On the Enumeration of Association Rules: A Decomposition-based Approach

Yacine Izza, Said Jabbour, Badran Raddaoui, Abdelhamid Boudane

On the Enumeration of Association Rules: A Decomposition-based Approach

Yacine Izza[1,2] , **Said Jabbour**[2] , **Badran Raddaoui**[3] and **Abdelhamid Boudane**[4]
[1]ANITI, Toulouse, France
[2]CRIL & CNRS, Université d'Artois, Lens, France
[3]SAMOVAR, Télécom SudParis, Institut Polytechnique de Paris, France
[4]LMTO, Ecole Militaire Polytechnique, Algeria
{izza, jabbour}@cril.fr, badran.raddaoui@telecom-sudparis.eu, boudane.abdelhamid@gmail.com

Abstract

While traditional data mining techniques have been used extensively for finding patterns in databases, they are not always suitable for incorporating user-specified constraints. To overcome this issue, CP and SAT based frameworks for modeling and solving pattern mining tasks have gained a considerable audience in recent years. However, a bottleneck for all these CP and SAT-based approaches is the encoding size which makes these algorithms inefficient for large databases. This paper introduces a practical SAT-based approach to discover efficiently (minimal non-redundant) association rules. First, we present a decomposition-based paradigm that splits the original transaction database into smaller and independent subsets. Then, we show that without producing too large formulas, our decomposition method allows independent mining evaluation on a multi-core machine, improving performance. Finally, an experimental evaluation shows that our method is fast and scale well compared with the existing CP approach even in the sequential case, while significantly reducing the gap with the best state-of-the-art specialized algorithm.

1 Introduction

In the recent years, there is an increasing need to discover relations among large transactional databases, which may be tackled by association rules mining. The outputs generated from the association rule mining are some rules, which pass the user-specified minimum support and confidence measures. Each association rule consists of a couple of itemsets representing the *antecedent* and *consequent* of the rule. As the set of association rules (ARs, for short) in large databases can rapidly grow to be unwieldy, many other kinds of ARs have been studied essentially to reduce the redundancy inherent in a collection of rules. The most common ones are *minimal non-redundant association rules* (MNRs, for short).

Mining ARs is beneficial to the correct and appropriate decision made by decision makers. Since the first well known application usually referred to as market basket data analysis [Agrawal and Srikant, 1994], today, research work on ARs is motivated by a range of application areas, such as banking, manufacturing, medical diagnosis, and telecommunications.

Over the last decade, there have been a lot of proposals of ARs mining (see [Ceglar and Roddick, 2006] for a survey). These work can be generally partitioned into two main categories: *specialized* and *declarative* approaches. Specialized algorithms, such as Apriori [Agrawal and Srikant, 1994] and FP-Growth [Han *et al.*, 2004], are generally a two-step process: (1) finding all frequent itemsets with adequate supports, and (2) generating ARs with high confidence by combining these frequent or large itemsets. Nevertheless, the main bottleneck of this line of research is that additional user-specified constraints, i.e., rules with an antecedent and/or consequent of a given size or rules with or without some specific items, often require new implementations to filter out the ARs violating or satisfying the user's constraints, which can be computationally infeasible for large databases.

Since there are usually various constraints over patterns to mine (i.e., support, confidence, etc.), new research have began connecting data mining to symbolic Artificial Intelligence (AI). Such fertilization leads to a number of algorithms that have been proposed within Constraints Programming (CP), Satisfiability (SAT), and Answer Set Programming (ASP) for mining sequences [Jabbour *et al.*, 2013; Négrevergne and Guns, 2015; Gebser *et al.*, 2016], frequent itemsets [Henriques *et al.*, 2012; Lazaar *et al.*, 2016; Schaus *et al.*, 2017], association rules [Boudane *et al.*, 2016; Belaid *et al.*, 2019], clustering [Dao *et al.*, 2017; Boudane *et al.*, 2017a], and overlapping communities in networks [Jabbour *et al.*, 2017; Ganji *et al.*, 2017; Jabbour *et al.*, 2020], etc. The main advantage of symbolic AI approaches for pattern mining is their declarativity and flexibility, which include the ability to incorporate new user-specified constraints without the need to modify the underlying system. To the best of our knowledge, earlier work that deal with ARs mining are that of [Boudane *et al.*, 2016] and [Belaid *et al.*, 2019]. In the first one, the authors developed a SAT-based approach to discover closed and indirect ARs. More specifically, they used propositional satisfiability with different linear inequalities to ensure the minimum frequency and the confidence of rules. Encoding ARs mining into SAT enables to take advantage of the recent and impressive progress achieved in SAT solving. Unfortunately, the counterpart of this method is the big encoding making such approach far from to be

scaling. The proposed approach of [Belaid *et al.*, 2019] presents a CP model to mine ARs and MNRs from transactional databases. The authors extended the global constraint *coversize* [Schaus *et al.*, 2017] by adding a new one called *confidence* to ensure the confidence of the rules. However, these two constraints cannot been expressed within existing global constraints. Hence, such extension generally borrowed techniques from specialized algorithms (to define propagators over the coversize and confidence constraints) which is far from the hoped-for declarativity.

In this paper, in order to break through these barriers and to scale the symbolic AI based approaches to larger databases, a declarative method for mining ARs and MNRs is first proposed using a SAT representation. At the core of our approach is a novel decomposition method that allows to split the original database into subsets that can be mined independently. Hopefully, without causing too large formulas and the associated memory problems, the advantage of our SAT encoding scheme is to allow independent evaluation on a multi-core machine, improving performance. We stress the fact that the introduced partitioning scheme in this paper is independent of any particular solver and widely applicable. Finally, a last contribution of this work consists of an empirical evaluation of the decomposition-based method, showing significant computational benefits. Specifically, our approach outperforms the CP-based algorithm CP4AR while significantly reducing the performance gap with the specialized system ECLAT-Z.

