On maximal frequent itemsets mining with constraints

Said Jabbour, Fatima Ezzahra, Imen Ouled Dlala, Badran Raddaoui, Lakhdar Saïs

On Maximal Frequent Itemsets Mining with Constraints

Said Jabbour[1], Fatima Ezzahra Mana[1,3], Imen Ouled Dlala[1,4], Badran Raddaoui[2], and Lakhdar Sais[1]

[1]CRIL-CNRS, Université d'Artois, F-62307 Lens Cedex, France
[2]SAMOVAR, Télécom SudParis, CNRS, Univ. Paris-Saclay, Evry, France
[3]INPT, Institut National des Postes et Telecommunications, Rabat, Morocco
[4]LARODEC, University of Tunis, Tunisia
{dlala,jabbour,sais}@cril.fr,badran.raddaoui@telecom-sudparis.eu

Abstract. Recently, a new declarative mining framework based on constraint programming (CP) and propositional satisfiability (SAT) has been designed to deal with several pattern mining tasks. The itemset mining problem has been modeled using constraints whose models correspond to the patterns to be mined. In this paper, we propose a new propositional satisfiability based approach for mining maximal frequent itemsets that extends the one proposed in [20]. We show that instead of adding constraints to the initial SAT based itemset mining encoding, the maximal itemsets can be obtained by performing clause learning during search. A major strength of our approach rises in the compactness of the proposed encoding and the efficiency of the SAT-based maximal itemsets enumeration derived using blocked clauses. Experimental results on several datasets, show the feasibility and the efficiency of our approach.

1 Introduction

Frequent Itemsets Mining (abbreviated as FIM) is well-known and essential in data mining, knowledge discovery and data analysis. It plays an increasingly important role in a series of data mining applications, such as the discovery of associations rules, correlations, causality, sequential patterns, episodes, partial periodicity, emerging patterns, gradual patterns, and many other important discovery tasks. FIM 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 [4] on association rules and itemset mining, the huge number of works, challenges, datasets and projects show the actual interest in this problem (see [3,25,15,30] and [29] for a survey).

Unfortunately, mining only frequent itemsets generates an overwhelming number of patterns, from which it is difficult to retrieve useful informations. Consequently, for practical data mining, it is important to reduce the size of the output by exploiting the structure of the itemsets data. A well-known condensed representation is the *closed* sets [26,32]; an itemset is closed if it has no superset with the same frequency. Nevertheless, in many applications, especially

in dense data, the set of all closed itemsets remains too large [13]. One of the most known recourse is then to mine the so-called *maximal* itemsets, where an itemset is maximal frequent if it has no superset that is frequent. So, maximal frequent itemsets is a subset of closed frequent itemsets.

In this paper we introduce SATMax, a new algorithm that makes an original use of SAT solvers for efficiently enumerating all maximal patterns embedded in a transaction database. Technically, the idea is to represent a maximal frequent itemset mining task as a propositional formula such that each of its models corresponds to a maximal pattern of interest. The main argument for this encoding is that it allows us to incorporate domain knowledge in the mining process in an easy and flexible manner, among which the maximal constraint, without presupposing deep insights into the mining mechanism. We address this issue by means of propositional satisfiability solving, a prime technology for knowledge representation and reasoning. This extends earlier results in the application of CP and SAT formalisms to data mining, by allowing to deal with optimization problems. SATMax uses a number of optimizations to efficiently prune away a large portion of the search space. It uses a novel progressive focusing technique to eliminate non-maximal itemsets and exploits blocking clauses for fast frequency checking. We conduct an extensive comparative experimental evaluation of SATMax against DMCP [24] a declarative state-of-the-art maximal itemsets mining approach and Eclat [5] a specialized algorithm.

2 Related Works

In the literature, various proposals have been introduced to mine maximal frequent itemsets from a database of transactions. Many of these existing algorithms are based on the enumeration of frequent itemsets. In [22], Bayardo proposed the MaxMiner algorithm which extends the Apriori algorithm. MaxMiner employs a breadth-first traversal of the search space to limit the database scanning. Furthermore, it uses a dynamic heuristic to increase the effectiveness of superset-frequency pruning. Later, several other enhancements have been suggested for mining maximal frequent itemsets. Pincer-Search algorithm combined the top-down and bottom-up techniques to discover the maximal frequent itemsets [23]. Agarwal et al. [2] implemented a depth-first search technique with bitmap representation (DepthProject), in which columns denote the items and rows denote the transactions. Like MaxMiner algorithm, the authors used dynamic reordering and look-ahead pruning. A projection mechanism is used to reduce the size of the database. The authors efficiently find the support counts and give a superset of the maximal frequent itemsets. Burdick et al. introduced MAFIA [6], an extension of DepthProject. They used vertical bit-vector data format. Compression and projection on bitmaps are applied to increase the performance of the proposed algorithm. Unlike DepthProject and MaxMiner pruning techniques, MAFIA used Parent Equivalence Pruning. Also, GenMax [13] is a backtrack search based algorithm for identifying maximal frequent itemsets

from a transactional database. More specifically, this algorithm integrates numerous optimization techniques to prune the search space including progressive focusing that perform maximality checking and diffset propagation for fast support counting. To search maximal frequent itemsets, SmartMiner [33] records at each step tail information to guide the search for new maximal frequent itemsets. Moreover, Eclat algorithm [5] is proposed to find maximal frequent itemsets in transaction databases. This method carries out a depth first search on the subset lattice and determines the support of itemsets by intersecting transaction lists.

