Artificial Intelligence 244 (2017) 30–47

Contents lists available at ScienceDirect

Artificial Intelligence

www.elsevier.com/locate/artint

Mining Top-*k* motifs with a SAT-based framework

Said Jabbour *, Lakhdar Sais *, Yakoub Salhi *

CRIL – CNRS, Université d'Artois, F-62307 Lens Cedex, France

ARTICLE INFO

Article history:
Received in revised form 15 October 2015
Accepted 27 November 2015
Available online 30 November 2015

Keywords:
Boolean satisfiability
Data mining
Modeling
Top-*k* motifs

ABSTRACT

In this paper, we introduce a new problem, called Top-*k* SAT, that consists in enumerating the Top-*k* models of a propositional formula. A Top-*k* model is defined as a model with less than *k* models preferred to it with respect to a preference relation. We show that Top-*k* SAT generalizes two well-known problems: the Partial MAX-SAT problem and the problem of computing minimal models. Moreover, we propose a general algorithm for Top-*k* SAT. Then, we give an application of our declarative framework in data mining, namely, the problem of mining Top-*k* motifs in the transaction databases and in the sequences. In the case of mining sequence data, we introduce a new mining task by considering the sequences of itemsets. Thanks to the flexibility and to the declarative aspects of our SAT-based approach, an encoding of this task is obtained by a very slight modification of mining motifs in the sequences of items.

1. Introduction

The problem of mining frequent itemsets is well-known and essential in data mining, knowledge discovery and data analysis. It has applications in various fields and becomes fundamental for data analysis as datasets and datastores are becoming very large. Since the first article of Agrawal [1] on association rules and itemset mining, the huge number of works, challenges, datasets and projects show the actual interest in this problem (see [2] for a recent survey of works addressing this problem).

We are also interested in frequent sequence data mining which 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 work, 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 [3–5]. 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 [3] "*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*".

Important progress has been achieved for data mining and knowledge discovery in terms of implementations, platforms, libraries, etc. As pointed out in [2], several works deal with designing highly scalable data mining algorithms for large scale datasets. An important problem of data mining problems, in general, concerns the huge size of the output, from which it is difficult for the user to retrieve relevant information. Consequently, for practical data mining, it is important to reduce the size of the output by exploiting the structure of the patterns. Computing for example, closed, maximal, condensed, discriminative patterns are some of the well-known and useful techniques. Most of the works on itemset and sequential

* Corresponding authors.
E-mail addresses: jabbour@cril.fr (S. Jabbour), sais@cril.fr (L. Sais), salhi@cril.fr (Y. Salhi).

http://dx.doi.org/10.1016/j.artint.2015.11.003
0004-3702/

mining require the specification of a minimum support threshold λ. This constraint allows the user to control at least to some extent the size of the output by mining only patterns covering at least λ transactions (locations). However, in practice, it is difficult for users to provide an appropriate threshold. As pointed out in [6], a too small threshold may lead to the generation of a huge number of patterns, whereas a too high value of the threshold may result in no answer. In [6], based on a complete ranking between itemsets, the authors propose to mine the n most interesting itemsets of arbitrary length. In [7], the proposed task consists in mining Top-k frequent closed itemsets of length greater than a given lower bound min, where k is the desired number of frequent closed itemsets to be mined, and min is the minimal length of each itemset. The authors show that setting the minimal length of the itemsets to be mined is much easier than setting the usual frequency threshold. Since the introduction of Top-k mining, several research works investigated its use in graph mining (e.g. [8,9]) and other datamining tasks (e.g. [10,11]). This new framework can be seen as a nice way to mine the k preferred patterns according to some specific constraints or measures. Starting from this observation, our goal in this paper is to define a general logic based framework for enumerating the Top-k preferred patterns according to a predefined preference relation.

The notion of preference has a central role in several disciplines such as economics, operations research and decision theory in general. Preferences are relevant for the design of intelligent systems that support decisions. Modeling and reasoning with preferences play an increasing role in Artificial Intelligence (AI) and its related fields such as nonmonotonic reasoning, planning, diagnosis, configuration, constraint programming (CP) and other areas in knowledge representation and reasoning (KR). For example, in nonmonotonic reasoning the introduction of preferential semantics by Shoham [12] gives a unifying framework where nonmonotonic logic is reduced to a standard logic with a preference relation (order) on the models of that standard logic. Several models for representing and reasoning about preferences have been proposed. For example, soft constraints [13] are one of the most general way to deal with quantitative preferences, while CP-net (Conditional Preferences networks) [14] is most convenient for qualitative preferences. There is a huge literature on preferences (see [15–17] for a survey at least from the AI perspective). In data mining, preferences have also been investigated by several authors (e.g. [18–20]). For example, in [18], the authors introduced a new paradigm of pattern discovery based on Soft Constraints, where constraints are no longer rigid boolean functions. More recently, Ugarte et al. [20] introduced a generic and efficient method based on constraint programming for mining (soft-)skypatterns that enable to express user preference according to dominance relations. From the observation that dominance relations can be found in many pattern mining settings, in [21], the authors propose an algebra that combines constraints and dominance relations that can be used to describe a broad range of pattern mining settings.

In this paper we focus on qualitative preferences defined by a preference relation on the models of a propositional formula. Preferences in propositional satisfiability (SAT) have not received a lot of attention. In [22], a new approach for solving satisfiability problems in the presence of qualitative preferences on literals (defined as partial ordered set) is proposed. The authors particularly show how DPLL procedure can be easily adapted for computing optimal models induced by the partial order. The issue of computing optimal models using DPLL has also been investigated in SAT [23].

Recently, a constraint programming (CP) based data mining (DM) framework was proposed by Luc De Raedt et al. in [24] for itemset mining (CP4IM). 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 [25]. This study leads to the first CP approach for itemset mining displaying nice declarative opportunities. Encouraged by these promising results, several contributions addressed other data mining problems using the two well-known CP and SAT AI formalisms. For example, in [26], the authors proposed a SAT-based encoding for the problem of discovering frequent, closed and maximal patterns in a sequence of items and a sequence of itemsets. In [27], the authors solve the frequent itemset mining problem by compiling the set of all itemset into a binary decision diagram (BDD) (augmented with counts). Frequent itemset are then extracted by querying the BDD. By considering the relationship between local constraint-based mining and constraint satisfaction problems, Khiari et al. [28] proposed a model for mining patterns combining several local constraints, i.e., patterns defined by n-ary constraints. Also, several constraint-based language for modeling and solving data mining problems have been designed. Let us mention the constraint-based language defined in [29], which enables the user to define queries in a declarative way addressing pattern sets and global patterns. All primitive constraints of the language are modeled and solved using the SAT framework. More recently, Guns at al. [30], introduced a general-purpose declarative mining framework, called MiningZinc. Compared to CP4IM framework [31], MiningZinc supports a wide variety of different solvers (including DM algorithms and general purpose solvers) and uses a significantly more high-level modeling language.

This new research trend offers nice opportunities for cross-fertilization between AI and data mining. The work presented in this paper fit into this framework. Our goal is to provide a step forward towards the integration of AI and Data mining. Our approach is based on SAT as the underlined data mining constraints involve Boolean variables.

The contributions of this paper are the following:

1. Firstly, we propose a generic framework for dealing with qualitative preferences in propositional satisfiability. Our qualitative preferences are defined using a reflexive and transitive relation (preorder) over the models of a propositional formula. Such preference relation on models is first used to introduce a new problem, called Top-k SAT, defined as the problem of enumerating the Top-k models of a propositional formula. Here a Top-k model is defined as a model with no more than k-1 models preferred to it with respect to the considered preference relation. Then, we show that Top-k

SAT generalizes the two well-known problems, the Partial MAX-SAT problem and the problem of generating minimal models. We also define a particular type of preference relations that allows us to introduce a general algorithm for solving the Top-k SAT problem.

2. Secondly, we introduce our first application of our declarative framework to data mining. More precisely, we consider the problem of mining Top-k frequent closed itemsets of minimum length *min* [32]. In this problem, the minimum support threshold usually used in frequent itemset mining is not known, while the minimum length can be set to 0 if one is interested in itemsets of arbitrary length. In itemset mining, the notion of Top-k itemset was introduced in [7] as an alternative to finding the appropriate value for the minimum support threshold. It is also an elegant way to control the size of the output. Consequently, itemset mining is clearly a nice application of our new defined Top-k SAT problem. In this context, we provide a SAT encoding and we show that computing the Top-k closed itemsets of length at least *min* corresponds to computing the Top-k models of the obtained propositional formula.
3. Thirdly, we provide an application of our declarative framework to sequence mining. Indeed, we propose an encoding of the problem of enumerating the Top-k patterns with wildcards in a sequence of items in our framework. The notion of Top-k pattern in this context is defined in the same way as in itemset mining. The SAT encoding that we use comes from our encoding proposed in [26].
4. In the fourth contribution, we consider a variant of the problem of discovering patterns with wildcards in a sequence, proposed in our work in [26], 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 mining problem introduced in [33]. 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. In this case, an encoding of the problem of enumerating the Top-k patterns in our framework is derived from the one used for the sequences of items with a very slight modification demonstrating its flexibility.

In this paper, we extend our work published in [34]. The extension includes the application of our declarative framework to sequence mining (contributions 3 and 4) using our SAT encodings introduced in [26]. We also provide an extensive experimental results showing the feasibility of our proposed approach.

2. Preliminary definitions and notations

In this section, we describe the Boolean satisfiability problem (SAT), the itemset mining problem and introduce some necessary notations.

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. A *CNF formula* (Conjunctive Normal Form) 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 [35]. We denote by $Var(\Phi)$ the set of propositional variables occuring in Φ.

An *interpretation* $\mathcal{M}$ of a propositional formula Φ is a function which associates a value $\mathcal{M}(p) \in \{0, 1\}$ (0 for *false* and 1 for *true*) to each propositional variable p in $Var(\Phi) \subseteq V$. A *model* of a formula Φ is an interpretation $\mathcal{M}$ that satisfies the formula. The *SAT problem* consists in deciding whether a given CNF formula admits a model.

We denote by $\bar{l}$ the complementary literal of l. More precisely, if $l = p$ then $\bar{l}$ is $\neg p$ and if $l = \neg p$ then $\bar{l}$ is p. For a set of literals L, $\bar{L}$ is defined as $\{\bar{l} \mid l \in L\}$. Moreover, we denote by $\overline{\mathcal{M}}$ ($\mathcal{M}$ is an interpretation over $Var(\Phi)$) the clause $\bigvee_{p \in Var(\Phi)} s(p)$, where $s(p) = p$ if $\mathcal{M}(p) = 0$, $\neg p$ otherwise. Let Φ be a CNF formula and $\mathcal{M}$ an interpretation over $Var(\Phi)$. We denote by $\mathcal{M}(\Phi)$ the set of clauses satisfied by $\mathcal{M}$. Let us now consider a set X of propositional variables such that $X \subseteq Var(\Phi)$. We denote by $\mathcal{M} \cap X$ the set of variables $\{p \in X | \mathcal{M}(p) = 1\}$. Moreover, we denote by $\mathcal{M}_{|X}$ the restriction of the model $\mathcal{M}$ to X.

Let us now introduce the itemset mining problem.