2 Formal Preliminaries

2.1 Propositional Logic and SAT problem

We consider a propositional language $\mathcal{L}$ defined from a finite set of propositional variables $Var = \{p, q, r, \ldots\}$, the logical constants $\bot$, $\top$, and the usual logical connectives (namely, $\neg$, $\wedge$, $\vee$, $\rightarrow$, and $\leftrightarrow$). Propositional formulas will be denoted by Greek letters Σ, Δ, etc. $Var(\Sigma)$ denotes the set of propositional variables appearing in the formula Σ. It is common for logical reasoning algorithms to operate on *normal form* representations instead of arbitrary sentences. A formula in *conjunctive normal form* (CNF) is a conjunction ($\wedge$) of clauses, where a *clause* is a disjunction ($\vee$) of literals. A *literal* is a propositional variable (p) or its negation ($\neg p$). In addition, a *Boolean interpretation* (or world) μ of a formula $\Sigma \in \mathcal{L}$ is a total function from $Var(\Sigma)$ to $\{0, 1\}$ (0 corresponds to $false$ and 1 to $true$). We denote by $\Sigma_{|x}$ the formula Σ where x assigned *true*, i.e., $\Sigma \wedge x$. μ is a *model* of Σ iff it makes it true in the usual truth functional way. Then, Σ is *satisfiable* if there exists a model of Σ. $models(\Sigma)$ denotes the set of models of a formula Σ. Lastly, SAT is the NP-complete problem that consists in deciding whether a given CNF formula is satisfiable or not.

2.2 A Glimpse at ARs Mining

Let Ω denote a universe of items (or symbols), called alphabet. We use the letters a, b, c, etc. to range over the elements of Ω. An *itemset* I over Ω is defined as a subset of Ω, i.e., $I \subseteq \Omega$. We denote by 2^Ω the set of all itemsets over Ω and we use the capital letters I, J, K, etc. to range over the elements of 2^Ω. Typically, a *transaction* is an ordered pair (i, I) where $1 \leq i \leq m$, called the *transaction identifier*, and I an itemset, i.e., $(i, I) \in \mathbb{N} \times 2^\Omega$. A *transaction database* $\mathcal{D}$ is defined as a finite non empty set of transactions ($\mathcal{D} \subseteq \mathbb{N} \times 2^\Omega$) where each transaction identifier refers to a unique itemset. Given a transaction database $\mathcal{D}$ and an itemset I, the *cover* of I in $\mathcal{D}$, denoted $\mathcal{C}(I, \mathcal{D})$, is defined as follows: $\{i \in \mathbb{N} \mid (i, J) \in \mathcal{D} \; and \; I \subseteq J\}$. The *support* of I in the database $\mathcal{D}$, denoted as $Supp(I, \mathcal{D})$, is defined as the cardinality of $\mathcal{C}(I, \mathcal{D})$, i.e., $Supp(I, \mathcal{D}) = |\mathcal{C}(I, \mathcal{D})|$. An itemset $I \subseteq \Omega$ such that $Supp(I, \mathcal{D}) \geq 1$ is *closed* iff, for all itemsets J with $I \subset J$, $Supp(J, \mathcal{D}) < Supp(I, \mathcal{D})$.

An *association rule* is a pattern of the form $X \rightarrow Y$ where X and Y are two disjoint itemsets. X and Y are called the *antecedent* and the *consequent* of the rule, respectively. In addition, the interestingness of ARs is defined through the notions of support and confidence. The *support* of an AR $X \rightarrow Y$ in a transaction database $\mathcal{D}$, defined as $Supp(X \rightarrow Y, \mathcal{D}) = \frac{Supp(X \cup Y, \mathcal{D})}{|\mathcal{D}|}$, determines how often a rule is applicable to a given dataset, i.e., the occurrence frequency of the rule. The *confidence* of $X \rightarrow Y$ in $\mathcal{D}$, defined as $\mathcal{C}onf(X \rightarrow Y, \mathcal{D}) = \frac{Supp(X \cup Y, \mathcal{D})}{Supp(X, \mathcal{D})}$, provides an estimate of the conditional probability of Y given X. Then, a *valid* AR is an AR with support and confidence greater than or equal to the minimum support α and minimum confidence β thresholds, respectively. More specifically, given a transaction database $\mathcal{D}$, a minimum user-specified support α and confidence β thresholds, the problem of mining association rules consists in computing the following set: $\{X \rightarrow Y \mid X, Y \subseteq \Omega \wedge Supp(X \rightarrow Y, \mathcal{D}) \geq \alpha \wedge \mathcal{C}onf(X \rightarrow Y, \mathcal{D}) \geq \beta\}$.

It is well-recognized that the main factor that hinders the applications of ARs is the huge number of ARs returned by the mining process. Typically, for reasonable thresholds α and β, the number of ARs can reach impractical amounts, such that analyzing the rules themselves becomes a challenging task. Moreover, many of these rules have no value to the user since they can be considered redundant. In this work, we are also interested in the well-known *minimal non-redundant* association rules variant, namely MNRs [Kryszkiewicz, 1998; Bastide *et al.*, 2000].