Several recent contributions to pattern mining exploit constraint programming and propositional satisfiability [7,12,18,20,27,9,21]. In this context, Guns et al. [14] studied the problem of mining maximal frequent itemsets using CP formalism. More precisely, the authors show how the typical constraint of maximality used in itemset mining can be formulated for use in CP. Besides, in [28], the authors formulate the problem of maximal frequent itemset mining as the enumeration of a set of models of a constraint network by adding a constraint to force the required models to be maximal.

3 Technical Background

This section introduces the preliminaries related to propositional satisfiability, maximal frequent itemset mining and its associated encoding in propositional logic.

3.1 Propositional Satisfiability (SAT)

We consider a standard propositional logical language $\mathcal{L}$ built on a finite set of Boolean variables $p, q, r, \ldots$ and usual connectives $\neg$, $\vee$, $\wedge$, $\rightarrow$ and $\leftrightarrow$ standing for negation, disjunction, conjunction, implication and equivalence connectives, respectively. A literal is either a Boolean variable p or its negation $\neg p$. The two literals p and $\neg p$ are called *complementary*. A clause is a formula that consists of a finite disjunction of literals. A conjunctive normal form formula (abbreviated as CNF) is defined over a set of Boolean variables as a conjunction (also represented as a set) of clauses. Let Φ be a CNF formula. We refer to the set of Boolean variables appearing in Φ as $Var(\Phi)$. Any formula in $\mathcal{L}$ can be represented (while preserving satisfiability) in CNF using a set of clauses interpreted conjunctively.

A truth assignment, or boolean interpretation $\mathcal{B}$ assigns truth values from $\{0, 1\}$ (0 corresponds to $false$ and 1 to $true$) to every Boolean variable. An interpretation can be also seen as conjunctions or sets of literals. It is lifted to clauses and CNF formulas of $\mathcal{L}$ following usual compositional rules. A CNF formula Φ is satisfiable when there exists at least one boolean interpretation $\mathcal{B}$ satisfying it, i.e., $\mathcal{B}(\Phi) = 1$. Otherwise, it is unsatisfiable. If $\mathcal{B}$ satisfies a formula Φ, $\mathcal{B}$ is then called a *model* of Φ and is represented by the set of variables that it satisfies. We refer to the set of models of Φ as $\mathcal{M}(\Phi)$.

SAT is the decision problem of determining the satisfiability of a CNF formula, i.e., whether or not there exists a model of all clauses in the CNF. This well known NP-Complete problem has seen spectacular progress these recent years. Interestingly, state-of-the-art SAT solvers have been shown of practical use, solving real-world instances encoding industrial problems up to millions of variables and clauses. As a consequence, providing SAT encoding for a given problem allows us to benefit from this continous and spectacular progress.

3.2 Frequent Itemset Mining

We are given a set of distinct items (symbols) denoted as $\Omega = \{a, b, c, \ldots\}$. A transaction database $\mathcal{D}$ is a set of transactions $\{t_1, t_2, \ldots, t_n\}$ such that each transaction $t_i \in \mathcal{D}$ ($i \in [1..n]$) is a subset of Ω, i.e., $t_i \subseteq \Omega$. Transactions can represent things such as the supermarket items purchased by a customer during a shopping visit, or the characteristics of a person as described by his or her replies in a census questionnaire. For instance, Table 1 gives a transaction dataset containing seven transactions $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$ described by five items, which will be used as a running example. Besides, each transaction $t_i \in \mathcal{D}$ ($i \in [1..n]$) has an associated unique identifier i called *TID*. A non-null finite subset of items I of Ω is more succinctly called an *itemset* (or pattern). An itemset with k items is called a k-itemset. The notation $I \subseteq t$ will be used to denote that the itemset I is a subset of the set of items that t contains. For convenience, we will often directly refer to a transaction as the set of items that it contains.

Classical data mining problems are usually concerned with itemsets that frequently occur in a database of transactions. The number of occurrences of an itemset in a database is commonly referred to as the support of this itemset. Informally, the support of an itemset measures how often an itemset X occurs in the database. In other words, the support of an itemset is the number of transactions in which that itemset occurs as a subset.

Definition 1. *Given a transaction database $\mathcal{D}$ and an itemset X, the cover of X in $\mathcal{D}$, denoted $Cover(X)$, is defined as follows: $\{J \in \mathcal{D}$ and $X \subseteq J\}$. The support of X in $\mathcal{D}$, denoted $Supp(X)$, corresponds to the cardinality of $Cover(X)$, i.e., $Supp(X) = |Cover(X)|$.*

Example 1. Let us consider the transaction database $\mathcal{D}$ stated in Table 1. There are five different items $\{a, b, c, d, e\}$ and seven transactions $\{t_1, t_2, t_3, t_4, t_5, t_6, t_7\}$. Then, the support of the itemset $X = abc$ is equal to 2, since X occurs in the transactions t_1 and t_2.

Given a transaction database $\mathcal{D}$ over Ω and a minimum support threshold set as λ according to users' preference, the problem of finding the complete set of frequent itemset is called the *frequent itemset mining problem* defined as:

$$FI(\mathcal{D}, \lambda) = \{X \subseteq \Omega \mid Supp(X) \geqslant \lambda\}.$$

TID	Itemset			
t_1	a	b	c	d
t_2	a	b	c	e
t_3	a	e		
t_4	a	e	d	
t_5	a	b		
t_6	b	d		
t_7	b	e		

Table 1: A transaction database $\mathcal{D}$

Unfortunately, identifying the complete set of frequent itemsets may lead to a huge number of patterns. In order to overcome this problem, the concept of closed itemsets is afterward proposed.