Transaction database. Let $\mathcal{I}$ be a set of *items*. A *transaction* is a couple (tid, I) where *tid* is the *transaction identifier* and I is an *itemset*, i.e., $I \subseteq \mathcal{I}$. A *transaction database* is a finite set of transactions over $\mathcal{I}$ where each transaction identifier refers to only one itemset.

Cover, support and frequency of an itemset. We say that a transaction (tid, I) *supports* an itemset J if $J \subseteq I$. The *cover* of an itemset I in a transaction database $\mathcal{D}$ is the set of transaction identifiers in $\mathcal{D}$ supporting I: $\mathcal{C}(I, \mathcal{D}) = \{tid \mid (tid, J) \in \mathcal{D}, I \subseteq J\}$. The *support* of an itemset I in $\mathcal{D}$ is defined by: $\mathcal{S}(I, \mathcal{D}) = \mid \mathcal{C}(I, \mathcal{D}) \mid$. Moreover, the frequency of I in $\mathcal{D}$ is defined by: $\mathcal{F}(I, \mathcal{D}) = \frac{\mathcal{S}(I, \mathcal{D})}{|\mathcal{D}|}$.

For instance, consider the following transaction database:

tid	itemset
1	a, b, c, d
2	a, b, e, f
3	a, b, c, m
4	a, c, d, f, j
5	j, l
6	d
7	d, j

Transaction database $\mathcal{D}$

In this database, we have $\mathcal{S}(\{a, b, c\}, \mathcal{D}) = |\{1, 3\}| = 2$ and $\mathcal{F}(\{a, b\}, \mathcal{D}) = \frac{3}{7}$.

Let $\mathcal{D}$ be a transaction database over $\mathcal{I}$ and λ a minimum support threshold. The *frequent itemset mining problem* consists in computing the following set:

$$\mathcal{FIM}(\mathcal{D}, \lambda) = \{I \subseteq \mathcal{I} \mid \mathcal{S}(I, \mathcal{D}) \geq \lambda\}$$

Let us now define one of the well known condensed representation of frequent itemsets.

Definition 1 *(Closed itemset).* Let $\mathcal{D}$ be a transaction database (over $\mathcal{I}$) and I an itemset ($I \subseteq \mathcal{I}$) such that $\mathcal{S}(I, \mathcal{D}) \geq 1$. I is closed if, for all itemset J with $I \subset J$, $\mathcal{S}(J, \mathcal{D}) < \mathcal{S}(I, \mathcal{D})$.

One can easily see that all frequent itemsets can be obtained from the closed frequent itemsets by computing their subsets. Since the number of closed frequent itemsets is smaller than or equal to the number of frequent itemsets, enumerating all closed itemsets allows us to reduce the size of output without losing information.

3. Preferences and Top-*k* models

Let Φ be a propositional formula and Λ_Φ the set of all its models. A preference relation $\succeq$ over Λ_Φ is a reflexive and transitive binary relation (a preorder). The statement $\mathcal{M} \succeq \mathcal{M}'$ means that $\mathcal{M}$ is at least as preferred as $\mathcal{M}'$. We denote by $P(\Phi, \mathcal{M}, \succeq)$ the subset of Λ_Φ defined as follows:

$$P(\Phi, \mathcal{M}, \succeq) = \{\mathcal{M}' \in \Lambda_\Phi \mid \mathcal{M}' \succ \mathcal{M}\}$$

where $\mathcal{M}' \succ \mathcal{M}$ means that $\mathcal{M}' \succeq \mathcal{M}$ holds but $\mathcal{M} \succeq \mathcal{M}'$ does not. It corresponds to all models that are strictly preferred to $\mathcal{M}$.

We now introduce an equivalence relation $\approx_X$ over $P(\Phi, \mathcal{M}, \succeq)$, where X is a set of propositional variables. It is defined as follows:

$$\mathcal{M}' \approx_X \mathcal{M}'' \text{ iff } \mathcal{M}' \cap X = \mathcal{M}'' \cap X$$

Thus, the set $P(\Phi, \mathcal{M}, \succeq)$ can be partitioned into a set of equivalence classes by $\approx_X$, denoted by $[P(\Phi, \mathcal{M}, \succeq)]^X$. Each equivalence class is a set of equivalent models with respect to X that are strictly preferred to $\mathcal{M}$. Then, $|[P(\Phi, \mathcal{M}, \succeq)]^X|$ represents the number of equivalent classes in the set of strictly preferred models to $\mathcal{M}$ with respect to X. In our context, this equivalence relation is used to take into consideration only a subset of propositional variables. For instance, we introduce new variables in Tseitin's translation [35] of propositional formula to CNF, and such variables are not important in the case of some preference relations.

Definition 2 *(Top-k model).* Let Φ be a propositional formula, $\mathcal{M}$ a model of Φ, $\succeq$ a preference relation over the models of Φ and X a set of propositional variables. $\mathcal{M}$ is a Top-k model w.r.t. $\succeq$ and X iff $|[P(\Phi, \mathcal{M}, \succeq)]^X| \leq k - 1$.

Let us note that the number of the Top-k models of a formula is not necessarily equal to k. Indeed, it can be strictly greater or smaller than k. For instance, if a formula is unsatisfiable, then it does not have a Top-k model for any $k \geq 1$. Nevertheless, if the considered preference relation is a complete order, then the number of Top-k models is always smaller than or equal to k.

It is easy to see that we have the following *monotonicity property*: if $\mathcal{M}$ is a Top-k model and $\mathcal{M}' \succeq \mathcal{M}$, then $\mathcal{M}'$ is also a Top-k model.

Top-*k* SAT problem. Let Φ be propositional formula, $\succeq$ a preference relation over the models of Φ, X a set of propositional variables and k a strictly positive integer. We call the tuple $(\Phi, \succeq, X, k)$ a *Top-k instance*. The Top-k SAT problem consists in computing a set $\mathcal{L}$ of Top-k models of Φ with respect to $\succeq$ and X satisfying the two following properties:

1. for all Top-k model $\mathcal{M}$, there exists $\mathcal{M}' \in \mathcal{L}$ such that $\mathcal{M} \approx_X \mathcal{M}'$; and
2. for all $\mathcal{M}$ and $\mathcal{M}'$ in $\mathcal{L}$, if $\mathcal{M} \neq \mathcal{M}'$ then $\mathcal{M} \not\approx_X \mathcal{M}'$.

The two previous properties come from the fact that we are only interested in the truth values of the variables in X. Indeed, the first property means that, for all Top-k model, there is a model in $\mathcal{L}$ equivalent to it with respect to $\approx_X$. Moreover, the second property means that $\mathcal{L}$ does not contain two equivalent Top-k models.

In the following definition, we introduce a particular type of preference relations, called δ-preference relations, that allows us to introduce a general algorithm for computing Top-k models.

Definition 3. Let Φ be a CNF formula and $\succeq$ a preference relation on the models of Φ. Then $\succeq$ is a δ-preference relation, if there exists a function $\delta^{\phi}_{\succeq}$ from Λ_Φ to the set of CNF formulae, called *bound function*, such that (i) it is polynomially computable in the size of Φ and (ii) for all $\mathcal{M} \in \Lambda_\Phi$, $\mathcal{M}'$ is a model of $\Phi \wedge \delta^{\phi}_{\succeq}(\mathcal{M})$ iff $\mathcal{M}'$ is a model of Φ and $\mathcal{M} \not\succ \mathcal{M}'$, for every Boolean interpretation $\mathcal{M}'$.

Let us note that, given a model $\mathcal{M}$ of a CNF formula Φ, $\delta^{\Phi}_{\succeq}(\mathcal{M})$ is a CNF formula so that when added (with conjunction) to Φ together with $\overline{\mathcal{M}}$, the models of the resulting formula are different from $\mathcal{M}$ and they correspond to all the models of Φ which are not less preferred than $\mathcal{M}$. Intuitively, this can be seen as a way to introduce a lower bound during the enumeration process. In other words, at each step of the enumeration process, a δ-preference relation allows us to generate a formula from the current found model to force the algorithm to enumerate the models that are not less preferred.

Clearly, if we ignore the condition (i), each preference relation is a δ-preference relation. Indeed, for every preference relation $\succeq$ on Λ_Φ, we only have to define the bound function $\delta^{\phi}_{\succeq}$ as follows:

$$\delta^{\phi}_{\succeq}(\mathcal{M}) = \bigwedge_{\mathcal{M}' \in \Lambda_\Phi, \mathcal{M} \not\succ \mathcal{M}'} \overline{\mathcal{M}'}$$

This definition means that we exclude all the models of Φ that are less preferred than $\mathcal{M}$. One can easily see that in this case $\delta^{\phi}_{\succeq}(\mathcal{M})$ is not polynomially computable in the worst case.

From now, we only consider the δ-preference relations. We also consider that their bound functions are provided.

3.1. Top-k SAT and Partial MAX-SAT

In this section, we show that the Top-k SAT problem generalizes the Partial MAX-SAT problem (e.g. [36]). In Partial MAX-SAT each clause is either relaxable (soft) or non-relaxable (hard). The objective is to find an interpretation that satisfies all the hard clauses together with the maximum number of soft clauses. The MAX-SAT problem is a particular case of Partial MAX-SAT where all the clauses are relaxable.

Let $\Phi = \Phi_h \wedge \Phi_s$ be a Partial MAX-SAT instance such that Φ_h is the hard part and Φ_s the soft part. The relation denoted by $\succeq_{\Phi_s}$ corresponds to a preference relation defined as follows: for all $\mathcal{M}$ and $\mathcal{M}'$ models of Φ_h defined over $Var(\Phi_h \wedge \Phi_s)$, $\mathcal{M} \succeq_{\Phi_s} \mathcal{M}'$ if and only if the number of soft clauses satisfied by $\mathcal{M}$ is greater than or equal to the number of soft clauses satisfied by $\mathcal{M}'$, i.e., $|\mathcal{M}(\Phi_s)| \geq |\mathcal{M}'(\Phi_s)|$. Clearly, the set of models of Φ_h over $Var(\Phi_h \wedge \Phi_s)$ is isomorphic to that of $\Phi' = \Phi_h \wedge \bigwedge_{C \in \Phi_s} p_C \leftrightarrow C$, where p_C for $C \in \Phi_s$ are fresh propositional variables. Indeed, for every model $\mathcal{M}$ of Φ', $\mathcal{M}_{|Var(\Phi)}$ is a model of Φ_h. Further, for every model $\mathcal{M}$ of Φ_h, the following Boolean interpretation $\mathcal{M}'$ is a model of Φ':

$$\mathcal{M}'(p) = \begin{cases} \mathcal{M}(p) & \text{if } p \in Var(\Phi) \\ 1 & \text{if } p \equiv p_C \text{ and } \mathcal{M}(C) = 1 \\ 0 & \text{otherwise} \end{cases}$$

We define the preference relation $\succeq$ over the models of Φ' as follows: $\mathcal{M} \succeq \mathcal{M}'$ if and only if $\mathcal{M}_{|Var(\Phi)} \succeq_{\Phi_s} \mathcal{M}'_{|Var(\Phi)}$. It is a δ-preference relation since its bound function $\delta^{\Phi'}_{\succeq}$ can be defined as follows:

$$\delta^{\Phi'}_{\succeq}(\mathcal{M}) = \sum_{C \in \Phi_s} p_C \geq |\mathcal{M}(\Phi_s)|$$

If S is the set of all the Top-1 models of Φ' with respect to $\succeq$ and $Var(\Phi)$, then the $S' = \{\mathcal{M}_{|Var(\Phi)} \mid \mathcal{M} \in S\}$ correspond to the set of all solutions of Φ in Partial MAX-SAT. Naturally, the models in S' are the most preferred models with respect to $\succeq_{\Phi_s}$, and that means they satisfy Φ_h and satisfy the maximum number of clauses in Φ_s. Thus, in a sense, the Top-k SAT problem can be seen as a generalization of Partial MAX-SAT.

