, the string a$\circ\circ\circ\circ\circ\circ\circ\circ$ of size 9 representing the pattern a of size 1 (a pattern starts and ends with a solid character) appears 5 times in s when s is completed with 8 wildcards; 2 times otherwise.

We now describe our Boolean encoding of the EMS problem. Suppose that $\Sigma = \{a_1 \dots a_{|\Sigma|}\}$. Let $p = p_1 p_2 \dots p_m$ be a candidate pattern, where for all $i \in \{1 \dots, m\}$, $p_i \in \Sigma \cup \{\circ\}$. An instantiation of the variables associated to a candidate pattern p to $a_1 \dots a_m$ represents a pattern $a_1 \dots a_l$ such that $a_l \neq \circ$ and for all $i \in \{l+1, \dots, m\}$, $a_i = \circ$, i.e. l is the last position of a solid character in $a_1 \dots a_m$. For example, an instantiation of $p_1 \dots p_6$ to a$\circ$b$\circ\circ\circ$ represents the pattern $a \circ b$.

For each p_i $(1 \leq i \leq m)$, we associate $|\Sigma| + 1$ boolean variables $\{p_i = a_1 \dots p_i = a_j \dots p_i = a_{|\Sigma|+1}\}$, where $p_i = a_j$, for $j \in \{1, \dots, |\Sigma|\}$, expresses that p_i takes the value a_j and $p_i = a_{|\Sigma|+1}$ expresses that p_i is a joker ($\circ$). The number of introduced boolean variables, denoted by $\mathcal{P}$, is $m \times (|\Sigma|+1)$. In the sequel, for simplicity reason, we note the literal $\neg(p_i = a_j)$ as $p_i \neq a_j$.

3.3 EMS as Boolean Formulae

Let us now introduce the Boolean formulation of the different constraints necessary for the encodings of the problems we consider.

Domain Constraint We first need to encode that each p_i, for $i \in \{1, \dots, m\}$, must take only one value in $\Sigma \cup \{\circ\}$. This domain constraint is reformulated by the following two boolean formulas that correspond to the *at least* and *at most* constraints respectively:

$$\bigwedge_{i=1}^{m} [\bigvee_{j=1}^{|\Sigma|+1} (p_i = a_j)] \tag{1}$$

$$\bigwedge_{i=1}^{m} [\bigvee_{1 \leq j < k \leq |\Sigma|+1} (p_i \neq a_j \vee p_i \neq a_k)] \tag{2}$$

As mentioned below, the first symbol of p must be a solid character. This is expressed by the following simple unary constraint (unit clause):

$$p_1 \neq \circ \tag{3}$$

Location Constraint We define the location constraint $loc(k, p, s)$ in order to express that p is located in s at position k.

$$(p_1 = s_k) \wedge [\bigwedge_{i=2}^{m} (p_i = \circ \vee p_i = s_{k+i-1})] \tag{4}$$

The location constraint is a CNF formula. Let us note that, since s is completed with wildcards, for all $i \in \{1, \dots, m\}$ and for all $k \in \{1, \dots, n\}$, s_{k+i-1} exists.

Support Constraint The support constraint $supp(p, s)$ allows us to capture the locations of p in s:

$$\bigwedge_{k=1}^{n} (b_k \Leftrightarrow loc(k, p, s)) \tag{5}$$

where $\mathcal{B} = \{b_1 \dots b_n\}$ is a set of new Boolean variables. In the previous formula, $b_k = true$ if the pattern p is located at the position k in s; $false$ otherwise.

Frequency Constraint To search only for frequent patterns (motifs), we introduce the frequency constraint $freq(p, s, \lambda)$ which express that p must occurs at least λ times in s.

$$\sum_{k=1}^{n} b_k \geq \lambda \tag{6}$$

The frequency constraint 6 is a boolean cardinality constraint (0/1 linear inequality). Several polynomial encodings of this kind of constraints into 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 [17]. Recently, efficient encoding of the cardinality constraint to CNF have been proposed and most of them try to improve the efficiency of constraint propagation (e.g; [3, 16]).