Definition 2 (Closed Frequent Itemset). *Let $\mathcal{D}$ be a transaction database and X an itemset. Then, X is a closed itemset if there exists no itemset X' such that $X \subseteq X'$, and $\forall t \in \mathcal{D}$, $X \in t \rightarrow X' \in t$.*

That is, enumerating all closed itemsets allows us to reduce the size of the output.

Extracting all the elements of $FI(\mathcal{D}, \lambda)$ can be obtained from the closed itemsets by computing their subsets. Then, we have $CFI(\mathcal{D}, \lambda) \subseteq FI(\mathcal{D}, \lambda)$.

Example 2. Let us consider again our example described in Table 1. The set of closed frequent itemsets with the minimal support threshold equal to 2 is: $CFI(\mathcal{D}, 2) = \{a, d, ab, ae, abc\}$.

In order to reduce the large number of extracted closed frequent itemsets, another condensed representation, called maximal frequent itemsets, has been introduced.

Definition 3 (Maximal Frequent Itemset). *Let $\mathcal{D}$ be a transaction database over Ω and $X \subseteq \Omega$ an itemset. We say that X is a maximal frequent itemset in $\mathcal{D}$ given a minimum threshold λ, if $X \in FI(\mathcal{D}, \lambda)$, and there exists no other itemset Y s.t. $X \subset Y$ and $Y \in FI(\mathcal{D}, \lambda)$.*

That is, if the itemset X is frequent and no superset of X is frequent, then we say that X is a maximal frequent itemset. This condensed representation is the one which store most of the information contained in frequent itemsets using less space.

In this work, we are interested in the problem of mining maximal frequent itemsets, abbreviated as **MFI**. More formally,

$$MFI(\mathcal{D}, \lambda) = \{X \subseteq \Omega \mid Supp(X) \geqslant \lambda \text{ and } \nexists Y \supset X, \text{ s.t. } Y \in FI(\mathcal{D}, \lambda)\}$$

Given a transaction database $\mathcal{D}$, it is important to note that the set of maximal frequent itemsets is a subset of frequent closed ones, i.e., $MFI(\mathcal{D}, \lambda) \subseteq CFI(\mathcal{D}, \lambda)$.

3.3 SAT-Based Itemset Mining

This section presents a brief overview of the SAT-based approach for enumerating all frequent itemsets in a transaction database proposed in [16,20]. The authors have shown that such mining task can be encoded as a propositional formula whose models are in bijection with the patterns to be mined.

The basic idea consists in the use of two kinds of propositional variables: the *i-variable* p_a to represent each item $a \in \Omega$, and the *t-variable* q_i to represent each transaction t_i.

Next, the SAT encoding is based on the following three CNF formulas built over the previous propositional variables.

$$\bigwedge_{i=1}^{n} (\neg q_i \leftrightarrow \bigvee_{a \in \Omega \setminus t_i} p_a) \tag{1}$$

The first constraint (1) allows to model the transaction database and then to catch the itemsets. So, an itemset appears in a transaction t_i (i.e., $q_i = 1$) iff the boolean variables associated to items not involved in t_i are set to false. Notice that the formula $(\neg q_i \leftrightarrow \bigvee_{a \in \Omega \setminus t_i} p_a)$ can be translated into the following CNF formula:

$$\bigwedge_{a \in \Omega \setminus t_i} (\neg q_i \vee \neg p_a) \wedge (q_i \vee \bigvee_{a \in \Omega \setminus t_i} p_a)$$

$$\sum_{i=1}^{n} q_i \geqslant \lambda \tag{2}$$

Constraint (2) allows us to consider the itemsets having a support greater than or equal to the minimum threshold λ. This encoding is defined as a 0/1 linear inequality, usually called *cardinality constraint*. Because of the presence of such constraint in several applications, many efficient CNF encodings have been proposed over the years. Mostly, such encodings try to derive the best compact representation while maintaining constraint propagation (e.g. [19]).

$$\bigwedge_{a \in \Omega} ((\bigvee_{a \notin t_i} q_i) \vee p_a) \tag{3}$$

Formula (3) expresses the closure property. Intuitively, if the itemset is involved in all transactions containing the item a, then a must be added to the candidate itemset. In other words, when in all the transactions where a does not appear, the candidate itemset is not included, this implies that the candidate itemset appears only in transactions containing the item a. Consequently, to be closed, the item a must be added to the final candidate itemset.

The main advantage of the SAT-based approach is its ability to easily integrate other user constraints. For instance, enumerating itemsets of size at most k can be expressed by simply adding the linear constraint $\sum_{a \in \Omega} p_a \leqslant k$.

4 SAT-based Approach for Efficient MFI Mining

In this section, we introduce our SAT-based formuation that enables us to specify in term of constraints maximal frequent pattern mining problem. Given a transaction database and a user specified threshold value, our goal is to provide a simple and efficient way to model and enumerate all maximal frequent itemsets.