The formula $\delta^{\Phi'}_{\succeq}(\mathcal{M})$ involves the well-known cardinality constraint (0/1 linear inequality). Several polynomial encodings of this kind of constraints into CNF formulas have been proposed in the literature. The first linear encoding of general linear inequalities to CNF has been proposed by Warners [37]. 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. [38–41]).

Algorithm 1: Top-k.

Input: a Top-k instance $(\Phi, \succeq, X, k)$
Output: A solution $\mathcal{L}$ of $(\Phi, \succeq, X, k)$

1 $\Phi' \leftarrow \Phi$;
2 $\mathcal{L} \leftarrow \emptyset$; /* Set of all Top-$k$ models */
3 **while** $(\mathcal{M} = \texttt{solve}(\Phi'))$ **do** /* $\mathcal{M}$ is a model of Φ' */
4 | **if** $(\exists \mathcal{M}' \in \mathcal{L}$ *s.t.* $\mathcal{M} \approx_X \mathcal{M}'$ & $\mathcal{M} \succ \mathcal{M}')$ **then**
5 | | $\texttt{replace}(\mathcal{M}, \mathcal{M}', \mathcal{L})$;
6 | **else if** $(\forall \mathcal{M}' \in \mathcal{L}$ *s.t.* $\mathcal{M} \not\approx_X \mathcal{M}'$ & $|\texttt{preferred}(\mathcal{M}, \mathcal{L})| < k)$ **then**
7 | | $S \leftarrow \texttt{min\_top}(k, \mathcal{L})$;
8 | | $\mathcal{L} \leftarrow \mathcal{L} \cup \mathcal{M}$;
9 | | $\texttt{removeNotTop-k}(k, \mathcal{L})$;
10 | | $S \leftarrow \texttt{min\_top}(k, \mathcal{L}) \setminus S$;
11 | | $\Phi' \leftarrow \Phi' \wedge \bigwedge_{\mathcal{M}' \in S} \delta^{\Phi}_{\succeq}(\mathcal{M}')$;
12 | **else**
13 | | $\Phi' \leftarrow \Phi' \wedge \delta^{\Phi}_{\succeq}(\mathcal{M})$
14 | $\Phi' \leftarrow \Phi' \wedge \overline{\mathcal{M}}$;
15 **return** $\mathcal{L}$;

3.2. Top-k SAT and X-minimal model generation problem

Let Φ be a formula, $\mathcal{M}$ and $\mathcal{M}'$ two models of Φ and X a set of propositional variables. Then, $\mathcal{M}$ is said to be smaller than $\mathcal{M}'$ with respect to X, written $\mathcal{M} \leq_X \mathcal{M}'$, if $\mathcal{M} \cap X \subseteq \mathcal{M}' \cap X$. We now define a preference relation $\succeq_X$ as follows: $\mathcal{M} \succeq_X \mathcal{M}'$ if and only if $\mathcal{M} \leq_X \mathcal{M}'$ i.e. $\mathcal{M}$ is at least as preferred as $\mathcal{M}'$.

We now show that $\succeq_X$ is a δ-preference relation. We can define $\delta^{\Phi}_{\succeq_X}$ as follows:

$$\delta^{\Phi}_{\succeq_X}(\mathcal{M}) = (\bigwedge_{p \in \mathcal{M} \cap X} p) \rightarrow \bigwedge_{p' \in X \setminus \mathcal{M} \cap X} \overline{p'}$$

Indeed, $\mathcal{M}'$ is a model of a formula $\Phi \wedge \overline{\mathcal{M}} \wedge \delta^{\Phi}_{\succeq_X}(\mathcal{M})$ if and only if $\mathcal{M}'$ is a model of Φ, $\mathcal{M}' \neq \mathcal{M}$, and either $(\mathcal{M} \cap X) \setminus (\mathcal{M}' \cap X) \neq \emptyset$ or $\mathcal{M}' \cap X \subseteq \mathcal{M} \cap X$. The two previous statements mean that $\mathcal{M} \not\succeq_X \mathcal{M}'$. In fact, if $\mathcal{M}'$ satisfies $(\bigwedge_{p \in \mathcal{M} \cap X} p)$ then $\mathcal{M}'$ must satisfy $\bigwedge_{p' \in X \setminus \mathcal{M} \cap X} \overline{p'}$. As a consequence, we deduce that $\mathcal{M}' \cap X \subseteq \mathcal{M} \cap X$. Otherwise, $\mathcal{M}'$ falsify $(\bigwedge_{p \in \mathcal{M} \cap X} p)$ and that means that $(\mathcal{M} \cap X) \setminus (\mathcal{M}' \cap X) \neq \emptyset$. This latter statement expresses that either $\mathcal{M}' \cap X \subset \mathcal{M} \cap X$ or $\mathcal{M}$ and $\mathcal{M}'$ are incomparable with respect to $\succeq_X$.

Let Φ be a propositional formula, X a set of propositional variables and $\mathcal{M}$ a model of Φ. Then $\mathcal{M}$ is said to be an *X-minimal model* of Φ if there is no model strictly smaller than $\mathcal{M}$ with respect to $\succeq_X$. In [42], it was shown that finding an X-minimal model is $P^{NP[O(log(n))]}$-hard, where n is the number of propositional variables.

The set of all X-minimal models corresponds to the set of all Top-1 models with respect to $\succeq_X$ and $Var(\Phi)$. Indeed, if $\mathcal{M}$ is a Top-1 model, then there is no model $\mathcal{M}'$ such that $\mathcal{M}' \succ_X \mathcal{M}$, and that means that $\mathcal{M}$ is an X-minimal model. In this context, let us note that computing the set of Top-k models for $k \geq 1$ can be seen as a generalization of X-minimal model generation problem.

3.3. An algorithm for Top-k SAT

In this section, we describe our algorithm for computing Top-k models in the case of the δ-preference relations (Algorithm 1). The basic idea is simply to use the formula $\delta^{\Phi}_{\succeq}(\mathcal{M})$ associated to a model $\mathcal{M}$ to obtain models that are not less preferred than $\mathcal{M}$. This algorithm takes as input a CNF formula Φ, a preference relation $\succeq$, a strictly positive integer k, and a set X of propositional variables allowing to define the equivalence relation $\approx_X$. It has as output a set $\mathcal{L}$ of Top-k models of Φ satisfying the two properties given in the definition of the Top-k SAT problem.

3.3.1. Algorithm description

In the while-loop, we use lower bounds for finding optimal models. These lower bounds are obtained by using the fact that the preorder relation considered is a δ-preference relation. In each step, the lower bound is integrated by using the formula:

$$\bigwedge_{\mathcal{M}' \in S} \delta^{\Phi}_{\succeq}(\mathcal{M}')$$

- **Lines 3.** Let us first mention that the function $solve(\phi')$ refer to any SAT solver or DPLL procedure that enumerates all the models of a given CNF formula.
- **Lines 4–5.** The procedure $\texttt{replace}(\mathcal{M}, \mathcal{M}', \mathcal{L})$ replaces $\mathcal{M}'$ with $\mathcal{M}$ in $\mathcal{L}$. We apply this replacement because there exists a model $\mathcal{M}'$ in $\mathcal{L}$ which is equivalent to $\mathcal{M}'$ and $\mathcal{M}$ allows to have a better bound.

- **Lines 6–11.** In the case where $\mathcal{M}$ is not equivalent to any model in $\mathcal{L}$ and the number of models in $\mathcal{L}$ preferred to it is strictly less than k ($|\texttt{preferred}(\mathcal{M}, \mathcal{L})| < k$), we add $\mathcal{M}$ to $\mathcal{L}$ (line 8). Note that $\mathcal{S}$ contains first the models of $\mathcal{L}$ before adding $\mathcal{M}$ that have exactly $k-1$ models preferred to them in this set. After adding $\mathcal{M}$ to $\mathcal{L}$, we remove from $\mathcal{L}$ the models that are not Top-k, i.e., they have more than $k-1$ models in $\mathcal{L}$ that are strictly preferred to them ($\texttt{removeNotTop-k}(k, \mathcal{L})$). Next, we modify the content of S. Note that the elements of S before adding $\mathcal{M}$ are used as bounds in the previous step. Hence, in order to avoid adding the same bound several times, the new content of S corresponds to the models in $\mathcal{L}$ that have exactly $k-1$ models preferred to them in $\mathcal{L}$ ($\texttt{min\_top}(k, \mathcal{L})$) deprived of the elements of the previous content of S. In line 11, we integrate lower bounds in Φ' by using the elements of S. Indeed, for all model $\mathcal{M}$ of a formula $\Phi' \wedge \bigwedge_{\mathcal{M}' \in S} \delta^{\Phi}_{\succeq}(\mathcal{M}')$, $\mathcal{M}' \not\succ \mathcal{M}$ holds, for any $\mathcal{M}' \in S$.
- **Lines 12–13.** In the case where $\mathcal{M}$ is not a Top-k model, we integrate its associated lower bound.
- **Line 14.** This instruction enables us to avoid finding the same model in two different steps of the while-loop.

Proposition 1. *Algorithm 1 (Top-k) is correct.*

Proof. The proof of the partial correctness is based on the definition of the δ-preference relation. Indeed, the function $\delta^{\Phi}_{\succeq}$ allows us to exploit bounds to systematically improve the preference level of the models. Indeed, using the condition $(\bar{ii})$ of the δ-preference relation, adding $\bigwedge_{\mathcal{M}' \in S} \delta^{\Phi}_{\succeq}(\mathcal{M}')$ at Line 11 and $\delta^{\Phi}_{\succeq}(\mathcal{M})$ at Line 13 ensure that we consider in the next iterations only the models that are not less preferred than $k-1$ found models. Further, as the number of models is bounded, adding the negation of the found model at each iteration leads to an unsatisfiable formula. Consequently the algorithm terminates. □

3.4. Complete preference relations

Our aim in this section is to show how Top-k SAT problem can benefit from algorithms of standard optimization problems in the case of complete preference relations. We provide in particular a simple algorithm that allows to find a Top-1 solution from a single Top-1 model, which may be found using an algorithm of an optimization problem.