In this paper, we introduce a new polynomial inductive formulation of the cardinality constraint. We first define by simultaneous induction on $i \in \{1, \dots, n\}$ and $l \in \{1, \dots, \lambda\}$ the formula $M(i, l)$ as follows:

$$\begin{cases} i) & M(i,0) & = & \top \\ ii) & M(n-l+1,l) & = & (b_{n-l+1} \wedge \dots \wedge b_n) \\ iii) & M(i,l) & = & (b_i \wedge \mathcal{X}_{i+1}^{l-1}) \vee \mathcal{X}_{i+1}^{l} \end{cases}$$

where for all $i \in \{1, \dots, n\}$ and for all $l \in \{\lambda - i + 1, \dots, \lambda\}$ with $i \leq n-l+1$ and $l \geq 1$, $\mathcal{X}_i^l$ is a new Boolean variable. $M(i, l)$ allows us to express that there are at least l variables in $\{b_k | i \leq k \leq n\}$ instantiated to $true$. Thus, the following formula in the case of our model expresses that p must occurs at least λ times in s:

$$M(1, \lambda) \tag{7}$$

Moreover, for all $i \in \{1, \dots, n\}$ and for all $l \in \{\lambda - i + 1, \dots, \lambda\}$ with $i \leq n - l + 1$ and $l \geq 1$, we have to add the following Boolean formula:

$$\mathcal{X}_i^l \leftrightarrow M(i, l) \tag{8}$$

We denote by $\mathcal{X}$ the set of Boolean variables $\mathcal{X}_i^l$ where $i \in \{1, \dots, n\}$, and $l \in \{\lambda - i + 1, \dots, \lambda\}$, with $i \leq n - l + 1$ and $l \geq 1$. It is clear that the following proposition is satisfied:

Proposition 2 *The size of the set $\mathcal{X}$ is bounded by $n \times \lambda$.*

Proposition 3 *The number of clauses of the inductive formulation of the cardinality constraint $\sum_{k=1}^{n} b_k \geq \lambda$ is in $\mathcal{O}(n \times \lambda)$.*

Proof: Since the number of basic cases of the form (ii) is bounded by λ, the number of their corresponding clauses is bounded by

$\lambda \times (\lambda + 1)$. The number of Boolean formulas of the form (iii) is bounded by $n \times \lambda$. Since each occurrence corresponds to 6 clauses, the total number of their associated clauses is bounded by $6 \times n \times \lambda$ (see also (8)). As $\lambda \leq n$, the whole number of clauses encoding the cardinality constraint is in $\mathcal{O}(n \times \lambda)$. □

Theorem 1 *The problem of enumerating all motifs in a given sequence s is expressed by the constraints (1), (2), (3), (5), (7) and (8).*

Proof: A direct consequence of our encoding. □

If we take a closer look to the EMS SAT model, we can see that the set of boolean variables representing the candidate pattern p is clearly a strong backdoor set. Indeed, when such variables are assigned, the values of the other Boolean variables $\mathcal{B} \cup \mathcal{X}$ are trivially deduced. The constraints (5), (7) and (8) express such dependencies (the value of the variable b_k depends on the value of $loc(k, p, s)$, and the value of $\mathcal{X}_i^l$ depends on the values of the variables b_k). Consequently, one can enumerate only on $\mathcal{P}$. Hence, the complexity is exponentially bounded by the size of $\mathcal{P}$. The Strong backdoor set is an interesting structural information that we will provide to the SAT solver.

Closedness Constraint Let us now introduce the closedness constraint:

$$\bigwedge_{k=1}^{n} (\neg b_k \vee s_{k+i-1} = a) \rightarrow p_i = a \ for \ 1 \leq i \leq m, \ a \in \Sigma \quad (9)$$

$$(\bigwedge_{i=m-j}^{m} p_i = \circ) \rightarrow \Phi_c(j, a) \quad for \quad 0 \leq j \leq m-2, \ a \in \Sigma \quad (10)$$

where $\Phi_c(j, a) = \neg(\bigwedge_{k=1}^{n}(b_k \rightarrow s_{k-j-1} = a))$.