Definition 1. *Let $\mathcal{D}$ be a transaction database. Then, an AR $X \rightarrow Y$ is a MNR iff there is no AR $X' \rightarrow Y'$ in $\mathcal{D}$ different from $X \rightarrow Y$ s.t. (i) $Supp(X \rightarrow Y, \mathcal{D}) = Supp(X' \rightarrow Y', \mathcal{D})$, (ii) $Conf(X \rightarrow Y, \mathcal{D}) = Conf(X' \rightarrow Y', \mathcal{D})$, and (iii) $X' \subseteq X$ and $Y \subseteq Y'$.*

3 SAT Encoding Scheme for ARs Mining

Here, we review the SAT encoding scheme for the problem of mining ARs proposed in [Boudane *et al.*, 2016]. Basically, to encode the ARs mining problem into SAT one must define a set of variables and a set of constraints on the variables. Then, the idea was to build a one-to-one mapping between the set of ARs and the models of the corresponding CNF formula. More precisely, different variables are used to represent the covers of the two parts of the rule $X \rightarrow Y$, namely, X and $X \cup Y$. These variables are used in 0/1 linear inequalities to ensure the support and the confidence of the rules.

Given a transaction database $\mathcal{D} = \{(1, I_1), \ldots, (m, I_m)\}$, a minimum support α and confidence β thresholds. To represent the itemsets X and Y of each candidate rule $X \rightarrow Y$, two propositional variables x_a and y_a are associated to each item a in order to guarantee if a belongs to X (i.e., $x_a = true$) or Y (i.e., $y_a = true$). For the covers of X and $X \cup Y$, new variables p_i and q_i are also introduced to each transaction identifier $i \in \{1 \ldots m\}$. In addition, the following constraints (see Figure 1) are introduced on the variables to build a one-to-one mapping between the models of the obtained CNF formula, denoted as $\Sigma^{AR}_{\mathcal{D},\alpha,\beta}$, and the ARs. More precisely, Constraint **(1)** ensures the non-overlapping between X and Y. The formulas **(2)** and **(3)** capture the cover of X and $X \cup Y$, respectively. Moreover, the inequality constraints **(4)** and **(5)** express the support and the confidence constraints w.r.t. the user-specified thresholds α and β. Then, the ARs mining problem from $\mathcal{D}$ corresponds to the enumeration of the models of the CNF formula: $\Sigma^{AR}_{\mathcal{D},\alpha,\beta} = (\mathbf{1}) \wedge (\mathbf{2}) \wedge (\mathbf{3}) \wedge (\mathbf{4}) \wedge (\mathbf{5})$. Under a Boolean interpretation μ, the candidate rule is $X = \{a \in \Omega \mid \mu(x_a) = 1\} \rightarrow Y = \{b \in \Omega \mid \mu(y_b) = 1\}$ where $\mathcal{C}(X, \mathcal{D}) = \{i \in \mathbb{N} \mid \mu(p_i) = 1\}$ and $\mathcal{C}(X \cup Y, \mathcal{D}) = \{i \in \mathbb{N} \mid \mu(q_i) = 1\}$. Now, Constraint **(6)** ensures the closure of the itemset $X \cup Y$, i.e., if $\mathcal{C}(X \cup Y, \mathcal{D}) = \mathcal{C}(X \cup Y \cup \{a\}, \mathcal{D})$, then the item a has to belong to $X \cup Y$ (a is either in X or Y). We note here that Constraint (**6**) allows to add n clauses to the SAT encoding where n is the number of items.

To simplify the notation, in the sequel we use Σ instead of $\Sigma^{AR}_{\mathcal{D},\alpha,\beta}$ whenever it is clear from the context.

4 Our Approach

Encoding problems in CNF format and solving them with SAT solvers is indeed a competitive approach. SAT has the advantage of being very easy in its formulation. Nonetheless, the simplicity of the CNF format makes its use very restrictive. For instance, a constraint problem with a few dozen of variables may result in a SAT problem with thousands of variables and millions of clauses. Also, one may argue that a cause of inefficiency is the huge number of models of the CNF formula. Clearly, this is also the bottleneck for the SAT encoding scheme [Boudane *et al.*, 2016] for ARs mining problem. Indeed, in this encoding, the number of clauses is, in the worst case, $|\mathcal{D}| \times |\Omega|$.

In order to prune the search space, one can add more constraints to the previous model. For instance, Constraint (**7**) allows us to propagate the literal y_a when the number of transactions supporting such literal is less than α. Similarly, Constraint (**8**) allows us to propagate the literal q_i when the associated transaction has all items assigned *false*.

$$\bigwedge_{a \in \Omega} (y_a \rightarrow \sum_{T_i \in \mathcal{D} \mid a \in T_i} q_i \geq \alpha) \tag{7}$$

$$\bigwedge_{T \in \mathcal{D}} ((\bigwedge_{a \in T} \neg x_a) \rightarrow \neg q_i) \tag{8}$$

Unfortunately, these two cardinality constraints can outgrow the available memory and can consequently make SAT solving inefficient. To break through this barrier, in the sequel we propose to use a *decomposition* method that allows to split the search space into numerous but smaller sub-problems and easier to solve. Such decomposition based method allows us to avoid encoding the whole transaction database. Our main idea is as follows. First, given a CNF formula Σ and a variable $x_a \in Var(\Sigma)$, the models of Σ can be partitioned into those of $\Sigma \wedge x_a$ and $\Sigma \wedge \neg x_a$. By generalizing such principle, one can split a CNF formula into subformulas such that each subformula can be evaluated in an independent way.

Definition 2. *Let $\Sigma \in \mathcal{L}$ be a CNF formula. $\mathcal{P} = \{\Sigma \wedge \Gamma_1, \ldots, \Sigma \wedge \Gamma_n\}$ is a partition of Σ if $\Gamma = \{\Gamma_1, \ldots, \Gamma_n\}$ is a set of formulas defined over $Var(\Sigma)$ such that $\bigvee_{i=1}^{n} \Gamma_i \equiv \top$, and $\Gamma_i \wedge \Gamma_j \models \bot$, for all $1 \leq i \neq j \leq n$. $\Gamma = \{\Gamma_1, \ldots, \Gamma_n\}$ is called the Σ-partition.*

That is, by Definition 2, the set of models of Σ is equal to $\biguplus models(\Sigma \wedge \Gamma_i)$ $(1 \leq i \leq n)$. The above definition is a general definition that allows for a range of possible splitting techniques to be defined. In what follows, we consider an instantiation that consists in splitting Σ into its literal conjuncts w.r.t. a given ordering on the variables of Σ.