As mentioned in Section 3.2, an itemset X is maximal if X is frequent and each superset of X is not frequent. Clearly, this requirement can be expressed by the following constraint:

$$\bigwedge_{a \in \Omega} \neg p_a \rightarrow (\sum_{t_i | a \in t_i} q_i < \lambda) \tag{4}$$

That is, formula (4) expresses that if the item a is not added to the final candidate itemset X, this means that the occurrence frequency of X in the transactions containing a is lower than the minimum threshold λ. Notice that the constraint (4) represents a conditional cardinality constraint.

Interestingly, we can naturally translate the formula (4) to a *Pseudo Boolean constraint* [1] as follows:

$$\bigwedge_{a \in \Omega} (((\lambda - Supp(\{a\}) - 1) \times \neg p_a + \sum_{t_i | a \in t_i} q_i) < \lambda) \tag{5}$$

In the literature, various approaches proposed different efficient encodings of Pseudo Boolean constraints as CNF formula [1,11,31]. This transformation can be useful if the number of items and their associated transactions are small. Unfortunately, it is ineffective for large datasets, since it can lead to large CNF formulas. Indeed, each item will be associated with a Pseudo Boolean constraint (5). Consequently, the weakness of SAT-based approaches resides in the size of the encoding, which for large formulas can outgrow available memory or can make SAT solving otherwise inefficient.

Alternatively, another way to manipulate such Pseudo Boolean constraints is to associate a propagator to each constraint of the form (5), as done in constraint programming. However, this case can be challenging since we have to go through each constraint at each decision to check the satisfiability of such constraints.

In order to avoid the addition of conditional cardinality constraints to our initial encoding, we propose in the sequel an original method that allows to insert additional clauses in an incremental manner, throughout the search process, with the aim of ensuring that the found models correspond exactly to the maximal frequent itemsets of the given transaction database. For this purpose, we consider as our SAT solver a DPLL-like procedure that firstly assigns the i-variables. To illustrate our approach, we assume that the solver assigns the truth value *true*

[1] A pseudo Boolean constraint over boolean variables is defined by $\sum_i c_i.l_i \triangleright k$ where c_i are the coefficients, k an integer constant, l_i are literals and $\triangleright$ is one of the operators $\{=, <, \leqslant, >, \geqslant\}$.

to the i-variables. Let us refer to a given model of the CNF formulas encoding the FIM task as $\mathcal{B}$, and $P(\mathcal{B}) = \{a \mid \mathcal{B}(p_a) = 1\}$ will denote the corresponding frequent itemset. Clearly, the first found model $\mathcal{B}$ corresponds to a maximal frequent itemset $P(\mathcal{B})$. In fact, assigning to true the i-variables is a way to derive a maximal itemset.

Now, in order to discard retrieving a model $\mathcal{B}'$ such that $P(\mathcal{B}') \subset P(\mathcal{B})$, one need to eliminate (or block) all itemsets $X \subset P(\mathcal{B})$. To do so, it is sufficient to add the blocking clause $C = (\bigvee_{a \in \Omega \setminus P(\mathcal{B})} p_a)$ to the original encoding. The solver can then backtrack and explore new search space by performing positive assignment of the i-variables.

So, the main idea of our approach consists in adding blocking clauses every time a model is found. It is worth to remark that such clauses are composed of the literals that are assigned to false under the current assignment. This means that such clauses are false before backtracking. In order to enumerate more effectively the set of all maximal frequent itemsets, one need to take the level of literals of each blocking clause C into account to backtrack at the adequate level. This can be seen as a new form of clause learning.

For real-word problems, the items not taking part in each transaction t_i are generally more numerous than those involved in t_i, i.e., $|t_i| \leqslant |\Omega \setminus t_i|$. Consequently, each blocking clause C used to discard non-maximal itemsets can be large. For example, let us suppose that the current itemset appears in the transaction t_i. Clearly, the clause C can be written as $C = (\bigvee_{a \in t_i \setminus P(\mathcal{B})} p_a \vee \bigvee_{a \in \Omega \setminus t_i} p_a)$. On the other hand, using the constraint (1), the size of C can be considerably reduced by rewriting it as $C = (\bigvee_{a \in t_i \setminus P(\mathcal{B})} p_a \vee \neg q_i)$.
Additionally, we can choose the most suitable clause C by choosing the smallest transaction t_i containing $P(\mathcal{B})$. Roughly speaking, the size of the blocking clause C clearly depends on the choice of the transaction t_i.

Example 3. Let us consider the transaction database depicted in Table 1. We further assume a minimum threshold $\lambda = 2$. Now, suppose that the SAT solver chooses the following variables ordering during the search process: p_a, p_b, p_c, $\neg p_d$, and $\neg p_e$ (see the search tree depicted by Figure 1). Then, a first model $\mathcal{B} = \{p_a, p_b, p_c, \neg p_d, \neg p_e\}$ can be obtained by assigning p_a at level 1, p_b at level 2, and p_c at level 3. Hence, the added blocking clause is $C = (p_d \vee p_e)$. In this case, the solver must backtrack to the level 2 since C becomes falsified in level 3 and causes a conflict.