Theorem 2 *The problem of enumerating all closed motifs in a given sequence s is expressed by the constraints (1), (2), (3), (5), (7), (8), (9) and (10).*

Proof: The key idea in the closedness constraint consists in replacing the maximum number of wildcards in a pattern with solid characters w.r.t its support and forbiding the pattern to be included at location j in another one with positions $\mathcal{L}_s(p) - j$. □

Maximality Constraint We introduce here the maximality constraint. In a sense, it is similar to the closedness constraint:

$$\sum_{k=1}^{n} b_k \wedge (s_{k+i-1} = a) \geq \lambda \rightarrow p_i = a \quad for \quad 1 \leq i \leq m, \ a \in \Sigma \quad (11)$$

$$(\bigwedge_{i=m}^{m-j} p_i = \circ) \rightarrow \Phi_{max}(j, a) \quad for \quad 0 \leq j \leq m-2, \ a \in \Sigma \quad (12)$$

where $\Phi_{max}(j, a) = \sum_{k=1}^{n} b_k \wedge (s_{k-j-1} = a) \leq \lambda - 1$. The expressions of the form $\sum_{k=1}^{n} x_k \triangleleft \lambda$, with $\triangleleft \in \{\leq, \geq\}$, are encoded inductively as the frequency constraint.

Theorem 3 *The problem of enumerating all maximal motifs in a given sequence s is expressed by the constraints (1), (2), (3), (5), (7), (8), (11) and (12).*

Proof: In the maximality constraint, the idea consists in replacing the maximum number of wildcards in a pattern with solid characters w.r.t λ. The constraint (12) expresses that no frequent pattern includes p at location j. □

3.4 Anti-monotonicity: a nogood-based approach

To exploit the anti-monotonicity property, one need to prove that a given pattern p is infrequent. Consequently, each time a pattern is proven infrequent, we dynamically add new constraints called nogoods to the constraints database.

We assume that $\mathcal{I}$ is a Boolean interpretation such that he frequency constraint is violated and $p' = \{p'_1, \ldots, p'_k\}$ corresponds to the infrequent pattern extracted from $\mathcal{I}$ using the set of variables $\mathcal{P}$. Let $i_1, \ldots i_l$ be the ordered sequence of positions in p' such that $\forall 1 \leq j \leq l, p'_{i_j} \neq \circ$. The following nogoods are added dynamically to the constraints database in order to avoid future patterns p such that $p' \subseteq p$.

$$antiMon(p', p) = \bigwedge_{x=1}^{m-i_l+1} \bigvee_{y=1}^{l} (p'_{i_y} \neq p_{i_y+x-1}) \quad (13)$$

These nogoods state for each position x in p s.t. $p' \not\sqsubseteq_x p$, at least one of the solid characters in p' is not matched in p at the expected position $i_y + x - 1$.

It is important to note that the exploitation of the anti-monotonicity property in our approach requires the use of the strong backdoor set (pattern variables). By branching on these pattern variables, the extraction of the infrequent pattern can be done in a simple way as the pattern variables are first assigned in the current branch. In our encoding using inductive formulation for the cardinality constraint, all the conflicts are always encountered because of the frequency constraint. However, this is not always the case in other encodings of the cardinality constraint [17, 3, 16]. In the case of such encodings, one has to use an additional procedure to determine if a conflict is encountered because of the frequency constraint. For instance, a such procedure can consist in analyzing and checking if all the conflict set of decisions (or literals involved in the conflict) are in the strong backdoor set $\mathcal{P}$.