4.1 A Decomposition-based Encoding for ARs

Here, we present our decomposition method to extract ARs. Given an AR $X \rightarrow Y$, we select the variables encoding the antecedent $X = \{a_1, \ldots, a_n\}$ to define the partition $\mathcal{P}$. Let $\langle x_{a_1}, \ldots, x_{a_n} \rangle$ be an ordering on the set of variables encoding X. Then, the Σ-partition $\Gamma = \{\Gamma_1, \ldots, \Gamma_n\}$ over $\{x_{a_1}, \ldots, x_{a_n}\}$ is defined as: $\Gamma_i = x_{a_i} \wedge \bigwedge_{1 \leq k < i} \neg x_{a_k} (1 \leq i \leq n)$. Now, the main idea is to split recursively the formula Σ w.r.t. its positive and negative literals. Then, the formula $\Sigma \wedge \Gamma_i \equiv \Sigma_{|\ldots \neg x_{a_{i-1}}, x_{a_i}}$ enforces x_{a_i} to be *true*, that is the item a_i belongs to X. Consequently, the encoding can be restricted to transactions containing the item a_i. Also, the literal x_{a_k} for all $1 \leq k < i$ assigned *false* allows to exclude the item a_k to be in X. Clearly, this allows to avoid encoding the entire database and without causing too large formulas and the associated memory problems.

Clearly, the splitting of Σ leads to many independent subformulas encoding subsets of a specific list of transactions of the original database. However, some of such sub-problems might remain large, hence there is again a firm barrier to the scalability of the approach to larger databases. To illustrate, let us suppose that an item a_i appears in each transaction of the database $\mathcal{D}$. Then, the encoding of the cover of $X \cup Y$ requires considering all the transactions but excluding a_i.

To limit such blow up in terms of encoding size, we therefore opt to develop a different encoding based on a double splitting over X. More formally, the Σ-partition Γ in Definition 2 can be redefined as follows:

$$\begin{cases} \Gamma_{i,j} = x_{a_i} \wedge x_{a_j} \wedge \bigwedge_{k<j,\ k \neq i} \neg x_{a_k} & \text{if } i < j \\ \Gamma_{i,i} = x_{a_i} \wedge \bigwedge_{k \neq i} \neg x_{a_k} \end{cases}$$

Clearly, $\Gamma_{i,j}$ extends Γ_i by performing a second call to Γ_i at each branch. In other words, after the selection of an item a_i, we iterate the same reasoning over transactions involving a_i.

Our algorithm for ARs mining is illustrated in Algorithm 1, that we refer to as SATAR (**SAT** approach for **AR**s mining). Given a transaction database $\mathcal{D}$ and following an ordering over items of $\mathcal{D}$ at each iteration an item a is fixed to be

$$\bigwedge_{a\in\Omega}(\neg x_a \vee \neg y_a) \quad (1) \qquad \bigwedge_{i=1}^{m}(\neg p_i \leftrightarrow \bigvee_{a\in\Omega\setminus I_i} x_a) \quad (2) \qquad \bigwedge_{i=1}^{m}(\neg q_i \leftrightarrow \neg p_i \vee (\bigvee_{a\in\Omega\setminus I_i} y_a)) \quad (3)$$

$$\sum_{i=1}^{m} q_i \geq m \times \alpha \quad (4) \qquad 100 * \sum_{i=1}^{m} q_i - \beta * \sum_{i=1}^{m} p_i \geq 0 \quad (5) \qquad \bigwedge_{a\in\Omega}((\bigwedge_{i=1..m,\ a\notin I_i} \neg q_i) \rightarrow x_a \vee y_a) \quad (6)$$

Figure 1: SAT Encoding Scheme for ARs.

Algorithm 1: Decomposition-based ARs Mining

Data: A transaction database $\mathcal{D}$, α, β
Result: The set of all ARs R

$R \leftarrow \emptyset$, $\Gamma \leftarrow \emptyset$, $V = \langle a_1 \dots a_n \rangle \leftarrow items(\mathcal{D})$;
for $a \in V$ **do**
  $\mathcal{D}' \leftarrow \{T \in \mathcal{D} \mid a \in T\}$;
  $V' \leftarrow items(\mathcal{D}') \setminus \{a\}$;
  $\Gamma \leftarrow \Gamma \cup \{a\}$;
  for $a' \in V'$ **do**
    $\mathcal{D}'' \leftarrow \{T \in \mathcal{D}' \mid a' \in T\}$;
    $\Gamma \leftarrow \Gamma \cup \{a'\}$;
    $F = encode\_cnf(\Gamma, \mathcal{D}'')$; // $F = \Sigma \wedge \Gamma_{i,j}$
    $R = R \cup enumerate\_models(F)$;
    $\Gamma \leftarrow (\Gamma \setminus \{a'\}) \cup \{\neg a'\}$;
  end
  $F = encode\_cnf(\Gamma, \mathcal{D}')$; // $F = \Sigma \wedge \Gamma_{i,i}$
  $R = R \cup enumerate\_models(F)$;
  $\Gamma \leftarrow (\Gamma \setminus \{a\}) \cup \{\neg a\}$;
end
return R;

in X restricting the encoding to transactions containing a denoted $\mathcal{D}'$ over a subset of items $V' \subseteq V$. A second iteration is performed by fixing a second item a' to be in X leading to more restricted subset of transactions $\mathcal{D}''$. Then, the function *encode_cnf* is called over $\mathcal{D}''$ to encode the problem into a CNF formula. Finally, the enumeration of the models of the CNF formula is performed using the function *enumerate_models* incarnating a SAT-based enumeration solver.