Let us recall that a preference relation $\succeq$ is complete if, for all models $\mathcal{M}$ and $\mathcal{M}'$, we have $\mathcal{M} \succeq \mathcal{M}'$ or $\mathcal{M}' \succeq \mathcal{M}$. It is worth noting that the size of a solution of a Top-k SAT instance may also be greater than k in the case of complete preference relations, since solutions might be equivalent w.r.t. the equivalence relation induced by the preference relation. However, we get the following interesting property:

Proposition 2. *Let Φ be a propositional formula, $\succeq$ a complete preference relation, X a set of propositional variables and $\mathcal{M}$ a Top-1 model of Φ w.r.t. $\succeq$ and X. Then, for all $\mathcal{M}' \in \Lambda_{\Phi}$, $\mathcal{M}'$ is a Top-1 model of Φ w.r.t. $\succeq$ and X iff $\mathcal{M}' \approx \mathcal{M}$, where $\approx$ is the equivalence relation induced by $\succeq$.*

Proof. The "if part" is a consequence of the fact that $\mathcal{M}' \succeq \mathcal{M}$ and $\mathcal{M}$ is a Top-1 model. We now consider the "only if part". Let $\mathcal{M}'$ be a Top-1 model. Knowing that $\mathcal{M}$ is a Top-1 model and $\succeq$ is a complete preference relation, we get $\mathcal{M} \succeq \mathcal{M}'$. We also get $\mathcal{M}' \succeq \mathcal{M}$ since $\mathcal{M}$ is a Top-1 model. As a consequence, $\mathcal{M} \approx \mathcal{M}'$ holds. □

Note that the property highlighted in Proposition 2 is not true for any preference relation. Indeed, there exist preference relations in Top-k instances for which two distinct Top-1 models are incomparable.

In a sense, Proposition 2 describes a simple way to compute a solution of a Top-1 instance from a single Top-1 model in the case of complete preference relations. Indeed, such computation process is described in Algorithm 2, where we use an approach widely used for solving model enumeration problems. Usually, the algorithms designed to solve this problem are based on the use of additional clauses, called blocking clauses, to avoid producing repeated models [43]. Similarly, in Algorithm 2, we add to the Top-1 instance a blocking clause $\overline{\mathcal{M}_{|X}}$ (line 1) to avoid models equivalent to $\mathcal{M}$ w.r.t. the equivalence relation $\approx_X$, and a formula $\delta^{\Phi}_{\succeq}(\mathcal{M})$ (line 1) to only consider the models that are at least as preferred as the input Top-1 model $\mathcal{M}$, i.e. the models that are equivalent to $\mathcal{M}$ w.r.t. the equivalence relation induced by the preference relation. In this way, at each iteration of the while loop, the next found model $\mathcal{M}'$ (line 3) is a Top-1 model. At each iteration, we also make use of a blocking clause $\overline{\mathcal{M}'_{|X}}$ (line 5) to avoid equivalent models w.r.t. the equivalence relation $\approx_X$.

As a consequence, for some complete preference relations, methods for computing a solution for a Top-1 instance can be obtained by combining Algorithm 2 with algorithms for standard optimization problems, such as Partial MAX-SAT (see Section 3.1). In this approach, the optimization algorithm is used to compute a single Top-1 model, and then Algorithm 2 provides a Top-1 solution using this first Top-1 model. Moreover, a generalization of this approach to Top-k instances can be obtained by using a recursive algorithm. Indeed, consider a Top-k instance $(\Phi, \succeq, X, k)$ with a complete preference relation. We first compute a solution S of the Top-1 instance $(\Phi, \succeq, X, 1)$ using the approach that combines an optimization algorithm with Algorithm 2. Then, if $k - |S| \geq 1$ and $S \neq \emptyset$, a recursive call is made on the instance $(\Phi \wedge \bigwedge_{\mathcal{M} \in S} \overline{\mathcal{M}_{|X}}, \succeq, X, k - |S|)$ to compute a solution S' ($S \cup S'$ is a Top-k solution of $(\Phi, \succeq, X, k)$), otherwise S is returned.

The correctness of the recursive algorithm sketched above is a consequence of the following proposition:

Algorithm 2: Top-1^C.

Input: Top-1 instance $(\Phi, \succeq, X, 1)$ s.t. $\succeq$ is a complete preference relation and $\mathcal{M}$ a Top-1 model
Output: Solution $\mathcal{L}$ of $(\Phi, \succeq, X, 1)$

$\Phi' \leftarrow \Phi \wedge \overline{\mathcal{M}_{|X}} \wedge \delta_{\succeq}^{\Phi}(\mathcal{M})$;
$\mathcal{L} \leftarrow \{\mathcal{M}\}$;
while $(\mathcal{M}' = solve(\Phi'))$ **do** /* $\mathcal{M}'$ is a model of Φ' */
  $\mathcal{L} \leftarrow \mathcal{L} \cup \{\mathcal{M}'\}$;
  $\Phi' \leftarrow \Phi' \wedge \overline{\mathcal{M}'_{|X}}$;
return $\mathcal{L}$;

Proposition 3. *Let $I_1 = (\Phi, \succeq, X, k)$ be a Top-k SAT instance and S a Top-1 solution of $(\Phi, \succeq, X, 1)$. If $k - |S| \geq 1$ and $S \neq \emptyset$ then $S \uplus S'$ is a Top-k solution of I_1, where S' is a Top-$(k - |S|)$ solution of the instance $I_2 = (\Phi \wedge \bigwedge_{\mathcal{M} \in S} \overline{\mathcal{M}_{|X}}, \succeq, X, k - |S|)$ and $\uplus$ is the disjoint union operator; otherwise, S is a Top-k solution of I_1.*

Proof. It is worth noting that if $S = \emptyset$, then we know that Φ is unsatisfiable and there is no Top-k model. Moreover, if $k - |S| < 1$, then the Top-1 solution S contains at least k Top-1 models and it means that each Top-k model is a Top-1 model. Assume that $k - |S| \geq 1$ and $S \neq \emptyset$. Using the sub-formula $\bigwedge_{\mathcal{M} \in S} \overline{\mathcal{M}_{|X}}$, we know that (i) there is no model of $\Psi \equiv \Phi \wedge \bigwedge_{\mathcal{M} \in S} \overline{\mathcal{M}_{|X}}$ which is a Top-1 model of Φ w.r.t. $\succeq$ and X. We also know that (ii) each Top-k model which is not a Top-1 model of Φ w.r.t. $\succeq$ and X is a model of Ψ, since we only exclude in Ψ the Top-1 models of Φ. Let $\mathcal{M}$ be a Top-$(k - |S|)$ model of Ψ w.r.t. $\succeq$ and X. Then, $|[P(\Phi \wedge \bigwedge_{\mathcal{M} \in S} \overline{\mathcal{M}_{|X}}, \mathcal{M}, \succeq)]^X| \leq k - |S| - 1$ holds. Using Property (ii), we get $|[P(\Phi, \mathcal{M}, \succeq)]^X| \leq k - 1$ and, as a consequence, $\mathcal{M}$ is a Top-k model of Φ w.r.t. $\succeq$ and X. To complete our proof, we have to show that each Top-k model of Φ w.r.t. $\succeq$ and X, which is not a Top-1 model, is a Top-$(k - |S|)$ model of Ψ w.r.t. $\succeq$ and X. Assume that there exists a Top-k model $\mathcal{M}$ of Φ which is not a Top-1 model of Φ and not a Top-$(k - |S|)$ model of Ψ w.r.t. $\succeq$ and X. Using Property (i) and the fact that $\mathcal{M}$ is not a Top-$(k - |S|)$ model of Ψ, we get $|[P(\Phi \wedge \bigwedge_{\mathcal{M} \in S} \overline{\mathcal{M}_{|X}}, \mathcal{M}, \succeq)]^X| > k - |S| - 1$ and, as a consequence, $|[P(\Phi, \mathcal{M}, \succeq)]^X| > k - 1$ holds. We thus get a contradiction. □

Let us consider, for instance, a generalization of Partial MAX-SAT problem, called Top-k Partial MAX-SAT. It consists in computing a Top-k solution for an instance $\mathcal{T} = (\Phi_h, \succeq_{\Phi_s}, Var(\Phi_h \wedge \Phi_s), k)$ where $\Phi_h \wedge \Phi_s$ is a Partial MAX-SAT instance (Φ_h and Φ_s are the hard part and the soft part respectively) and $\succeq_{\Phi_s}$ is the preference relation defined in Section 3.1. An algorithm for computing a Top-1 solution can be defined as a combination of an algorithm for Partial MAX-SAT and Algorithm 2 as described above. Indeed, an algorithm for Partial MAX-SAT is first used to find a Top-1 model, and then, Algorithm 2 is used to find a Top-1 solution. Using this approach, we can also derive an algorithm for Top-k Partial MAX-SAT using our recursive approach.

In the sequel, we provide our SAT based encodings of itemset and sequence mining problems. We use standard propositional formula instead of CNF formula. Let us recall that every propositional formula can be translated in linear time to a CNF form using the well known extension principle (with fresh propositional variables) [35].

4. An application of Top-*k* SAT in itemset mining

The problem of mining frequent itemsets is well-known and essential in data mining [1], knowledge discovery and data analysis. Note that several data mining tasks are closely related to the itemset mining problem such as the ones of association rule mining, frequent pattern mining in sequence data, data clustering, etc. Recently, De Raedt et al. in [24,31] proposed the first constraint programming (CP) based data mining framework for itemset mining. This new framework offers a declarative and flexible representation model. It allows data mining problems to benefit from several generic and efficient CP solving techniques. This study leads to the first CP approach for itemset mining displaying nice declarative opportunities.

In itemset mining problem, the notion of Top-k frequent itemsets is introduced as an alternative to finding the appropriate value for the minimum support threshold. In this section, we propose a SAT-based encoding for enumerating all closed itemsets. Then we use this encoding in the Top-k SAT problem for computing all Top-k frequent closed itemsets.

In this work, we mainly consider the problem of mining Top-k frequent closed itemsets of minimum length *min*. In this problem, we consider that the minimum support threshold λ is not known.

Definition 4 *($\mathcal{FCI}^k_{min}$).* Let k and *min* be strictly positive integers. The problem of mining Top-k frequent closed itemsets $\mathcal{FCI}^k_{min}$ consists in computing all closed itemsets of length at least *min* such that, for each one, there exist no more than $k - 1$ closed itemsets of length at least *min* with supports greater than its support.

4.1. SAT-based encoding for $\mathcal{FCI}^k_{min}$

We here propose an encoding of $\mathcal{FCI}^k_{min}$ in Top-k SAT problem. Let $\mathcal{I}$ be a set of items, $\mathcal{D} = \{(0, t_i), \ldots, (n-1, t_{n-1})\}$ a transaction database over $\mathcal{I}$, and k and *min* strictly positive integers. In order to define our encoding, we associate to

each item a in $\mathcal{I}$ a propositional variable p_a. These propositional variables encode the candidate itemset $I \subseteq \mathcal{I}$, i.e., p_a is *true* iff $a \in I$. Moreover, for all $i \in \{0, \ldots, n-1\}$, we associate to the i-th transaction in $\mathcal{D}$ a propositional variable b_i. These propositional variables allow us to reason about the cover of the candidate itemset.

We first propose a constraint allowing to consider only the itemsets of length at least *min*. It corresponds to a cardinality constraint:

$$\sum_{a \in \mathcal{I}} p_a \geq min \tag{1}$$

We now introduce a constraint allowing to capture all the transactions where the candidate itemset does not appear:

$$\bigwedge_{i=0}^{n-1} (b_i \leftrightarrow \bigvee_{a \in \mathcal{I} \setminus t_i} p_a) \tag{2}$$

This constraint means that b_i is true if and only if the candidate itemset is not in t_i.