4 Implementation and preliminary experiments

4.1 Implementation details

In this section we describe our SAT based solver for EMS (Algorithm 1). It is based on a model of CDCL SAT algorithm proposed in [13]. Algorithm 1 (`SatEms+AM`) includes all the basic components of modern SAT solvers and integrates the anti-monotonicity property (AM) with strong backdoors ($\mathcal{P}$). It takes as input a CNF formula Φ and a set $\mathcal{P}$ of pattern variables and returns the set $\mathcal{M}$ of motifs. The algorithm is based on variable assignments called *decisions* ($\mathcal{D}$) followed by unit propagation. `SatEms+AM` starts with the following four empty sets (lines 1-4): decision literals ($\mathcal{D}$), motifs ($\mathcal{M}$), learnt clauses database (Δ) and the anti-monotone nogoods database (Γ). Then, it iterates until finding all the motifs. In each iteration, the conjunction of Φ, $\mathcal{D}$, Δ, and Γ are checked for inconsistency using unit propagation (line 7). If unit propagation finds an inconsistency ($\mathcal{S}^*$==$\bot$, line 7), the algorithm does one of the two things:
(1) The decision sequence $\mathcal{D}$ is empty, the algorithm terminates by returning the set of all motifs $\mathcal{M}$ (line 8).
(2) The decision sequence $\mathcal{D}$ is not empty, a clause α is generated by classical conflict analysis (line 9) and added to the learnt clauses database Δ (line 10). As a conflict occurs, an infrequent pattern p is then generated (line 11) and the set of associated anti-monotone nogoods Θ are built (line 12). From the nogood representing the infrequent pattern p we apply the classical conflict analysis and generate an additional learnt clause β (line 13). This last learnt clause together with the set of nogoods Θ are added to the anti-monotone nogoods database Γ (line 14). Then a level n is computed based on α and β by

Algorithm 1: SatEms+AM

Input: a CNF formula Φ, and a set $\mathcal{P}$ of pattern variables
Output: a set of all motifs $\mathcal{M}$

1 $\mathcal{D} \leftarrow \emptyset$; /* Decision literals */
2 $\mathcal{M} \leftarrow \emptyset$; /* Set of all motifs */
3 $\Delta \leftarrow \emptyset$; /* Learnt clauses */
4 $\Gamma \leftarrow \emptyset$; /* Anti monotone nogoods */
5 **while** (true) **do**
6 $\mathcal{S} \leftarrow (\Phi \wedge \mathcal{D} \wedge \Delta \wedge \Gamma)$;
7 **if** $(\mathcal{S}^* = \bot)$ **then**
8 **if** $(\mathcal{D} = \emptyset)$ **then return** $\mathcal{M}$;
9 $\alpha \leftarrow analyzeConflict(\mathcal{S}, \mathcal{D})$;
10 $\Delta \leftarrow \Delta \cup \{\alpha\}$;
11 $p \leftarrow extractPattern(\mathcal{D}, \mathcal{P})$;
12 $\Theta \leftarrow antiMonotone(p)$;
13 $\beta \leftarrow analyzeConflict(p, \mathcal{S}, \mathcal{D})$;
14 $\Gamma \leftarrow \Gamma \cup \Theta \cup \{\beta\}$;
15 $n \leftarrow min\{level(\alpha), level(\beta)\}$;
16 $\mathcal{D} \leftarrow \mathcal{D}_n$; /* n first decisions */
17 **else**
18 **if** $((l \leftarrow$ decide$(\mathcal{P}))=$ null) **then** $p \leftarrow extractPattern(\mathcal{D}, \mathcal{P})$;
19 $\mathcal{M} \leftarrow \mathcal{M} \cup \{p\}$;
20 $\sigma \leftarrow extractNogood(p)$;
21 $\Phi \leftarrow \Phi \cup \{\sigma\}$;
22 $\mathcal{D} \leftarrow \emptyset$;
23 **else**
24 $\mathcal{D} \leftarrow \mathcal{D} \cup \{l\}$;

taking the minimum of their assertion levels (line 15). The algorithm then erases all decisions made after level n (line 16), and moves on to the next iteration.

If unit propagation detects no inconsistency, the solver makes a decision by selecting a literal l from the backdoor set (line 18), and adds it to the decision sequence (line 24). If no such literal is found, a motif p is then extracted from the set of decisions and added to the set of motifs $\mathcal{M}$ (line 19). To avoid enumerating the same models several times, in line 20, a nogood σ is generated from the found motif and added to the original formula Φ (line 21). Search restarts by setting the set of decision to an empty set (line 22).