4.2 A Decomposition-based Encoding for MNRs

Now, we extend our decomposition-based approach to enumerate MNRs. To do so, new constraints have to be added to the previous encoding. Then, we perform the same splitting technique to limit the overhead in terms of the huge number of clauses. First, let us recall that MNRs have been characterized through the closure [Taouil *et al.*, 2000] and the notion of minimal generator. In fact, a rule $X \rightarrow Y$ is non-redundant iff $X \cup Y$ is closed and X is a minimal generator. In other words, the MNRs are the closed rules in which the antecedents are minimal w.r.t. set inclusion. Using such property, in [Boudane *et al.*, 2017b] the authors provided a characterization of the antecedents of the MNRs, called *minimal generators*. Let us first introduce some useful notions.

Definition 3. *Let $\mathcal{D}$ be a transaction database and X a closed itemset in $\mathcal{D}$. Then, an itemset $X' \subseteq X$ is a minimal generator of X iff $Supp(X', \mathcal{D}) = Supp(X, \mathcal{D})$, and there is no $X'' \subseteq X$ s.t. $X'' \subset X'$ and $Supp(X'', \mathcal{D}) = Supp(X, \mathcal{D})$.*

Proposition 1. *[Boudane* et al.*, 2017b] Given a transaction database $\mathcal{D}$, the rule $X \rightarrow Y$ is a MNR in $\mathcal{D}$ iff $X \rightarrow Y$ is a closed AR, and $|X| = 1$ or, for all items $a \in X$, $Supp(X, \mathcal{D}) < Supp(X \setminus \{a\}, \mathcal{D})$.*

Proposition 1 characterizes minimal generators by providing constraints over its items. A direct translation of such characterization into constraints can be done as follows:

$$\Delta = \bigwedge_{a\in\Omega}(x_a \rightarrow \bigvee_{(T\in\mathcal{D}\ \mid\ a\notin T)}(\sum_{b\notin T} x_b \leq 1)) \quad (9)$$

That is, Constraint (**9**) enforces that if a belongs to X then there exists a transaction T such that $X \not\subset T$ and $X \setminus \{a\} \subset T$. Hence, the MNRs correspond to the models of the CNF formula $\Sigma \wedge \Delta \wedge \Theta$ where Θ is the closure constraint (**6**).

Unfortunately, Constraint (**9**) leads to too large formulas. In particular, the inequality constraint $\sum_{b\notin T} x_b \leq 1$ has to be encoded for each transaction in the database. In the worst case, minimal generator constraints require a large number of clauses making the encoding of MNRs ever, large.

To tackle this problem, we consider the same splitting principle that leads to independent sub-problems of the form $\Sigma \wedge \Delta \wedge \Theta \wedge \Gamma_{i,j}$. The main difference with the enumeration of ARs lies in the fact that the minimal generator constraint depends on the whole transaction database. In fact, a minimal generator of the database $\mathcal{D}$ restricted to transactions with a_i and a_j is not necessarily a minimal generator of $\mathcal{D}$.

Next, let us show how the formula Δ is simplified by $\Gamma_{i,j}$ when $i < j$. The transaction database $\mathcal{D}$ can be partitioned into $\mathcal{D}_1$, $\mathcal{D}_2$, $\mathcal{D}_3$, and $\mathcal{D}_4$ such that $\mathcal{D}_1$ contains a_i and not a_j, $\mathcal{D}_2$ contains a_j and not a_i, $\mathcal{D}_3$ contains both a_i and a_j, and $\mathcal{D}_4$ does not contain neither a_i nor a_j. Under $\Gamma_{i,j}$, Δ can be rewritten as the conjunction of Δ over the items of $\mathcal{D}_3$ as:

$$\begin{aligned}\Delta \wedge \Gamma_{i,j} = \quad & (x_{a_i} \rightarrow \bigvee_{T\in\mathcal{D}_2}(\sum_{b\notin T} x_b \leq 1)) \ \wedge \\ & (x_{a_j} \rightarrow \bigvee_{T\in\mathcal{D}_1}(\sum_{b\notin T} x_b \leq 1)) \ \wedge \\ & (\bigwedge_{a_k\in\mathcal{D}_3,\ k\neq i,\ k\neq j}(x_{a_k} \rightarrow \bigvee_{T\in\mathcal{D}_3}(\sum_{b\notin T} x_b \leq 1)))\end{aligned}$$

More importantly, $\mathcal{D}_4$ is not used in Δ under $\Gamma_{i,j}$ assumption where $i < j$ thanks to the double splitting over X, i.e., none of a_i and a_j are present in the transactions of $\mathcal{D}_4$. Consequently, for all $T \in \mathcal{D}_4$, $\sum_{b\notin T} x_b \leq 1$ is already falsified by two items. Moreover, $\Delta \wedge \Gamma_{i,i} \equiv \top$ since X is singleton and thus is a minimal generator.

For the closure constraint Θ, we proceed by simplifying the constraint (**6**) under $\Gamma_{i,j}$ leading to:

$$\Theta \wedge \Gamma_{i,j} = \bigwedge_{c\in\mathcal{D}_3}((\bigwedge_{T\in\mathcal{D}_3,\ c\notin T} \neg q_i) \rightarrow x_c \vee y_c)$$

5 Empirical Investigation

We now present the experiments carried out to evaluate the performance of our approach. In particular, we study the running time for discovering ARs and MNRs in sequential and parallel setting. For baseline comparison, we retain the CP-based algorithm CP4AR [Belaid *et al.*, 2019] and the specialized method ECLAT-Z [Szathmary *et al.*, 2008], which in turn is shown more competitive than all existing approaches.