By the following constraint, we force the candidate itemset to be closed:

$$\bigwedge_{a \in \mathcal{I}} (\bigwedge_{i=0}^{n-1} \overline{b_i} \rightarrow a \in t_i) \rightarrow p_a \tag{3}$$

This formula means that if $\mathcal{C}(I) = \mathcal{C}(I \cup \{a\})$ where I is the candidate itemset, then $a \in I$ holds. In other words, when the transactions containing the candidate itemset (b_i is *false*) also contain the item a ($a \in t_i$), the candidate itemset can be extended by adding the item a (p_a is *true*).

Proposition 4. *The set of models of (1) ∧ (2) ∧ (3) corresponds to the set of closed itemsets of size at least min.*

Proof. Part ⇒. Let $\mathcal{M}$ be a model of (1) ∧ (2) ∧ (3) and I its corresponding itemset, i.e., $I = \{a \in \mathcal{I} \mid p_a \text{ is true}\}$. Then, using (1), we know that the size of I is greater than or equal to *min*. Moreover, using (2) ∧ (3), we know that, for all $a \in \mathcal{I}$, if $\mathcal{C}(I) = \mathcal{C}(I \cup \{a\})$ then $a \in I$ holds. Thus, we obtain that I is a closed itemset.

Part ⇐. Let I be a closed itemset of size greater than or equal to *min*. Then, we define its associated Boolean interpretation $\mathcal{M}$ as follows:

- for all $a \in \mathcal{I}$, $\mathcal{M}(p_a) = 1$ if and only if $a \in I$; and
- for all $i \in \{0, \ldots, n-1\}$, $\mathcal{M}(b_i) = 0$ if and only if t_i supports I.

Clearly, $\mathcal{M}$ satisfies the constraint (1), since the size of I is greater or equal to *min*. Moreover, we know that the set of all the transactions supporting I is $\{t_i \in \mathcal{D} \mid \mathcal{M}(b_i) = 0\}$. Hence, $\mathcal{M}$ satisfies the constraint (2). Using the fact that I is a closed itemset, we have, for all $a \in \mathcal{I}$, if $\mathcal{C}(I) = \mathcal{C}(I \cup \{a\})$ then $a \in I$ holds. Therefore, $\mathcal{M}$ satisfies the constraint (3). □

In this context, computing the Top-k closed itemsets of length at least *min* corresponds to computing the Top-k models of $\Phi \equiv (1) \wedge (2) \wedge (3)$ with respect to $\succeq_B$ and $X = \{p_a | a \in \mathcal{I}\}$, where $B = \{b_0, \ldots, b_{n-1}\}$ and $\succeq_B$ is defined as follows: $\mathcal{M} \succeq_B \mathcal{M}'$ if and only if $|\mathcal{M}(B)| \leq |\mathcal{M}'(B)|$. This preference relation means that a model $\mathcal{M}$ is more preferred than a model $\mathcal{M}'$ if the itemset $\{a \in \mathcal{I} \mid \mathcal{M}(p_a) = 1\}$ is more frequent than $\{a \in \mathcal{I} \mid \mathcal{M}'(p_a) = 1\}$. It is a δ-preference relation since one can define the bound function $\delta^{\Phi}_{\succeq_B}$ as follows:

$$\delta^{\Phi}_{\succeq_B}(\mathcal{M}) = (\sum_{i=0}^{n-1} b_i \leq |\mathcal{M}(B)|)$$

Indeed, this formula allows us to have models corresponding to closed itemsets with supports greater than or equal to the support of the closed itemset obtained from $\mathcal{M}$.

4.2. Some variants of $\mathcal{FCI}^k_{min}$

In this section, our aim is to illustrate the nice declarative aspects of our proposed framework. To this end, we simply consider variations of $\mathcal{FCI}^k_{min}$, and show that their encodings can be obtained by slight modifications of that of $\mathcal{FCI}^k_{min}$.

4.2.1. Variant 1 ($\mathcal{FCI}^k_{(min,max)}$)

In this variant, we consider the problem of mining Top-k closed itemsets of size between min and max. This variant can be used to reduce the size of the output by focusing the attention on a size interval (min, max).

Our encoding in this case is obtained by adding to (1), (2) and (3) the following constraint:

$$\sum_{a \in \mathcal{I}} p_a \leq max \tag{4}$$

In this case, we use the δ-preference relation $\succeq_B$ defined previously.

4.2.2. Variant 2 ($\mathcal{FCI}^k_\lambda$)

Let us now propose an encoding of the problem of mining Top-k closed itemsets of supports at least λ (minimal support threshold). In this context, a Top-k closed itemset is a closed itemset such that, for each one, there exist no more than $k-1$ closed itemsets of size greater than its size. In the same way as maximal frequent itemsets mining, the mining task $\mathcal{FCI}^k_\lambda$ allows us to focus on the largest frequent itemsets. Let us recall that a frequent itemset is maximal if all its supersets are infrequent. Formally, given a transaction database $\mathcal{D}$, a minimal support threshold λ and a frequent itemset I ($\mathcal{S}(I, \mathcal{D}) \geq \lambda$), I is said to be maximal w.r.t. λ if, for all J with $I \subset J$, $\mathcal{S}(J, \mathcal{D}) < \lambda$. $\mathcal{FCI}^k_\lambda$ can be used in cases where the size is related to the amount of information.

Our encoding in this case is obtained by adding to (2) and (3) the following constraint:

$$\sum_{i=0}^{n} \overline{b_i} \geq \lambda \tag{5}$$

The preference relation used in this case is $\succeq_I$ defined as follows: $\mathcal{M} \succeq_I \mathcal{M}'$ if and only if $|\mathcal{M}(I)| \geq |\mathcal{M}'(I)|$. It means that a model $\mathcal{M}$ is more preferred than a model $\mathcal{M}'$ if the size of the itemset $\{a \in \mathcal{I} \mid \mathcal{M}(p_a) = 1\}$ is greater than the size of $\{a \in \mathcal{I} \mid \mathcal{M}'(p_a) = 1\}$. It is a δ-preference relation because the bound function $\delta^\Phi_{\succeq_I}$ ($\Phi \equiv (2) \wedge (3) \wedge (5)$) can be defined as follows:

$$\delta^\Phi_{\succeq_I}(\mathcal{M}) = \sum_{a \in I} p_a \geq |\mathcal{M}(I)|$$

5. An application of Top-k SAT in sequence mining

5.1. Frequent pattern mining in a sequence of items ($\mathcal{FPS}$)

In this section, we present the datamining problem of enumerating frequent and closed patterns with wildcards in a sequence of items [3–5].

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 $p \sqsubseteq_l s$, 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 $p \sqsubseteq s$, if $\exists l \in \mathcal{P}_s$ such that $p \sqsubseteq_l s$. The *cover* of p in s is defined as the set $\mathcal{L}_s(p) = \{l \in \mathcal{P}_s | p \sqsubseteq_l s\}$. Moreover, the *support* of p in s is defined as the value $|\mathcal{L}_s(p)|$.

$\mathcal{FPS}$ 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 ($\mathcal{FPS}$)* consists in computing the set $\mathcal{FPS}(s, \lambda)$ of all the frequent patterns w.r.t. λ.

For instance, consider the sequence $s = aaccbcabcba$ and the pattern $p = a \circ c$. We have $\mathcal{L}_s(p) = \{0, 1, 6\}$, since $p \sqsubseteq_0 s$, $p \sqsubseteq_1 s$ and $p \sqsubseteq_6 s$. 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 patterns A frequent pattern p of a sequence s is said to be *closed* if for any frequent pattern q satisfying $p \sqsubset q$, there is no integer d such that $\mathcal{L}_s(q) = \mathcal{L}_s(p) + d$, where $\mathcal{L}_s(p) + d = \{l + d | l \in \mathcal{L}_s(p)\}$. Clearly, the set of closed frequent patterns is a condensed representation of the set of frequent patterns. Indeed, the frequent patterns can be obtained from the closed ones by replacing items with wildcards.

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

Definition 5 ($\mathcal{FCPS}^k_{min}$). Let k and *min* be strictly positive integers. The problem of mining Top-k frequent closed patterns in a sequence $\mathcal{FCPS}^k_{min}$ consists in computing all closed patterns with at least *min* items such that, for each one, there exist no more than $k-1$ closed patterns with at least *min* items and with supports greater than its support.

5.2. SAT-based encoding for $\mathcal{FCPS}^k_{min}$

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 $p_{a,0}, \ldots, p_{a,(k_a-1)}$ where $k_a = max(\mathcal{L}_s(a)) + 1$. 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 $max(\mathcal{L}_s(a)) + 1$ variables to each character a, because $\{0, \ldots, max(\mathcal{L}_s(a))\}$ 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:

$$\bigvee_{a \in \Sigma} p_{a,0} \tag{6}$$

The following constraints allow us to capture all the locations where the candidate pattern appears:

$$\bigwedge_{a \in \Sigma, 0 \leq l \leq n-1, 0 \leq i \leq k_a - 1} (p_{a,i} \wedge s_{l+i} \neq a) \rightarrow b_l \tag{7}$$

$$\bigwedge_{l=0}^{n-1} (b_l \rightarrow \bigvee_{a \in \Sigma, 0 \leq i \leq k_a - 1} (p_{a,i} \wedge s_{l+i} \neq a)) \tag{8}$$

Indeed, the previous constraints allow us to obtain that, if the Boolean interpretation $\mathcal{M}$ is a model of the constraints (6), (7) and (8), then the candidate pattern that corresponds to $\mathcal{M}$ appears only in the locations $\{0 \leq l \leq n-1 | \mathcal{M}(b_l) = 0\}$.

In order to consider the patterns with at least *min* items (solid characters), we just have to add the following constraint:

$$\sum_{a \in \Sigma, 0 \leq i \leq k_a - 1} p_{a,i} \geq min \tag{9}$$

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

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

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

Then, we introduce a 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:

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

where $k'_a = n - min(\mathcal{L}_s(a)) - 1$ for $a \in \Sigma$, for all $a \in \Sigma$, the integers from 1 to k'_a allows us to capture all the possible locations of a on the left side of the candidate pattern.

Proposition 5. *The set of models of (6) ∧ (7) ∧ (8) ∧ (9) ∧ (10) ∧ (11) correspond to the set of closed patterns with at least min items.*

Proof. Part ⇒. Let $\mathcal{M}$ be a model of (6) ∧ (7) ∧ (8) ∧ (9) ∧ (10) ∧ (11) and p its corresponding patterns. Then, using the constraints (6), (7), (8) and (9), we know that p contains more than or equal to *min* items and appears only in the locations $\{0 \leq l \leq n-1 | \mathcal{M}(b_l) = 0\}$. Moreover, using (10) and (11), we have, for all p_1, p_2 and a with $p = p_1 \circ p_2$, $\mathcal{L}_s(p) \neq \mathcal{L}_s(p_1 a p_2)$ holds. Thus, we obtain that p is a closed pattern.

Part ⇐. Let p be a closed pattern with at least *min* items. Then, we define its associated Boolean interpretation $\mathcal{M}$ as follows:

- for all $a \in \Sigma$ and $0 \leq i < k_a$, $\mathcal{M}(p_{a,i}) = 1$ if and only if a is in p at the location i; and
- for all $i \in \{0, \ldots, n-1\}$, $\mathcal{M}(b_i) = 0$ if and only if p appears at the location i.