Some of the basic SAT functions of Algorithm 1 such us $analyzeConflict()$ and $level()$ are informally described in Section 2.2. We will now provide some missing details of the other specific functions :

- $extractPattern(\mathcal{D}, \mathcal{P})$: given a set of decisions $\mathcal{D}$ and a set of pattern variables $\mathcal{P}$, a pattern p is extracted from $\mathcal{D}$.
- $antiMonotone(p)$: generates a set of anti-monotone nogoods (Equation 13).
- $extractNogood(p)$: let $p = d \circ c$ be a motif of s, the extracted nogood is $(p_1 \neq d \vee p_2 \neq \circ \vee p_3 \neq c)$.

4.2 Experiments

In this section, an experimental evaluation of our approach on some real world data are given. We consider sequences from two application domains:
Bioinformatics: proteinic data encoded as a sequence of amino-acid of arbitrary length[3].
Computer security: user data drawn from the command histories of UNIX computer users[4] [8].

These experiments aim (1) to show the feasibility of our approach, and (2) to analyze the power and weakness of our implementation. We compare two versions of Algorithm 1 based on MiniSAT 2.2 [5]:

1. SatEms+AM: a full version of the Algorithm 1. It integrates the anti-monotonicity property together with strong backdoors set $\mathcal{P}$.
2. SatEms: a basic version of Algorithm 1 without the anti-monotonicity property. SatEms can be obtained from Algorithm 1 by deleting the lines 11-14, and substituting line 15 with $n \leftarrow level(\alpha)$.

Both SatEms+AM and SatEms enumerate the set of frequent patterns (motifs), respectively with and without using the anti-monotonicity property.

We conducted two kind of experiments. In the first one, we illustrate the evolution of computation time while varying the length of the input sequence. The different sequences are build from the *User data* by taking the first k characters, where k is varied from 200 to 6000 by a step of 200. As in [1], to get approximately the same number of motifs, for each sequence i (dot in the figure), we use a quorum proportional to its length ($\lambda_i = \frac{length_i}{50}$). The results obtained on the problem of enumerating frequent patterns in a sequence by SatEms and SatEms+AM using two different encodings of the cardinality constraints (BDD [4] encoding using BoolVar/PB open source java library [6] and our inductive formulation (IND) presented in Section 3.3) are depicted in Figure 1. The results clearly show that applying the anti-monotone property with additional domain knowledge leads to dramatic improvements. In the second experiment, we consider proteinic data of fixed length and we measure the evolution of computation time with respect to the quorum. The quorum evolves linearly ($\lambda_0 = 10$ and $\lambda_i = \lambda_{i-1} + 10$). To analyze the behavior of our approach with respect to the number of motifs, we show in Figure 2 and Figure 3 the evolution of the ratio between CPU time and the number of motifs while varying the quorum. The considered sequence is of length 1200 characters. In Figure 2, we show the results obtained by SatEms and SatEms+AM using again two different encodings of the cardinality constraints (BDD encoding [4] and our inductive encoding presented in Section 3.3). Clearly, the anti-monotonicity property improves significantly the performances of SatEms when the frequency constraint is encoded using our inductive formulation. However, no real improvement is observed when we use BDD encoding with anti-monotonicity property. For both encodings, the ratio time/#motifs do not evolves significantly. The performances seem to be less sensitive with respect to the number of motifs, except in the hard region around $\lambda = 90$. Overall, the BDD encoding of the frequency constraint is better than the inductive formulation.

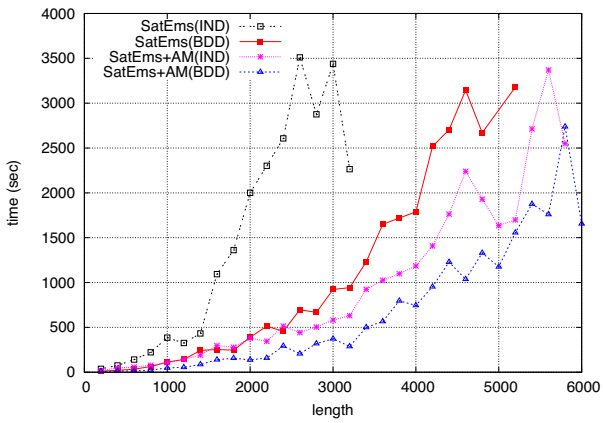

Figure 1. User data : time Vs length (Frequent)