Next, we show the potential behind using blocking clauses in order to significantly improve the mining efficiency. Let us first remark that the blocking clauses involve positive i-variables. More specifically, let $C = (p_{a_1} \vee \ldots \vee p_{a_k})$ be a blocking clause. According to the constraint (1), each item a_i is involved in many negative binary clauses of the form:

$$\bigwedge_{t_j \in \mathcal{D} | a_i \notin t_j} (\neg p_{a_i} \vee \neg q_j)$$

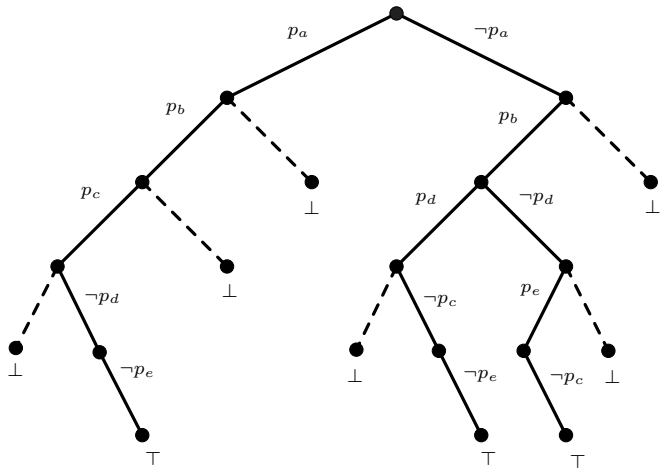

Fig. 1: Search tree of Example 3

Now, one of the most known form of resolution called *hyper binary* resolution [17] can be applied between C and the previous set of negative binary clauses clauses. This gives us the following general constraint of the form:

$$\bigwedge_{i_1 \notin Cover(\{a_1\}) \ldots i_k \notin Cover(\{a_k\})} (\neg q_{i_1} \vee \ldots \vee \neg q_{i_k}) \tag{6}$$

Interestingly enough, the constraint (6) involves only negative clauses (i.e., disjunction of negative literals) over t-variables. These clauses can help improving the efficiency of the frequency constraint (2) by requiring that at least one of the t-variables $\{q_{i_1}, \ldots, q_{i_k}\}$ must be false. Unfortunately, when the length of C is large, a great number of clauses can be derived by hyper binary resolution, which leads to excessive space complexity that might slowdown the solver. An alternative is to limit the application of hyper binary resolution to the case where the derived clauses are relevant or of small size. For efficiency reason, we consider the case where the constraint (6) gives rise to a unit clause:

$$\bigwedge_{i \notin \bigcup_{1 \leq j \leq k} Cover(\{a_j\})} (\neg q_i) \tag{7}$$

Intuitively, the constraint (7) aims to exclude each transaction that does not contain none of the i-variables of the blocking clause C. Indeed, any t-variable that do not belong to $\bigcup_{1 \leq j \leq k} Cover(\{a_j\})$ must be assigned to false by unit propagation. In fact, the clause $C = (p_{a_1} \vee \ldots \vee p_{a_k})$ requires that at least one of its literal must be *true* and consequently the transactions not involving none of

items corresponding to literals of C must be assigned to false. Doing so, we are able to effectively improve the resolution process when $\bigcup_{1 \leq j \leq k} Cover(\{a_j\}) \neq \emptyset$.

Example 4. Let us take the transaction database of Example 1. Assume that the first found model is $\{p_a, p_b, p_c, \neg p_d, \neg p_e\}$. Then, using the blocking clause $(p_d \vee p_e)$ and the constraint (7), we deduce that q_5 and q_6 must be assigned to $false$.

As mentioned previously, our method requires the addition of a blocking clause once a model is found. Then, the maximum number of blocking clauses that can be added is equal to the number of maximal frequent itemsets. Fortunately, even if the number of added blocked clauses might be large, the experiments show that it is feasible in practice.

Let us now present our general SATMax algorithm for SAT-based MFI enumeration task. To summarize the idea behind Algorithm 1, we first encode the closed frequent itemset mining task, then we use a DPLL-like algorithm, while adding a blocking clause each time a model is found. In this way, the models are restricted to those corresponding to maximal frequent itemsets in the given transaction database.

At first, our algorithm encodes the closed frequent itemsets mining task $CFI(\mathcal{D}, \lambda)$ into CNF (line 1). Then, a DPLL procedure is called. It iteratively picks an i-variable (line 22), assigns it to true and performs unit propagation (line 5). Then, we can distinguish two cases: (1) when a conflict occurs (line 6), in this case if the level of the conflict is 0, the enumeration terminates and the set of MFI is returned. Otherwise, a simple backtrack is performed; (2) when there is no conflict. Here, the procedure continues by checking the satisfiability of the frequency constraint (line 15). Then, the same later steps are performed. If all the i-variables are assigned without conflict, then a model is found and a maximal frequent itemset is extracted (line 16). Then, a blocking clause is built (line 17) and analyzed (line 18) to determine the backtracking level. A backtracking is performed accordingly and the procedure loops.

Proposition 1 (Correctness). *SATMax returns all and only the maximal frequent itemsets in the given transaction database.*

5 Experimental Validation

In this section, we evaluate the performance of SATMax. Our mining solver is implemented using a model enumeration MiniSAT solver based on the DPLL (Davis-Putnam-Logemann-Loveland) procedure [8], as described in Algorithm 1. All our experiments were performed on a 2.66 Ghz Intel Xeon quad-core PC with 32GB of memory, running Ubuntu Linux.