One can easily see that $\mathcal{M}$ satisfies the constraints (6), (7), (8) and (9), since p is a pattern appearing in s that contains a number of items greater than or equal to *min*. Furthermore, using the fact that I is a closed pattern, we obtain that $\mathcal{M}$ satisfies the constraints (10) and (11). Indeed, we have, for all p_1, p_2 and a with $p = p_1 \circ p_2$, $\mathcal{L}_s(p) \neq \mathcal{L}_s(p_1 a p_2)$ holds, and this property implies that $\mathcal{M}$ satisfies (10) $\wedge$ (11). □

The problem $\mathcal{FCPS}^k_{min}$ is encoded as the problem of computing the Top-k models of (6) $\wedge$ (7) $\wedge$ (8) $\wedge$ (9) $\wedge$ (10) $\wedge$ (11) with respect to $\succeq_B$ and $X = \{p_a | a \in \Sigma\}$, where $B = \{b_0, \ldots, b_{n-1}\}$ and $\succeq_B$ is the δ-preference relation defined in the same way as that in the case of $\mathcal{FCI}^k_{min}$, i.e., $\mathcal{M} \succeq_B \mathcal{M}'$ if and only if $|\mathcal{M}(B)| \leq |\mathcal{M}'(B)|$.

5.3. Frequent pattern mining in a sequence of itemsets ($\mathcal{FPSI}$)

We here define a 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. This problem admits some similarities and differences with the classical sequential pattern mining problem introduced in [33]. 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.

Our goal in this section is to illustrate the flexibility of our framework and its nice declarative aspects. As we can see below, a change in the problem specification induces a small change in the model. Considering sequences of itemsets is also interesting from the practical side. In the literature, several data mining papers consider sequences of itemsets (e.g. [44]). Such data can be found in several applications including web logs, trading where one is interested in analyzing the consumer purchases over time. Mining frequent patterns with periodic wildcard gaps can flexibly reflect the sequential behaviors and is often exhibited in many real-world applications. For example, in business, retail companies may want to know what products customers will usually purchase at regular time intervals rather than in continuous time according to time gaps [45]. As mentioned in the introduction, in biology, patterns with wildcards are redeemed as having significant biological and medical values. Minimum and maximum time gaps are also introduced to constrain two items/itemsets to occur neither too close nor too far apart in time.

Sequence of itemset. 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.

Pattern. 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 $p \sqsubseteq_l s$, at the location $l \in \mathcal{P}_s$ if $\forall i \in \{0 \ldots m-1\}$, $p_i \subseteq s_{l+i}$. The relation $\sqsubseteq$ 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 and closed patterns are also defined in the same way as in the sequences of items. For instance, a frequent pattern p of a sequence s is said to be *closed* if for any frequent pattern q satisfying $p \sqsubset q$, 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 ones by replacing itemsets with their subsets.

For example, 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 $p' \sqsubset p$.

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".

The problem of mining Top-k frequent closed patterns in a sequence of itemsets $\mathcal{FCPSI}^k_{min}$ is defined in the same way as $\mathcal{FCPS}^k_{min}$.

5.4. SAT-based encoding of $\mathcal{FCPSI}^k_{min}$

Our encoding of the problem $\mathcal{FCPSI}^k_{min}$ can be easily obtained from the one of $\mathcal{FCPS}^k_{min}$. We only have to replace the equalities (resp. inequalities) of the form $s_{l \pm i} \neq a$ (resp. $s_{l \pm i} = a$) with $a \notin s_{l \pm i}$ (resp. $a \in s_{l \pm i}$):

$$\bigvee_{a \in \Sigma} p_{a,0} \tag{12}$$

$$\bigwedge_{a \in \Sigma, 0 \leq l \leq n-1, 0 \leq i \leq k_a - 1} (p_{a,i} \wedge a \notin s_{l+i}) \rightarrow b_l \tag{13}$$

$$\bigwedge_{l=0}^{n-1}(b_l \rightarrow \bigvee_{a\in\Sigma, 0\leq i\leq k_a-1}(p_{a,i} \wedge a \notin s_{l+i})) \tag{14}$$

$$\sum_{l=0}^{n-1} b_l \leq n - \lambda \tag{15}$$

$$\bigwedge_{a\in\Sigma, 0\leq i\leq k_a-1}(\bigwedge_{l=0}^{n-1} \overline{b_l} \rightarrow a \in s_{l+i}) \rightarrow p_{a,i} \tag{16}$$

$$\bigwedge_{a\in\Sigma, 1\leq i\leq k'_a} \neg(\bigwedge_{l=0}^{n-1} \overline{b_l} \rightarrow a \in s_{l-i}) \tag{17}$$

Similarly to $\mathcal{FCPS}^k_{min}$, the problem $\mathcal{FCPSI}^k_{min}$ is encoded as the problem of computing the Top-k models of (12) $\wedge$ (13) $\wedge$ (14) $\wedge$ (15) $\wedge$ (16) $\wedge$ (17) with respect to $\succeq_B$ and $X = \{p_a | a \in \Sigma\}$, where $B = \{b_0, \ldots, b_{n-1}\}$.

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

6. Implementation and experiments

In this section, we carried out an experimental evaluation of the performance of our Algorithm for Top-k SAT empirically. The primary goal is to assess the declarativity and the effectiveness of our proposed framework. For this purpose, we consider the problems $\mathcal{FCIM}^k_{min}$ and $\mathcal{FCPS}^k_{min}$.

For our experiments, we implemented the Algorithm 1 (Top-k) on the top of the state-of-the-art SAT solver MiniSAT 2.2[1] adapted to efficiently enumerate models of a propositional formula. Compared to our previous version, the new model enumeration algorithm is based on a DPLL based procedure, without adding blocking and learned clauses [46]. We also set the decision literal polarity to false by default as it is clearly the best option in practice. This version is significantly better than our previous model enumerator implemented using a CDCL based solver with blocking clauses [26].

The cardinality constraints involved in our SAT encodings, are managed dynamically during search. In other words, similarly to constraint programming, a propagator is associated to a cardinality constraint, obtained by maintaining the sum of its assigned variables. Managing cardinality constraints on the fly outperforms our previous implementation [34] where we used the sorting networks, one of the state-of-the-art encoding of the cardinality constraint to CNF proposed in [39].

For the problem $\mathcal{FCIM}^k_{min}$, we considered a variety of datasets (18 data instances) taken from the `FIMI` repository[2] and `CP4IM`.[3] Regarding the problem $\mathcal{FCPS}^k_{min}$, we used two different datasets:

1. *Bioinformatics*: proteinic data encoded as a sequence of items, where an item is an amino-acid[4];
2. *Computer security:* user data drawn from the command histories of UNIX computer users[5] [47].

All the experiments were done on Intel core i7 machine with 8 GB of RAM running at 1.7 GHz.

Concerning the problem $\mathcal{FCIM}^k_{min}$, Table 1 details the characteristics of the different transaction databases ($\mathcal{D}$). The first column mentions the name of the considered data instance. In the second and third column, we give the size of $\mathcal{D}$ in terms of number of transactions (#trans) and number of items (#items) respectively. The fourth column shows the density (dens) of the transaction database, defined as the percentage of 1's in $\mathcal{D}$. The panel of datasets ranges from sparse (e.g. mushroom) to dense ones (e.g. Hepatitis). Finally, in the two last columns, we give the size of the CNF encoding (#vars, #clauses) of $\mathcal{FCIM}^k_{min}$. As we can see, our proposed encoding leads to CNF formula of reasonable size. The maximum size is obtained for the instance `Connect` (67 815 variables and 5 877 720 clauses).

In order to analyze the behavior of our Top-k algorithm on $\mathcal{FCIM}^k_{min}$, we set the minimum length *min* of the itemsets to 1, while the value of k is varied from 100 to 100 000.

In Table 2, we provide the number of Top-k frequent closed itemsets for each data instance according to different values of k. We also provide in parenthesis the total number of models including Top-k and intermediary non Top-k models, found by the model enumerator.

In general, the number of Top-k frequent closed itemsets is slightly around the value of k. We can also observe that there is only one data instance (`Audiology`), where the number of Top-k frequent closed itemsets is significantly greater

[1] MiniSAT: http://minisat.se/.
[2] FIMI: http://fimi.ua.ac.be/data/.
[3] CP4IM: http://dtai.cs.kuleuven.be/CP4IM/datasets/.
[4] http://www.biomedcentral.com/1471-2105/11/175/additional/.
[5] http://kdd.ics.uci.edu/databases/UNIX_user_data/.

Table 1
Characteristics of the datasets.

Instance	#trans	#items	*dens* (%)	#vars	#clauses
Zoo-1	101	36	44	173	2196
Hepatitis	137	68	50	273	4934
Lymph	148	68	40	284	6355
Audiology	216	148	45	508	17 575
Heart-cleveland	296	95	47	486	15 289
Primary-tumor	336	31	48	398	5777
Vote	435	48	33	531	14 454
Soybean	650	50	32	730	22 153
Australian-credit	653	125	41	901	48 573
Anneal	812	93	45	990	39 157
Tic-tac-toe	958	27	33	1012	18 259
German-credit	1000	112	34	1220	73 223
Kr-vs-kp	3196	73	49	3342	121 597
Hypothyroid	3247	88	49	3419	143 043
Chess	3196	75	49	3346	124 797
Splice-1	3190	287	21	3764	727 897
Mushroom	8124	119	18	8348	747 635
Connect	67 558	129	33	67 815	5 877 720

Table 2
Number of Top-*k* models and total number of enumerated models.

Instance/k	100	1000	5000	10 000	50 000	75 000	100 000
Australian-credit	103	1013	5070	10 072	50 389	76 975	100 569
	(261)	(2717)	(19 001)	(39 142)	(215 691)	(330 280)	(435 093)
Heart-cleveland	103	1021	5172	10 507	51 841	78 926	100 678
	(324)	(3909)	(21 855)	(43 641)	(219 591)	(332 703)	(448 248)
Primary-tumor	101	1005	5007	10 242	50 039	77 336	87 231
	(188)	(2030)	(10 935)	(20 873)	(71 173)	(85 003)	(87 231)
Anneal	100	1004	5029	10 148	50 403	75 524	100 996
	(268)	(2942)	(12 650)	(24 745)	(115 708)	(161 776)	(203 796)
Hepatitis	121	1025	5409	11 062	51 823	83 047	104 836
	(335)	(3397)	(20 211)	(42 950)	(191 396)	(274 096)	(346 937)
Zoo-1	103	1027	4568	4568	4568	4568	4568
	(282)	(1991)	(4568)	(4568)	(4568)	(4568)	(4568)
Lymph	100	1004	5226	10 356	51 862	79 185	108 720
	(246)	(2415)	(12 583)	(23 549)	(89 372)	(115 912)	(135 816)
Audiology	221	1583	8615	38 202	144 016	144 016	144 016
	(929)	(10 745)	(52 307)	(107 407)	(496 001)	(716 473)	(964 565)
Soybean	100	1029	5057	10 041	31 760	31 760	31 760
	(207)	(2043)	(9509)	(17 014)	(31 760)	(31 760)	(31 760)
Tic-tac-toe	103	1011	5411	10 717	42 712	42 712	42 712
	(341)	(3300)	(13 413)	(22 568)	(42 712)	(42 712)	(42 712)
Chess	100	1005	5019	10 018	50 174	75 251	100 051
	(152)	(2716)	(17 283)	(35 613)	(182 140)	(285 942)	(377 538)
Vote	105	1062	5147	10 324	50 855	75 016	101 717
	(416)	(4738)	(21 932)	(39 959)	(126 138)	(159 608)	(182 252)
German-credit	100	1013	5031	10 126	50 460	75 145	101 001
	(302)	(3889)	(21 814)	(46 044)	(253 203)	(379 607)	(504 992)
Kr-vs-kp	100	1005	5019	10 018	50 174	75 251	100 051
	(150)	(3589)	(19 929)	(43 726)	(222 536)	(331 336)	(436 830)
Mushroom	100	1000	5016	10 127	50 914	75 364	108 114
	(202)	(2316)	(11 632)	(22 014)	(94 294)	(133 102)	(160 138)
Connect	100	1000	5001	10 001	50 001	75 002	100 004
	(181)	(2084)	(11 127)	(22 300)	(141 083)	(223 922)	(325 512)
Splice-1	102	1005	5041	10 217	50 700	75 670	100 424
	(450)	(5450)	(30 682)	(64 958)	(90 941)	(145 505)	(190 976)
Hypotyroid	100	1004	5035	10 031	50 188	75 239	100 013
	(251)	(1881)	(11 364)	(22 921)	(110 745)	(161 237)	(215 279)