In the Figure 3, we show the performance of SatEms obtained on the generation of closed (*closed motifs*) and maximal (*maximal*

[3] http://www.biomedcentral.com/1471-2105/11/175/additional/
[4] http://kdd.ics.uci.edu/databases/UNIX_user_data/
[5] MiniSAT: http://minisat.se/
[6] BoolVAR/PB : http://boolvar.sourceforge.net/

motifs) patterns in a sequence using our inductive formulation of the cardinality constraint. We note similar observation on the sensitivity of the methods in terms of the number of motifs. The performances of our approach for enumerating closed motifs is clearly better than that of enumerating maximal ones.

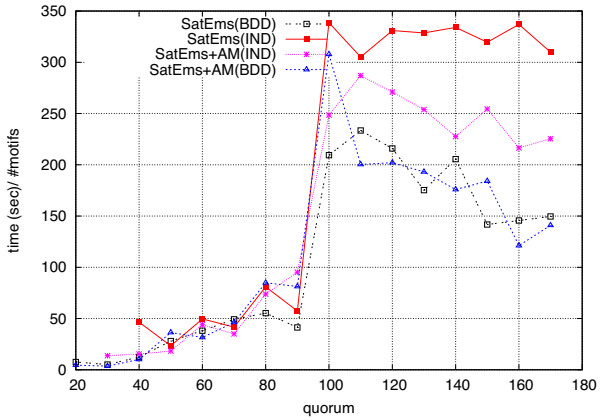

Figure 2. Bioinfo: time/#motifs Vs quorum (Frequent)

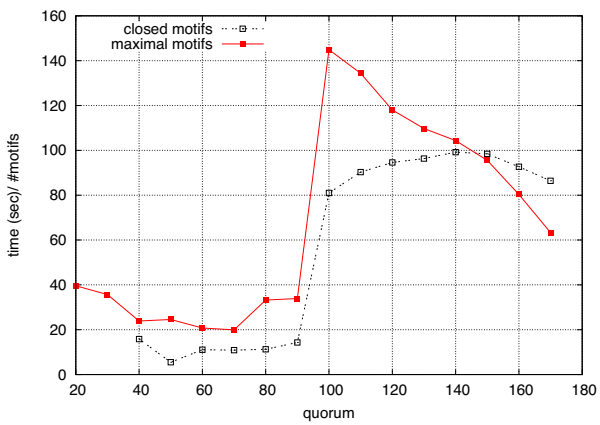

Figure 3. Bioinfo: time/#motifs Vs quorum (Closed and Maximal)

The experiments above show the feasibility of our proposed framework on some data sequences. In order to compare our approach to the state-of-the-art, we consider the imperative one proposed in [1]. As far as we now, it is considered as the most effective specialized approach for enumerating motifs in a sequence. The proposed algorithm is both polynomial space and polynomial delay. The key of the algorithm is depth search on tree shaped search route over all maximal motifs based on a technique called prefix-preserving closure extension. Let us note that Arimura and Uno approach is extremly efficient as it can handle huge sequences of over to 10 million length.

We run Arimura and Uno algorithm [7] on the data sequences presented above (bioinfo and user data). It is, as expected, significantly more efficient than our SAT based formulation. Indeed, most of the instances are solved in less than 5 seconds. This is the consequence, among other things, of the fact that the SAT-solver used in our approach is not suitable for model enumeration. Indeed, to find a model different from the previous outcomes, the SAT-solver injects simply the negations of the latter models. In fact, this issue is less studied in the SAT literature and we think that the design of effective SAT solvers for enumerating all models is an important perspective for both constraint based datamining and other application domains. Let us note that some additional gains can be obtained using more efficient encodings of the cardinality constraint (e.g. Cardinality networks [2]). Having said that, our approach provides a declarative and flexible framework that can be easily extended with other constraints. For instance, constraints for restrictions on the number of consecutive wildcards. The SAT based datamining approaches can be much more effective in the case where the interestingness predicate is non-montonic.