In our SATMax algorithm, the i-variables are firstly assigned. Note that our SAT solver does not branch on t-variables. In fact, the i-variables constitute a strong backdoor, i.e., the t-variables are boolean functions of i-variables (constraint (1)). In our algorithm, each time a model is found, we add a blocking

Algorithm 1: SAT-based MFI enumeration (SATMax)

Input: $\mathcal{D}$: a transaction database, λ: a minimum support threshold
Output: $\mathcal{M}$: maximal frequent itemsets of $\mathcal{D}$

$\Phi \leftarrow encodeCFI(\mathcal{D}, \lambda)$;
$\mathcal{B} = \emptyset$; /* Current interpretation */
$\mathcal{M} = \emptyset$; /* Set of Maximal Frequent Itemsets */
while *(true)* **do**
  C = propagate(Φ);
  if *(C!=null)* **then**
    if *(decisionLevel* $== 0$*)* **then return** M;
    backtrack();
  else
    if $(\sum_{i=1}^{n} q_i < \lambda)$ **then**
      if *(decisionLevel* $== 0$*)* **then return** M;
    else
      backtrack();
    end
    if $(Satisfiable(\Phi) == true)$ **then**
      $\mathcal{M} = \mathcal{M} \cup \{\mathcal{B} \cap \Omega\}$;
      $C \leftarrow \bigvee_{a \in \Omega \mid \neg p_a \in \mathcal{B}} p_a$;
      $\Phi \leftarrow \Phi \wedge C$;
      $k \leftarrow analyze(C)$;
      backtrackUntil(k);
    else
      $selectVariable(\Phi)$;
    end
  end
end

clause and perform a backtracking after analyzing the blocking clause as described in Section 4. For the variable ordering heuristic, we follow the one used in [10]. We empirically evaluated our novel approach using different datasets coming from FIMI [2] and CP4IM [3] repositories. A CPU time limit is fixed to 1200 seconds per instance. We also use the symbol (-) in Table 2 to mention that the algorithm is not able to scale on the considered dataset under the time limit. In our experiments, we considered different minimum support threshold values. For baseline comparison, we retain the dedicated algorithm `Eclat` [5] and also `DMCP` [24], a custom CP bitvector solver. For each method, we report the time needed to enumerate all MFI. Table 2 summarizes our empirical results.

While conducting experiments comparing the three different algorithms, we observed that the performance can vary significantly depending on the dataset characteristics and especially the minimum support threshold values. In many cases our `SATMax` solver is able to compute all MFI, and improves or meets the dedicated solver `Eclat`. More interesting enough, `SATMax` achieves better performances than the CP-based baseline on most considered datasets. In addition, on `BMS-WebView-1` we find that `SATMax` is significantly faster than DMCP for all the considered support threshold values. For the dataset `accidents`, our approach outperforms considerably `DMCP`. Moreover, for some minimum support threshold values, `DMCP` fails to get the maximal frequent itemsets in some instances under the time limit.

Compared with the specialized solver `Eclat`, this latter is generally the best. Nevertheless remarkably, for some dataset and minimum support threshold values our approach outperforms `Eclat`. This is the case for `connect`, `pumsb` and `accidents` when the minimum threshold becomes smaller, `Eclat` becomes worst.

Finally, we run our `SATMax` solver on large problem instances to evaluate its robustness and scalability. For this, we used the `kosarak` instance containing 990002 transactions. We find that `DMCP` for such dataset is not able to scale for all the minimum support threshold values under the time limit. Interestingly, `SATMax` enumerates all MFI for the different support values.

Overall, on maximal frequent itemsets task, the results seem to strongly suggest that SATMax is very promising.

Finally in Figure 2, we compare the number of closed and maximal frequent itemsets for some datasets when the minimum support threshold is varied. We have observed that the number of maximal frequent itemsets is limited relatively to closed ones. For the considered datasets, the maximal frequents itemsets does not exceed 10% of closed frequent patterns. More interesting enough, this number can be much more limited. This is the case for `connect` and `kosarak` datasets.

6 Conclusion

In this paper, we presented an efficient and scalable approach for computing all maximal frequent itemsets using propositional satisfiability. Based on the closed

[2] `http://fimi.ua.ac.be/data/`

[3] `http://dtai.cs.kuleuven.be/CP4IM/datasets/`

instance (#item, #trans, density)	min_supp λ	Eclat time(s)	DMCP time(s)	SATMax time(s)
chess (75, 3196 , 49%)	2000	0.11	0.09	0.28
	1500	1.09	1.44	0.52
	1000	5.67	10.56	**3.75**
	500	33.35	104.56	**85.46**
connect (129, 67558, 35.62%)	40000	0.29	1.29	5.14
	30000	0.66	2.11	6.06
	20000	3.4	4.65	9.22
	10000	36.83	90.95	**22.21**
	5000	94.46	–	**51.38**
kosarak (41267, 990002, 0.01%)	3000	2.52	–	**30.00**
	2500	3.08	–	**32.96**
	2000	7.97	–	**42.94**
	1500	31.52	–	**59.03**
	1000	67.96	–	**100.31**
pumsb (2113, 49046, 3%)	40000	0.30	2.92	5.51
	35000	1.05	11.43	**6.44**
	30000	3.48	32.71	**11.23**
	25000	89.29	473	**49.66**
	20000	878.02	–	**202.71**
retail (16470, 88162, 0.06%)	400	0.29	1.67	1.87
	350	0.26	1.19	2.62
	300	0.33	1.48	2.68
	250	0.58	2.34	**2.09**
	200	1.19	3.67	4.95
T10I4D100K (870, 100000, 1.0%)	500	0.21	1.77	2.88
	400	0.22	1.87	3.28
	300	0.24	2.48	4.14
	200	0.28	3.63	5.62
	100	0.32	6.44	12.43
T40I10D100K (8942, 100000, 4.31%)	10000	0.45	1.09	2.73
	8000	0.62	0.93	4.03
	6000	1.06	1.68	6.53
	4000	1.85	3.03	9.48
	2000	3.50	7.72	21.53
BMS-WebView-1 (497, 59602, 0.5%)	48	0.07	20.51	**2.94**
	36	0.22	195.68	**5.56**
	34	0.28	335.13	**7.05**
	32	0.36	553.39	**7.43**
	30	0.49	1049.28	**7.14**
accidents (468, 340183, 7%)	100000	12.92	33.59	54.73
	80000	46.41	50.22	92.29
	60000	128.85	407.75	**174.34**
	40000	324.49	–	**361.40**
	20000	1206.27	–	**994.07**