than *k*. Let us note that for a given data instance involving a total of *m* frequent closed itemsets, if *k* is greater than *m* (e.g., `Zoo-1`, `Soybean`, `Tic-tac-toe`) then the number of Top-*k* models remains equal to *m*. Let us also note that computing the Top-*k* models can leads to the same number for different values of *k* (e.g., `Audiology`). To illustrate such a particular case, given a data instance with a number of Top-1 models equal to 100. In this case, the number of Top-*k* models for each value of *k* less than 100 remains equal to 100. However, the total number of models necessary to the generation of the Top-*k* models can vary with *k*. Indeed, the constraints allowing us to cut the non Top-*k* models is activated when the first *k*

Table 3
CPU time (seconds) to compute Top-k itemsets: LCM(Top-k option)/$\mathcal{FCIM}_1^k$.

Instance/k	100	1000	5000	10 000	50 000	75 000	100 000
Chess	0.04/0.04	0.04/0.09	0.05/0.23	0.07/0.4	0.25/1.95	0.31/2.96	0.44/3.87
Heart-cleveland	0.01/0.01	0.03/0.05	0.05/0.16	0.13/0.29	0.53/1.2	0.8/1.73	1.04/2.32
Primary-tumor	0.01/0.03	0.01/0.01	0.01/0.03	0.03/0.05	0.15/0.14	0.18/0.17	0.28/0.16
Australian-credit	0.01/0.02	0.04/0.09	0.15/0.32	0.23/0.62	0.98/2.54	1.25/3.62	1.62/4.74
Anneal	0.07/2.57	0.02/0.04	0.04/0.11	0.07/0.18	0.28/0.67	0.37/0.91	0.55/1.18
Hepatitis	0.001/001	0.01/002	0.06/007	0.10/0.14	0.43/0.51	0.61/0.84	1.2/0.84
Lymph	0.001/0.005	0.01/0.01	0.03/0.05	0.08/0.08	0.38/0.3	0.59/0.41	0.8/0.48
Vote	0.01/0.03	0.03/0.15	0.1/0.44	0.16/0.66	0.47/1.41	0.59/1.60	0.71/1.77
Soybean	0.001/0.01	0.03/0.03	0.03/0.12	0.12/0.20	0.12/0.38	0.12/0.38	0.12/0.38
German-credit	0.01/0.07	0.04/0.17	0.11/0.55	0.17/1.10	0.76/3.94	1.1/5.48	1.43/7.01
Kr-vs-kp	0.01/0.07	0.04/0.15	0.11/0.37	0.12/0.66	0.25/3.02	0.43/4.3	0.61/5.67
Tic-tac-toe	0.001/0.03	0.01/0.14	0.05/0.39	0.06/0.57	0.17/0.77	0.17/0.76	0.17/0.78
Zoo-1	0.001/0.003	0.01/0.007	0.01/0.01	0.01/0.01	0.01/0.01	0.01/0.01	0.01/0.01
Audiology	0.01/0.07	0.01/0.06	0.15/0.26	0.43/0.52	1.71/2.32	2.10/3.20	2.69/4.23
Mushroom	0.01/0.73	0.08/2.3	0.17/5.8	0.26/9.1	0.74/43.48	1.01/59.47	1.03/72.99
Connect	0.81/3.81	1.18/5.6	1.55/10.59	2.64/15.13	2.21/55.61	2.56/83.09	2.8/102.32
Splice-1	0.07/2.57	0.32/6.07	0.93/24.77	1.75/70.00	6.28/332.67	8.32/451.09	10.27/557.92
Hypotyroid	0.03/0.08	0.05/0.10	0.07/0.24	0.1/0.39	0.34/1.45	0.39/2.03	0.57/2.65

models are generated. Consequently, the search tree might depend on the value of k. As a summary, enumerating Top-k is clearly a method of choice for keeping the size of the output under control.

We compared, our method with a LCM algorithm proposed by Takeaki Uno et al. in [48], one of the best specialized algorithm, using the Top-k option. In Table 3, we provide the CPU time in seconds needed by LCM(Top-k option) and $\mathcal{FCIM}_1^k$ respectively (separated by the slash symbol) for different values of k varying from 100 to 100 000.

First, the CPU time needed for computing the Top-k models increases, in general, with k. The `Splice-1` dataset with low density, is the most challenging instance for our approach. Our algorithm computes the Top-100000 in 557.2 seconds. This is not the case for `Mushroom` which is even more sparse (density of 18%). Our experimental evaluation clearly shows that finding the Top-k models (the most interesting ones) can be computed efficiently for small values of k. For example, on most datasets the Top-1000 models are computed in less than 1 second of CPU time, except for `Mushroom`, `Connect` and `Splice-1` dataset, where the Top-1000 are computed in less than 6 seconds. As expected, on all the datasets and all tested values of k, LCM is able to enumerate the Top-k frequent closed itemsets in less than 10 seconds. It is important to note, that our new implementation is able to compute the Top-k on most of the tested instances in less than 100 seconds except for `Splice-1`. Let us give some elements of explanation. It is first important to note that in our SAT based itemset mining encoding, the propositional variables associated to items form a strong backdoor set [49]. Any assignment of the variables from these set leads to a tractable sub-formula that can be decided by unit-propagation. Indeed, the propositional variables associated to transactions are dependent on the variables associated to items. As a consequence, the size of the search tree depends on the number of items. The excessive CPU time obtained on the `Splice-1` data instance can be explained by its high number of items. Indeed, `Splice-1` involves 287 items, the highest number of items among all the data instances. For the `Mushroom` and `Connect` data instances, the number of items is lower, but the higher number of transactions increase the cost of unit propagation.

As a summary, comparatively, to our previously published results the gap with specialized approaches is significantly reduced. The results depicted in Table 3, demonstrate the competitiveness of our declarative approach with state-of-the-art LCM specialized algorithm.

Concerning sequence mining, we are not aware of any available tool that enumerates the Top-k frequent closed patterns with wildcards in a sequence ($\mathcal{FCPS}_{min}^k$) as described in this paper. Consequently, our last experimental evaluation (see Table 4) aims to simply show the flexibility of our proposed declarative approach and its feasibility. However, to provide an idea on the performance gap between our Top-k sequence mining approach and specialized algorithms, we compare with MaxMotif the state-of-the-art sequence mining tool [5].[6] As MaxMotif does not provide a Top-k option, we proceed in two steps. In the first step, we generate all the frequent closed patterns and in a second step we generate the Top-k by sorting the patterns according to their frequencies.

Similarly to $\mathcal{FCIM}_{min}^k$, we set the minimum length min of the patterns to 1, while the value of k is varied from 100 to 10 000.

Let us first mention that on the considered data sets, the encoding of $\mathcal{FCPS}_{min}^k$ leads to large CNF formula comparatively to those considered in $\mathcal{FCIM}_{min}^k$ encoding. Indeed, for the sequence `User-400` containing 400 solid characters, the set of clauses required for its encoding exceeds 4 millions of clauses. The same observation can be made on the number of variables.

[6] MaxMotif: http://research.nii.ac.jp/~uno/code/maxmotif.html.

Table 4
Top-k frequent closed patterns in a sequence ($\mathcal{FCPS}_1^k$).

Instance/k (#vars, #clauses)	100	250	500	1000	2500	5000	10000
User_200_ (3491, 685562)	1.97 (106, 348)	2.1 (392, 802)	2.24 (944, 1285)	2.19 (2084, 2084)	2.19 (2084, 2084)	2.19 (2084, 2084)	2.19 (2084, 2084)
User_400_ (12049, 4070356)	55.95 (113, 412)	61.32 (265, 1045)	64.60 (552, 2059)	73.36 (1832, 4130)	75.25 (3842, 9036)	77.34 (8803, 15893)	77.81 (21772, 25891)
Bioinfo_300_ (5582, 1640349)	8.66 (109, 490)	8.74 (491, 1059)	8.92 (1352, 1924)	9.57 (1352, 2806)	8.57 (3848, 3977)	9.51 (4105, 4105)	9.51 (4105, 4105)
Bioinfo_500_ (9712, 4749779)	66.00 (118, 440)	69.11 (297, 1127)	72.13 (852, 2161)	74.08 (1628, 3937)	79.27 (3436, 8809)	79.12 (9779, 14907)	79.39 (24174, 24323)
Bioinfo_650_ (12456, 7909598)	174.54 (109, 452)	185.08 (344, 1171)	196.69 (530, 2271)	200.19 (1280, 4290)	210.06 (4001, 10163)	222.96 (8759, 19013)	227.40 (26396, 35137)

Table 5
Top-10000 frequent closed patterns in a sequence (MaxMotif in two steps).

User_200_ (3491, 685562)	User_400_ (12049, 4070356)	Bioinfo_300_ (5582, 1640349)	Bioinfo_500_ (9712, 4749779)	Bioinfo_ 650_ (12456, 7909598)
0.3 (2084)	5.16 (33670)	0.41 (4105)	4.41 (24623)	12.80 (59382)

Similarly to $\mathcal{FCIM}_{min}^k$, the CPU time needed for computing the Top-k models increases, in general, with k. Although the model enumeration process using DPLL leads to an important improvement with respect to our first version of CDCL-based model enumerator [26]. In Table 4, for each k and for each instance, the CPU time in seconds needed to compute Top-k frequent closed patterns in a sequence is given. We also provide a couple (x, y) where x represents the number of Top-k models and y the total number of models including the intermediary models necessary to the generation of the Top-k models. We also mention for each data instance the number of variables (#vars) and clauses (#clauses) of its associated CNF encoding.

As expected, the numbers of Top-k patterns is close to k. We can also remark that we have to mine about a factor of 4 of non Top-k models to find the final Top-k models. Overall, less than 120 seconds is needed to enumerate all Top-k patterns for different values of k.

Table 5 illustrates the performances of MaxMotif using two steps as explained above. We tested the five sequence data instances with $k = 10000$. For each data instance, we provide the cumulated CPU time of the two steps in seconds and the total number of frequent closed patterns found in the first step (in parenthesis). As expected, to compute the Top-k, on all considered data instances, MaxMotif (in two steps) requires less than 13 seconds to enumerate the set of Top-10000 frequent closed patterns in a sequence.