[7] http://research.nii.ac.jp/ uno/code/maxmotif.html

5 Conclusion and future works

In this paper, we proposed a first Satisfiability based approach for enumerating frequent, closed and maximal motifs with wildcards in a sequence. This declarative approach offers an additional possibility to benefit from the recent progress in satisfiability testing and particularly from their efficient nogood (or clause learning) component.

This work opens several issues for future research. First, the study of how our encodings can be extended with constraints on the form of the enumerated patterns (restriction on the number of consecutive wildcards, regular expressions, etc) will be the next step to be developed in further works. Moreover, we plan to investigate how this framework can be extended to deal with other data mining problems such in sequences of itemsets, trees, graphs, etc.

REFERENCES

[1] 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**, 243–262, (2007).
[2] R. Asín, R. Nieuwenhuis, A. Oliveras, and E. Rodríguez-Carbonell, 'Cardinality networks: a theoretical and empirical study', Constraints, **16**(2), 195–221, (2011).
[3] O. Bailleux and Y. Boufkhad, 'Efficient cnf encoding of boolean cardinality constraints', in CP 2003, pp. 108–122, (2003).
[4] O. Bailleux, Y. Boufkhad, and O. Roussel, 'A translation of pseudo boolean constraints to sat', JSAT, **2**(1-4), (2006).
[5] Handbook of Satisfiability, eds., Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, volume 185 of Frontiers in AI and Applications, IOS Press, 2009.
[6] M. Davis, G. Logemann, and D. W. Loveland, 'A machine program for theorem-proving.', Comm. of the ACM, **5**(7), 394–397, (1962).
[7] T. Guns, S. Nijssen, and L. De Raedt, 'Itemset mining: A constraint programming perspective', Artif. Intell., **175**(12-13), 1951–1983, (2011).
[8] T. Lane, 'Filtering techniques for rapid user classification', in AAAI-98/ICML-98 Joint Workshop on AI Approaches to Time-series Analysis, pp. 58–63, (1998).
[9] J. P. Marques-Silva and K. A. Sakallah, 'GRASP - A New Search Algorithm for Satisfiability', in Proceedings of IEEE/ACM CAD, pp. 220–227, (1996).
[10] M. W. Moskewicz, C. F. Madigan, Y. Zhao, L. Zhang, and S. Malik, 'Chaff: Engineering an efficient SAT solver.', in Proceedings of the 38th Design Automation Conference, pp. 530–535, (2001).
[11] 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, pp. 297–308, (2000).
[12] L. Parida, I. Rigoutsos, and D. Platt, 'An output-sensitive flexible pattern discovery algorithm', in CPM'2001, pp. 131–142, (2001).
[13] K. Pipatsrisawat and A. Darwiche, 'On the power of clause-learning sat solvers with restarts', in CP'09, pp. 654–668, (2009).
[14] N. Pisanti, M. Crochemore, R. Grossi, and M. Sagot, 'Bases of motifs for generating repeated patterns with wild cards', IEEE/ACM TCBB'2003, **2**, 2005, (2003).
[15] L. De Raedt, T. Guns, and S. Nijssen, 'Constraint programming for itemset mining', in ACM SIGKDD, pp. 204–212, (2008).
[16] C. Sinz, 'Towards an optimal cnf encoding of boolean cardinality constraints', in CP'05, pp. 827–831, (2005).
[17] J. P. Warners, 'A linear-time transformation of linear inequalities into conjunctive normal form', Information Processing Letters, (1996).
[18] R. Williams, C. P. Gomes, and B. Selman, 'Backdoors to typical case complexity.', in IJCAI'2003, pp. 1173–1178, (2003).
[19] L. Zhang, C. F. Madigan, M. W. Moskewicz, and S. Malik, 'Efficient conflict driven learning in Boolean satisfiability solver', in IEEE/ACM CAD'2001, pp. 279–285, (2001).