Table 2: Maximal Itemsets: SATMax vs Eclat vs DMCP

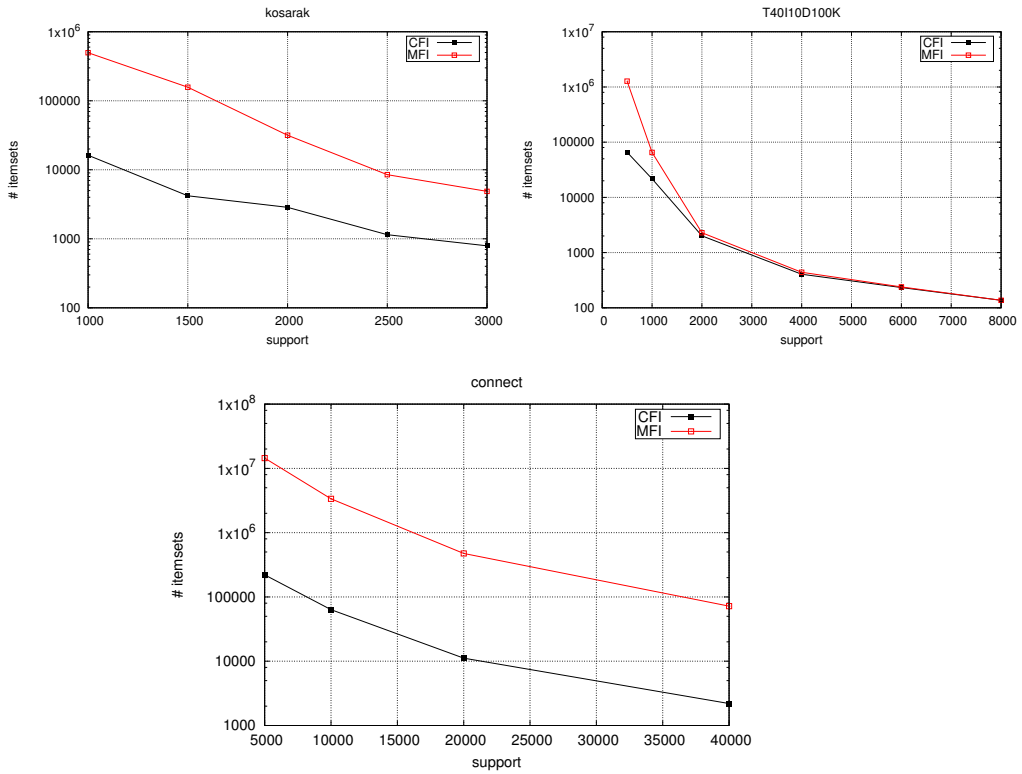

Fig. 2: Frequent itemsets: Closed vs Maximal

frequent itemset SAT encoding, an original DPLL-based model enumeration algorithm combined with clauses learning from models allows us to restrict the models to maximal frequent itemsets. Experimental results on several datasets have shown that our approach is very effective compared to Eclat and DMCP, a specialized and CP-based algorithms, respectively. Interestingly, our approach allows us reduce the size of the encoding by avoiding the integration of the maximality constraints.

As a future work, we plan to pursue our investigation in order to improve MFI task using propositional satifiability. For example, it would be interesting to parallelize our SATMax based approach. Finally, clause learning, an important component for the efficiency of modern SAT solvers, admits several limitations in the context of model enumeration. An important issue is to study how such pivotal mechanism can be efficiently integrated when maximal itemset generation is considered.