Let us also note that mining Top-k frequent closed patterns in a sequence using Top-k SAT approach suffers from the size of the encoding. However, the advantage of our declarative approach is its ability to not enumerate all the Top-k frequent closed patterns through a dynamic use of constraints to cut the search tree.

As a summary, the experiments show clearly the feasibility of our proposed framework. It is clearly competitive on Top-k itemsets mining problem. However, on Top-k sequence mining, MaxMotif (in two steps) is several orders of magnitude better.

7. Conclusion and perspectives

In this paper, we introduce a new problem, called Top-k SAT, defined as the problem of enumerating the Top-k models of a propositional formula. A Top-k model is a model having no more than $k-1$ models preferred to it with respect to the considered preference relation. We also show that Top-k SAT generalizes the two well-known problems: the Partial MAX-SAT problem and the problem of computing minimal models. A general algorithm for this problem is proposed and evaluated on the problem of enumerating Top-k patterns in data mining, namely, the problem of mining Top-k motifs in the transaction databases and in the sequences. In the case of mining sequence data, we introduce a 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.

While our new problem of computing the Top-k preferred models in Boolean satisfiability is flexible and declarative, there are a number of questions that deserve further research efforts. One direction is the study of (preferred/Top-k) model enumeration algorithm so as to achieve a further speedup of the runtime. This fundamental problem has not received a lot of attention in the SAT community, except some interesting works on enumerating minimal/preferred models. We also plan to investigate other variants of the considered data mining problems such as sequences of sequences of items or itemsets. It would be interesting to extend our encodings 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 datamining framework in general and to other important application domains. Finding a better approximation of the initial set of Top-k patterns to reduce the intermediary non Top-k patterns deserves to be investigated.

References

[1] R. Agrawal, T. Imielinski, A.N. Swami, Mining association rules between sets of items in large databases, in: ACM SIGMOD International Conference on Management of Data, ACM Press, Baltimore, 1993, pp. 207–216.
[2] A. Tiwari, R. Gupta, D. Agrawal, A survey on frequent pattern mining: current status and challenging issues, Inf. Technol. J. 9 (2010) 1278–1293.
[3] L. Parida, I. Rigoutsos, A. Floratos, D. Platt, 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, 2000, pp. 297–308.
[4] N. Pisanti, M. Crochemore, R. Grossi, M.F. Sagot, Bases of motifs for generating repeated patterns with wild cards, IEEE/ACM Trans. Comput. Biol. Bioinform. 2 (2005) 40–50.
[5] H. Arimura, T. Uno, An efficient polynomial space and polynomial delay algorithm for enumeration of maximal motifs in a sequence, J. Comb. Optim. 13 (2007) 243–262.
[6] A.W.-C. Fu, R.W.W. Kwong, J. Tang, Mining n-most interesting itemsets, in: Proceedings of the 12th International Symposium on Methodologies for Intelligent Systems, ISMIS 2000, in: Lecture Notes in Computer Science, Springer, 2000, pp. 59–67.
[7] J. Han, J. Wang, Y. Lu, P. Tzvetkov, Mining Top-k frequent closed patterns without minimum support, in: Proceedings of the 2002 IEEE International Conference on Data Mining, ICDM 2002, IEEE Computer Society, 2002, pp. 211–218.
[8] Y. Ke, J. Cheng, J.X. Yu, Top-k correlative graph mining, in: Proceedings of the SIAM International Conference on Data Mining, SDM 2009, 2009, pp. 1038–1049.
[9] E. Valari, M. Kontaki, A.N. Papadopoulos, Discovery of Top-k dense subgraphs in dynamic graph collections, in: Proceedings of the 24th International Conference on Scientific and Statistical Database Management, SSDBM 2012, 2012, pp. 213–230.
[10] H.T. Lam, T. Calders, Mining Top-k frequent items in a data stream with flexible sliding windows, in: Proceedings of the 16th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD 2010, 2010, pp. 283–292.
[11] H.T. Lam, T. Calders, N. Pham, Online discovery of Top-k similar motifs in time series data, in: Proceedings of the Eleventh SIAM International Conference on Data Mining, SDM 2011, 2011, pp. 1004–1015.
[12] Y. Shoham, Reasoning About Change: Time and Causation from the Standpoint of Artificial Intelligence, MIT Press, Cambridge, MA, USA, 1988.
[13] P. Meseguer, F. Rossi, T. Schiex, Soft constraints, in: F. Rossi, P. van Beek, T. Walsh (Eds.), Handbook of Constraint Programming, Elsevier, 2006.
[14] C. Boutilier, R.I. Brafman, C. Domshlak, D.L. Poole, H.H. Hoos, CP-nets: a tool for representing and reasoning with conditional Ceteris Paribus preference statements, J. Artif. Intell. Res. 21 (2004) 135–191.
[15] T. Walsh, Representing and reasoning with preferences, AI Mag. 28 (2007) 59–70.
[16] R.I. Brafman, C. Domshlak, Preference handling – an introductory tutorial, AI Mag. 30 (2009) 58–86.
[17] C. Domshlak, E. Hüllermeier, S. Kaci, H. Prade, Preferences in AI: an overview, Artif. Intell. 175 (2011) 1037–1052.
[18] S. Bistarelli, F. Bonchi, Soft constraint based pattern mining, Data Knowl. Eng. 62 (2007) 118–137.
[19] S. de Amo, M.S. Diallo, C.T. Diop, A. Giacometti, H.D. Li, A. Soulet, Mining contextual preference rules for building user profiles, in: Proceedings of the 14th International Conference on Data Warehousing and Knowledge Discovery, DaWaK'12, 2012, pp. 229–242.
[20] W. Ugarte Rojas, P. Boizumault, S. Loudni, B. Crémilleux, A. Lepailleur, Mining (soft-) skypatterns using dynamic CSP, in: Integration of AI and OR Techniques in Constraint Programming, CPAIOR'14, 2014, pp. 71–87.
[21] B. Négrevergne, A. Dries, T. Guns, S. Nijssen, Dominance programming for itemset mining, in: 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, December 7–10, 2013, 2013, pp. 557–566.
[22] E.D. Rosa, E. Giunchiglia, M. Maratea, Solving satisfiability problems with preferences, Constraints 15 (2010) 485–515.
[23] T. Castell, C. Cayrol, M. Cayrol, D.L. Berre, Using the Davis and Putnam procedure for an efficient computation of preferred models, in: ECAI, 1996, pp. 350–354.
[24] L.D. Raedt, T. Guns, S. Nijssen, Constraint programming for itemset mining, in: ACM SIGKDD, 2008, pp. 204–212.
[25] T. Guns, S. Nijssen, L.D. Raedt, Itemset mining: a constraint programming perspective, Artif. Intell. 175 (2011) 1951–1983.
[26] S. Jabbour, L. Sais, Y. Salhi, Boolean satisfiability for sequence mining, in: CIKM, 2013, pp. 649–658.
[27] H. Cambazard, T. Hadzic, B. O'Sullivan, Knowledge compilation for itemset mining, in: ECAI'10, 2010, pp. 1109–1110.
[28] M. Khiari, P. Boizumault, B. Crémilleux, Combining CSP and constraint-based mining for pattern discovery, in: Computational Science and Its Applications, ICCSA 2010, 2010, pp. 432–447.
[29] J.-P. Metivier, P. Boizumault, B. Crémilleux, M. Khiari, S. Loudni, A constraint-based language for declarative pattern discovery, in: IEEE 11th International Conference on Data Mining Workshops, ICDMW, Vancouver, Canada, 2011, pp. 1112–1119.
[30] T. Guns, A. Dries, G. Tack, S. Nijssen, L. De Raedt, Miningzinc: a modeling language for constraint-based mining, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI'13, 2013, pp. 1365–1372.
[31] T. Guns, S. Nijssen, L. De Raedt, Itemset mining: a constraint programming perspective, Artif. Intell. 175 (2011) 1951–1983.
[32] J. Wang, J. Han, Y. Lu, P. Tzvetkov, TFP: an efficient algorithm for mining Top-k frequent closed itemsets, IEEE Trans. Knowl. Data Eng. 17 (2005) 652–664.
[33] R. Agrawal, R. Srikant, Mining sequential patterns, in: A.L.P.C. Philip, S. Yu (Eds.), Proceedings of the Eleventh International Conference on Data Engineering, ICDE'1995, IEEE Computer Society, 1995, pp. 3–14.
[34] S. Jabbour, L. Sais, Y. Salhi, The Top-k frequent closed itemset mining using Top-k SAT problem, in: ECML/PKDD (3), 2013, pp. 403–418.
[35] G. Tseitin, On the complexity of derivations in the propositional calculus, in: Structures in Constructives Mathematics and Mathematical Logic, Part II, 1968, pp. 115–125.
[36] Z. Fu, S. Malik, On solving the Partial MAX-SAT problem, in: Proceedings of the Ninth International Conference on Theory and Applications of Satisfiability Testing, SAT'06, 2006, pp. 252–265.
[37] J.P. Warners, A linear-time transformation of linear inequalities into conjunctive normal form, Inf. Process. Lett. (1996).
[38] C. Sinz, Towards an optimal CNF encoding of boolean cardinality constraints, in: 11th International Conference on Principles and Practice of Constraint Programming, CP 2005, 2005, pp. 827–831.
[39] N. Eén, N. Sörensson, Translating pseudo-Boolean constraints into SAT, J. Satisf. Boolean Model. Comput. 2 (2006) 1–26.
[40] R. Asin, R. Nieuwenhuis, A. Oliveras, E. Rodriguez-Carbonell, Cardinality networks: a theoretical and empirical study, Constraints 16 (2011) 195–221.
[41] S. Jabbour, L. Saïs, Y. Salhi, A pigeon-hole based encoding of cardinality constraints, in: International Symposium on Artificial Intelligence and Mathematics, ISAIM 2014, Fort Lauderdale, FL, USA, January 6–8, 2014, 2014.
[42] M. Cadoli, On the complexity of model finding for nonmonotonic propositional logics, in: 4th Italian Conference on Theoretical Computer Science, 1992, pp. 125–139.

[43] A.R. Morgado, J.P. Marques-Silva, Good learning and implicit model enumeration, in: International Conference on Tools with Artificial Intelligence, ICTAI'2005, IEEE, 2005, pp. 131–136.
[44] E. Egho, C. Raïssi, T. Calders, N. Jay, A. Napoli, On measuring similarity for sequences of itemsets, Data Min. Knowl. Discov. 29 (2015) 732–764.
[45] Y. Wu, L. Wang, J. Ren, W. Ding, X. Wu, Mining sequential patterns with periodic wildcard gaps, Appl. Intell. 41 (2014) 99–116.
[46] S. Jabbour, L. Sais, Y. Salhi, On SAT models enumeration in itemset mining, CoRR arXiv:1506.02561, 2015.
[47] T. Lane, Filtering techniques for rapid user classification, in: AAAI-98/ICML-98 Joint Workshop on AI Approaches to Time-Series Analysis, 1998, pp. 58–63.
[48] T. Uno, M. Kiyomi, H. Arimura, LCM ver. 2: efficient mining algorithms for frequent/closed/maximal itemsets, in: Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, FIMI'04, Brighton, UK, November 1, 2004, 2004.
[49] R. Williams, C.P. Gomes, B. Selman, Backdoors to typical case complexity, in: Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence, 2003, pp. 1173–1178.