Algorithm 1 is implemented in C++ top-on the SAT solver MiniSAT [Eén and Sörensson, 2002], which is changed to enumerate all models by performing a DPLL procedure [Davis *et al.*, 1962] without restart. We use the API *OpenMP* that supports multi-platform shared memory multiprocessing programming in C/C++. The partition is performed by considering the items frequencies in increasing order.

Our experiments were performed on a machine with Intel Xeon quad-core processors with 32GB of RAM running at 2.66 GHz on Linux CentOS. Time-out was set to 1800 seconds and memory-out to 10 GB in all runs. We consider the datasets used in [Belaid *et al.*, 2019] coming from FIMI[1] and CP4IM[2] repositories. The minimum confidence threshold β is fixed to 75%[3] and different minimum support values are chosen w.r.t. the size of each dataset. All experiments are available at: https://github.com/crillab/satar-xp.

We perform two kinds of experiments. In the first evaluation, we carry out a comparison to enumerate both ARs and MNRs. In the second, we perform a parallel evaluation to mining ARs and MNRs by varying the number of cores.

Sequential Evaluation. Table 1 reports comparative results of SATAR against CP4AR and ECLAT-Z, using different values of α (in percent). Table 1 contains both results for ARs and MNRs. It reports the number of models (i.e., patterns) and the total CPU time (in seconds) for each dataset with different values of α. "TO" (resp. "MO") is shown when time limit is exceeded (resp. when the memory limit is reached). We also use the symbol "–" to indicate that the method is not able to scale on the number of patterns under the time limit. As one can observe, for ARs mining, SATAR outperforms CP4AR on almost all tested datasets. Moreover, for some datasets e.g., *connect* and *pumsb*, the gain is impressive. Interestingly enough, our approach overpasses the specialized algorithm ECLAT-Z, e.g., *anneal*, *chess* and *mushroom* datasets. Remarkably, for the largest dataset *kosarak*, our approach is able to enumerate all ARs while ECLAT-Z and CP4AR fail for all the considered values of α. In total, on 19/33 SATAR provides the best time when ECLAT-Z ranked first on 13/33. CP4AR is not able to provide the best time on any test.

For MNRs mining, we have similar observations as for ARs. SATAR and ECLAT-Z show better results than CP4AR. Indeed, SATAR wins on 20 from 33 datasets, followed by ECLAT-Z with a score of 10, whereas CP4AR has only 2 datasets. Furthermore, on some large datasets SATAR outperforms ECLAT-Z on MNRs, e.g., *pumsb* and *T10I4D100K* with $\alpha = 75\%$ and 0.02%, respectively.

[1] http://fimi.ua.ac.be/data/

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

[3] Similar results were observed when β is set to 50% and 90%.

Parallel Evaluation. The decomposition technique defined in this work allows to generate independent problems that can be handled on a multi-core machine. Hence, we extended SATAR algorithm for multi-core solving. The different sub-formulas $\Sigma \wedge \Gamma_{i,j}$ are then distributed over the used cores to be solved. The sub-problems are fairly assigned to cores. We perform similar experiments by considering 1,2 and 4 cores and varying the minimum support threshold values. Figure 2 shows the solving time on a representative sample of datasets. As expected, our parallel approach allows us to considerably reduce the computing time. For instance, with a single core, the *T10I4D100K* dataset is not solved for $\alpha = 0.01\%$ for MNRs, while this task is achieved when the number of cores is greater than 1. Moreover, the time needed to extract all MNRs from *pumsb* dataset with $\alpha = 75\%$ is reduced from 950 seconds with a single core to less than 400 seconds with 4 cores. Clearly, the runtime gain increases with the number of cores. As one can observe in Figure 2, this gain is more important when the number of cores passes from 1 to 2 cores rather than from 2 to 4.

Let us stress that our decomposition is performed before the solving process rather than in a dynamic way during the search. Consequently, the time required by cores to solve their assigned sub-problems is different in general. In the sequel, we analyze the load unbalancing of our approach.

Load Balancing. To assess the suitability of our load balancing strategy, we provide an empirical analysis of the CPU time with different number of cores on a sample of datasets. We limit our analysis to ARs mining. Similar observations are made on MNRs. We fix the number of cores to 4. In Figure 3, for each value of α we report the average time over cores, the minimum and maximum time to quantify the idleness of some cores. The tighter the difference between minimum and maximum time is, the better the load balancing is.

As we can observe, our approach allows to limit the relative load unbalancing, i.e., all the cores spent almost the same time to solve their assigned sub-problems. This is more impressive for the dataset *T10I4D100K*. Overall, our decomposition strategy allows to balance correctly the search between the different cores. Last, it allows to tackle large datasets e.g., *kosarak* and *T10I4D100K*, proving the efficiency and scalability of our SATAR approach.