References

1. I. Abío, R. Nieuwenhuis, A. Oliveras, E. Rodríguez-Carbonell, and V. Mayer-Eichberger. A new look at bdds for pseudo-boolean constraints. *J. Artif. Intell. Res. (JAIR)*, 45:443–480, 2012.
2. R. C. Agarwal, C. C. Aggarwal, and V. V. V. Prasad. Depth first generation of long patterns. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '00, pages 108–118, 2000.
3. R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, SIGMOD '93, pages 207–216, New York, NY, USA, 1993. ACM.
4. R. Agrawal and R. Srikant. Fast algorithms for mining association rules in large databases. In *VLDB'94, Proceedings of 20th International Conference on Very Large Data Bases*, pages 487–499, 1994.
5. C. Borgelt. Frequent item set mining. *Wiley Interdisc. Rew.: Data Mining and Knowledge Discovery*, 2(6):437–456, 2012.
6. D. Burdick, M. Calimlim, and J. Gehrke. Mafia: A maximal frequent itemset algorithm for transactional databases. In *In ICDE*, pages 443–452, 2001.
7. E. Coquery, S. Jabbour, L. Saïs, and Y. Salhi. A sat-based approach for discovering frequent, closed and maximal patterns in a sequence. In *Proceedings of the 20th European Conference on Artificial Intelligence (ECAI'12)*, pages 258–263, 2012.
8. M. Davis, G. Logemann, and D. Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.
9. I. O. Dlala, S. Jabbour, B. Raddaoui, L. Sais, and B. B. Yaghlane. A sat-based approach for enumerating interesting patterns from uncertain data. In *28th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2016, San Jose, CA, USA, November 6-8, 2016*, pages 255–262, 2016.
10. I. O. Dlala, S. Jabbour, L. Saïs, and B. B. Yaghlane. A comparative study of sat-based itemsets mining. In *Research and Development in Intelligent Systems XXXIII - Incorporating Applications and Innovations in Intelligent Systems XXIV. Proceedings of AI-2016, The Thirty-Sixth SGAI International Conference on Innovative Techniques and Applications of Artificial Intelligence, Cambridge, UK, December 13-15, 2016*, pages 37–52, 2016.
11. N. Eén and N. Sörensson. Translating pseudo-boolean constraints into SAT. *JSAT*, 2(1-4):1–26, 2006.
12. M. Gebser, T. Guyet, R. Quiniou, J. Romero, and T. Schaub. Knowledge-based sequence mining with ASP. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July*, 2016.
13. K. Gouda and M. J. Zaki. GenMax: An efficient algorithm for mining maximal frequent itemsets. *Data Min. Knowl. Discov.*, 11(3):223–242, 2005.
14. T. Guns, S. Nijssen, and L. D. Raedt. Itemset mining: A constraint programming perspective. *Artif. Intell.*, 175(12-13):1951–1983, 2011.
15. J. Han, J. Pei, and Y. Yin. Mining frequent patterns without candidate generation. *SIGMOD Rec.*, 29:1–12, 2000.
16. R. Henriques, I. Lynce, and V. M. Manquinho. On when and how to use sat to mine frequent itemsets. *CoRR*, abs/1207.6253, 2012.
17. M. Heule, M. Järvisalo, and A. Biere. Revisiting hyper binary resolution. In *International Conference on Integration of AI and OR Techniques in Constraint Programming*, pages 77–93, 2013.

18. S. Jabbour, L. Sais, and Y. Salhi. Boolean satisfiability for sequence mining. In *22nd ACM International Conference on Information and Knowledge Management (CIKM'13)*, pages 649–658. ACM, 2013.
19. S. Jabbour, L. Sais, and Y. Salhi. A pigeon-hole based encoding of cardinality constraints. *TPLP*, 13(4-5-Online-Supplement), 2013.
20. S. Jabbour, L. Sais, and Y. Salhi. The top-k frequent closed itemset mining using top-k sat problem. In *European Conference on Machine Learning and Knowledge Discovery in Databases (ECML/PKDD'03)*, pages 403–418, 2013.
21. S. Jabbour, L. Sais, and Y. Salhi. Mining top-k motifs with a sat-based framework. *Artif. Intell.*, 244:30–47, 2017.
22. R. J. B. Jr. Efficiently mining long patterns from databases. In *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA.*, pages 85–93, 1998.
23. D.-I. Lin and Z. M. Kedem. *Pincer-search: A new algorithm for discovering the maximum frequent set*, pages 103–119. Springer Berlin Heidelberg, 1998.
24. S. Nijssen and T. Guns. Integrating constraint programming and itemset mining. In *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML PKDD 2010, Barcelona, Spain, September 20-24, 2010, Proceedings, Part II*, pages 467–482, 2010.
25. J. Pei, J. Han, H. Lu, S. Nishio, S. Tang, and D. Yang. H-mine: hyper-structure mining of frequent patterns in large databases. In *Data Mining, 2001. ICDM 2001, Proceedings IEEE International Conference on*, pages 441–448, 2001.
26. J. Pei, J. Han, and R. Mao. CLOSET: an efficient algorithm for mining frequent closed itemsets. In *2000 ACM SIGMOD Workshop on Research Issues in Data Mining and Knowledge Discovery*, pages 21–30, 2000.
27. L. D. Raedt, T. Guns, and S. Nijssen. Constraint programming for itemset mining. In *ACM SIGKDD*, pages 204–212, 2008.
28. L. D. Raedt, T. Guns, and S. Nijssen. Constraint programming for itemset mining. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 204–212, 2008.
29. A. Tiwari, R. Gupta, and D. Agrawal. A survey on frequent pattern mining: Current status and challenging issues. *Inform. Technol. J*, 9:1278–1293, 2010.
30. T. Uno, M. Kiyomi, and H. Arimura. LCM ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *FIMI '04, Proceedings of the IEEE ICDM Workshop on Frequent Itemset Mining Implementations, Brighton, UK, November 1, 2004*, 2004.
31. J. P. Warners. A linear-time transformation of linear inequalities into conjunctive normal form. *Inf. Process. Lett.*, 68(2):63–69, 1998.
32. M. J. Zaki and C. Hsiao. CHARM: an efficient algorithm for closed itemset mining. In *Proceedings of the Second SIAM International Conference on Data Mining*, pages 457–473, 2002.
33. Q. Zou, W. W. Chu, and B. Lu. Smartminer: A depth first algorithm guided by tail information for mining maximal frequent itemsets. In *Proceedings of the 2002 IEEE International Conference on Data Mining (ICDM 2002), 9-12 December 2002, Maebashi City, Japan*, pages 570–577, 2002.