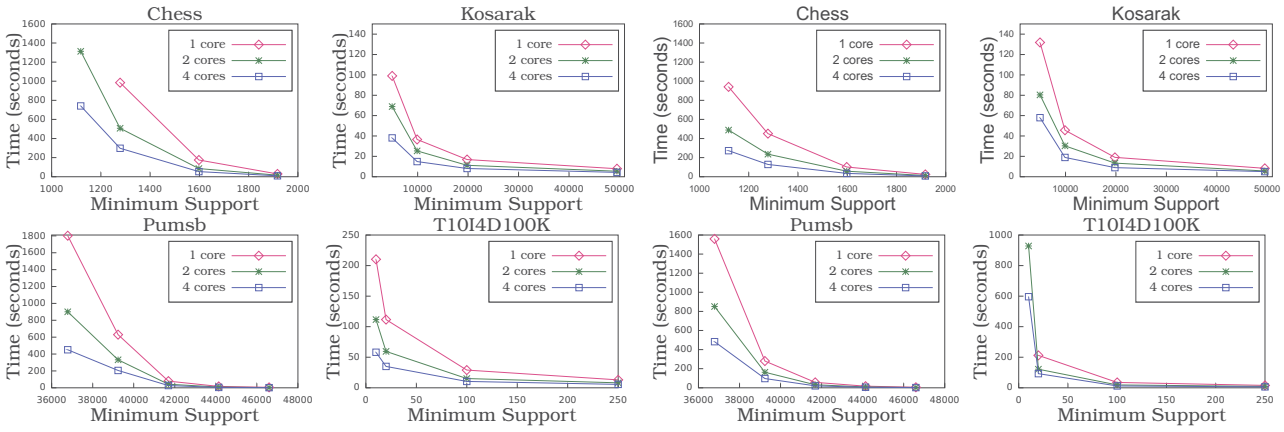
6 Conclusion

In this paper, we have proposed a novel SAT based approach to mine (minimal non-redundant) ARs from large transaction databases. Our approach relies on a decomposition strategy that allows us to achieve efficiency by suitably lowering the size of the sub-problems without jeopardizing completeness. An extensive campaign of experiments conducted over different real-world datasets has shown that our approach scales well and outperforms the CP based approach CP4AR while competed with the specialized approach ECLAT-Z.

Different research directions can improve this work. First, our decomposition approach can be performed dynamically in order to limit the load unbalancing. We are also interested in studying better ordering among variables to improve our decomposition technique.

Instance	$\mid \mathcal{D} \mid, \mid \Omega \mid$	α	ARs				MNRs			
			ECLAT-Z	CP4AR	SATAR	#Models	ECLAT-Z	CP4AR	SATAR	#Models
anneal	812, 89	70%	TO	TO	**1166.29**	48151597	243.58	8.08	**1.69**	8638086189
		80%	104.27	336.06	**16.00**	111119768	10.07	4.47	**0.39**	78954
		90%	1.11	4.49	**0.10**	187590	0.80	2.73	**0.10**	2557
chess	3196, 75	40%	TO	TO	**982.46**	3617072999	TO	TO	**453.56**	346164794
		50%	466.60	TO	**166.96**	477566469	TO	563.36	**102.05**	83324019
		60%	43.73	296.68	**28.50**	62457693	882.45	126.53	**21.18**	17970343
connect	67557, 129	85%	**29.65**	TO	100.37	43896880	107.11	113.71	**60.11**	1349555
		90%	**5.75**	175.59	26.09	3640704	**12.77**	33.62	24.03	319352
		95%	**2.47**	11.84	8.59	78656	**2.55**	11.83	10.23	25549
kosarak	990002, 41267	0.5%	MO	MO	**96.85**	2865	MO	MO	**125.18**	2865
		2%	MO	MO	**16.44**	116	MO	MO	**18.53**	116
		5%	MO	MO	**6.62**	30	MO	MO	**8.13**	30
mushroom	8124, 112	5%	279.27	529.54	**36.49**	129322936	31.19	**6.45**	22.46	55371
		10%	21.35	79.55	**8.59**	17406340	6.85	**5.69**	7.05	22756
		20%	3.67	11.60	**2.29**	1373250	**2.32**	4.20	2.38	5262
pumsb	49046, 7117	75%	**700.81**	TO	TO	351194878	TO	TO	**1578.32**	39406675
		80%	**53.95**	1738.93	639.80	28276846	**126.87**	526.54	286.11	5504722
		85%	**8.67**	113.11	77.07	1408950	**12.31**	72.81	58.04	580651
retail	88162, 16470	0.1%	**9.95**	67.77	63.47	1425	**23.33**	194.73	102.50	1412
		0.3%	**8.79**	12.02	11.89	244	**8.85**	24.23	16.51	244
		0.5%	8.22	10.00	**4.97**	117	7.98	16.81	**6.01**	117
T10I4D100K	100000, 870	0.02%	**21.86**	69.43	109.35	1752672	1333.98	372.72	**217.59**	516743
		0.1%	**4.82**	18.02	26.29	296921	42.27	36.54	**34.37**	280831
		0.5%	**1.62**	8.27	5.28	964	**2.03**	13.04	5.67	964
T40I10D100K	100000, 942	0.3%	TO	TO	TO	–	TO	TO	TO	444178969
		1.0%	**9.95**	189.36	239.74	3545811	**125.28**	319.74	357.84	3545811
		1.6%	**3.48**	15.35	77.73	1080	**6.26**	31.81	101.16	1080
vote	435, 48	5%	13.03	26.96	**5.59**	8455184	55.96	13.36	**10.35**	2520334
		10%	4.70	8.64	**1.48**	1875960	15.52	8.97	**2.47**	1288014
		30%	0.42	2.84	**0.00**	3836	0.44	2.51	**0.00**	3836
zoo	101, 36	5%	55.04	143.48	**8.80**	60865572	12.99	4.36	**0.16**	40804
		30%	1.53	3.91	**0.00**	387113	1.18	2.71	**0.00**	6462
		50%	0.13	1.83	**0.00**	867	0.32	1.86	**0.00**	548

Table 1: Discovering Association Rules: ECLAT-Z vs. CP4AR vs. SATAR



(a) ARs (b) MNRs

Figure 2: Performance gain w.r.t. the number of cores.

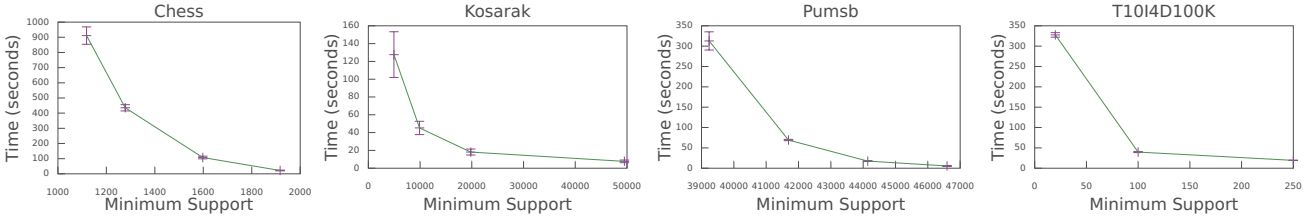


Figure 3: Load unbalancing between cores.

References

[Agrawal and Srikant, 1994] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In *VLDB*, pages 487–499, 1994.

[Bastide *et al.*, 2000] Yves Bastide, Nicolas Pasquier, Rafik Taouil, Gerd Stumme, and Lotfi Lakhal. Mining minimal non-redundant association rules using frequent closed itemsets. In *Computational Logic*, volume 1861, pages 972–986, 2000.

[Belaid *et al.*, 2019] Mohamed-Bachir Belaid, Christian Bessiere, and Nadjib Lazaar. Constraint programming for association rules. In *SDM*, pages 127–135, 2019.

[Boudane *et al.*, 2016] Abdelhamid Boudane, Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. A SAT-based approach for mining association rules. In *IJCAI*, pages 2472–2478, 2016.

[Boudane *et al.*, 2017a] Abdelhamid Boudane, Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. Clustering complex data represented as propositional formulas. In *PAKDD*, pages 441–452, 2017.

[Boudane *et al.*, 2017b] Abdelhamid Boudane, Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. Enumerating non-redundant association rules using satisfiability. In *PAKDD*, pages 824–836, 2017.

[Ceglar and Roddick, 2006] Aaron Ceglar and John F. Roddick. Association mining. *ACM Comput. Surv.*, 38(2):5, 2006.

[Dao *et al.*, 2017] Thi-Bich-Hanh Dao, Khanh-Chuong Duong, and Christel Vrain. Constrained clustering by constraint programming. *Artif. Intell.*, 244:70–94, 2017.

[Davis *et al.*, 1962] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem proving. *Communications of the ACM*, 5:394–397, 1962.

[Eén and Sörensson, 2002] Niklas Eén and Niklas Sörensson. An extensible sat-solver. In *SAT*, pages 502–518, 2002.

[Ganji *et al.*, 2017] Mohadeseh Ganji, James Bailey, and Peter J. Stuckey. A declarative approach to constrained community detection. In *CP*, pages 477–494, 2017.

[Gebser *et al.*, 2016] Martin Gebser, Thomas Guyet, René Quiniou, Javier Romero, and Torsten Schaub. Knowledge-based sequence mining with ASP. In *IJCAI*, pages 1497–1504, 2016.

[Han *et al.*, 2004] Jiawei Han, Jian Pei, Yiwen Yin, and Runying Mao. Mining frequent patterns without candidate generation: A frequent-pattern tree approach. *Data Min. Knowl. Discov.*, 8(1):53–87, 2004.

[Henriques *et al.*, 2012] Rui Henriques, Inês Lynce, and Vasco M. Manquinho. On when and how to use SAT to mine frequent itemsets. *CoRR*, abs/1207.6253, 2012.

[Jabbour *et al.*, 2013] Saïd Jabbour, Lakhdar Sais, and Yakoub Salhi. The top-k frequent closed itemset mining using top-k sat problem. In *ECML/PKDD*, pages 403–418, 2013.

[Jabbour *et al.*, 2017] Saïd Jabbour, Nizar Mhadhbi, Badran Raddaoui, and Lakhdar Sais. A SAT-based framework for overlapping community detection in networks. In *PAKDD*, pages 786–798, 2017.

[Jabbour *et al.*, 2020] Said Jabbour, Nizar Mhadhbi, Badran Raddaoui, and Lakhdar Sais. SAT-based models for overlapping community detection in networks. *Computing*, 102(5):1275–1299, 2020.

[Kryszkiewicz, 1998] Marzena Kryszkiewicz. Representative association rules and minimum condition maximum consequence association rules. In *PKDD*, pages 361–369, 1998.

[Lazaar *et al.*, 2016] Nadjib Lazaar, Yahia Lebbah, Samir Loudni, Mehdi Maamar, Valentin Lemière, Christian Bessiere, and Patrice Boizumault. A global constraint for closed frequent pattern mining. In *CP*, pages 333–349, 2016.

[Négrevergne and Guns, 2015] Benjamin Négrevergne and Tias Guns. Constraint-based sequence mining using constraint programming. In *CPAIOR*, pages 288–305, 2015.

[Schaus *et al.*, 2017] Pierre Schaus, John O. R. Aoga, and Tias Guns. Coversize: A global constraint for frequency-based itemset mining. In *CP*, pages 529–546, 2017.

[Szathmary *et al.*, 2008] Laszlo Szathmary, Petko Valtchev, Amedeo Napoli, and Robert Godin. An efficient hybrid algorithm for mining frequent closures and generators. In *CLA*, page 47–58, 2008.

[Taouil *et al.*, 2000] Rafik Taouil, Nicolas Pasquier, Yves Bastide, and Lotfi Lakhal. Mining bases for association rules using closed sets. In *Proceedings of ICDE*, page 307, 2000.