Symmetry Breaking in Itemset Mining

Belaïd Benhamou [1,2], Saïd Jabbour[2], Lakhdar Sais[2] and Yacoub Salhi[2]
[1]*Aix-Marseille Université, Laboratoire des Sciences de l'information et des Systèmes (LSIS), Domaine Universitaire de Saint Jérôme, Avenue Escadrille Normandie Niemen, 13397 Marseille Cedex 20, France*
[2]*Centre de Recherche en Informatique de Lens (CRIL), Université d'Artois, Rue Jean Souvenir, SP 18 F 62307 Lens Cedex, France*

Keywords: Data Mining, Itemset Mining, Symmetry, Satisfiability, Constraint Programming.

Abstract: The concept of symmetry has been extensively studied in the field of constraint programming and in propositional satisfiability. Several methods for detection and removal of these symmetries have been developed, and their integration in known solvers of these domain improved dramatically their effectiveness on a large variety of problems considered difficult to solve. The concept of symmetry may be exported to other domains where some structures can be exploited effectively. Particularly in data mining where some tasks can be expressed as constraints. In this paper, we are interested in the detection and elimination of symmetries in the problem of finding frequent itemsets of a transaction database and its variants. Recent works have provided effective encodings as Boolean constraints for these data mining tasks and some recent works on symmetry detection and elimination in itemset mining problems have been proposed. In this work we propose a generic framework that could be used to eliminate symmetries for data mining task expressed in a declarative constraint language. We show how symmetries between the items of the transactions are detected and eliminated by adding symmetry-breaking predicate (SBP) to the Boolean encoding of the data mining task.

1 INTRODUCTION

In this paper, we investigate the notion of symmetry elimination in Frequent Itemset Mining (FIM) (Agrawal et al., 1993). The itemset mining problem has several applications and remains central in the Data mining research field. The most known example is the one considered by large retail organizations called *basket data*. A record of such data contains essentially the customer identification, the transaction date and the items bought by the customer. Advances in bar-codes technology, the use of credit cards of frequent-customer card make it now possible to collect and store a great amounts of sale data. It is then important for the retail firms to know the set of items that are frequently bought by customers. This is the frequent itemset mining problem. Since its introduction in 1993 (Agrawal et al., 1993), several highly scalable algorithms are introduced ((Agrawal and Srikant, 1994), (Han et al., 2000), (Zaki and Hsiao, 2005),(Uno et al., 2003) (Uno et al., 2004), (Burdick et al., 2001),(Grahne and Zhu, 2005), (Minato et al., 2007)) to enumerate the sets of frequent items. The two challenging questions investigated in such algorithms are: in one hand how to compute all the frequent itemsets in a reasonable CPU time and in the other hand how to compact the output and reduce its size when there is a huge number of frequent itemsets. Many other data mining tasks exist, such as the association rule mining, the frequent pattern, clustering and episode mining, but almost all of them are closely in relationship to itemset mining which looks to be the canonical problem. A lot of efficient and scalable algorithms are developed for target and specific mining tasks. As stated in (Tiwari et al., 2010), different methods for the itemset mining are provided. Mainly they differ from each other in the way they explore the search space, the data structure they use, the exploitation of the anti-monotonicity property. The other important point is the size of the output of such algorithms. Some solutions are found, for instance one can enumerate only the closed, the maximal, the condensed, the preferred, or discriminative itemsets instead of all the frequent itemsets.

Data mining community introduced the *constraint-based mining* framework in order to specify in terms of constraints the properties of the patterns to be mined ((Bonchi and Lucchese, 2007),(Bucilă et al., 2003), (Pei et al., 2004), (Besson et al., 2010)). A wide variety of constraints are

Benhamou B., Jabbour S., Sais L. and Salhi Y..
Symmetry Breaking in Itemset Mining.
DOI: 10.5220/0005078200860096
In *Proceedings of the International Conference on Knowledge Discovery and Information Retrieval* (KDIR-2014), pages 86-96
ISBN: 978-989-758-048-2

successfully integrated and implemented in different specific data mining algorithms.

Recently De Raedt et al. ((Raedt et al., 2008; Guns et al., 2011a)) introduced the alternative of using constraint programming in data mining. They showed that a such alternative can be efficiently applied for a wide range of pattern mining problems. Most of the pattern mining constraint (e.g. frequency, closeness, maximality, and anti-monotonicity constraints) had been expressed in a declarative constraint programming language. The data mining problem is modeled as a constraint satisfaction problem (CSP) and a solver (e.g. Gecode) is then used to enumerate solutions corresponding to the set of interesting patterns. A strength point here is that different constraints can be combined without the need to modify the solver, unlike in the existing specific data mining algorithms. Since the introduction of this declarative approach, there is a growing interest in finding generic and declarative approaches to model and solve data mining tasks. For instance, several works expressed data mining problems as propositional satisfiability ((Jabbour et al., 2013c), (Henriques et al., 2012), (Métivier et al., 2012), (Khiari et al., 2010), (Raedt et al., 2010), (Jabbour et al., 2013b)) and used efficient modern SAT solvers as black-box to solve them. More recently, a constraint declarative framework for solving Data mining tasks called MiningZinc (Guns et al., 2013), had been introduced.

On the other hand, symmetry is by definition a multidisciplinary concept. It appears in many fields ranging from mathematics to Artificial Intelligence, chemistry and physics. It reveals different forms and uses, even inside the same field. In general, it returns to a transformation, which leaves invariant (does not modify its fundamental structure and/or its properties) an object (a figure, a molecule, a physical system, a formula or a constraints network...). For instance, rotating a chessboard up to 180 degrees gives a board that is indistinguishable from the original one. Symmetry is a fundamental property that can be used to study these various objects, to finely analyze these complex systems or to reduce the computational complexity when dealing with combinatorial problems.

As far as we know, the principle of symmetry has been first introduced by Krishnamurthy (Krishnamurty, 1985) to improve resolution in propositional logic. Symmetries for Boolean constraints are studied in depth in (Benhamou and Sais, 1992a; Benhamou and Sais, 1994a). The authors showed how to detect them and proved that their exploitation is a real improvement for several automated deduction algorithms efficiency. Since that, many research works on symmetry appeared. For instance, the static approach used by James Crawford et al. in (Crawford et al., 1996) for propositional logic theories consists in adding constraints expressing global symmetry of the problem. This technique has been improved in (Aloul et al., 2003b) and extended to 0-1 Integer Logic Programming in (Aloul et al., 2004). The notion of interchangeability in Constraint Satisfaction Problems (CSPs) is introduced in (Freuder, 1991) and symmetry for CSPs is studied earlier in (Puget, 1993; Benhamou, 1994).

In the context of constraint programming, Guns et al. (Guns et al., 2011b) used symmetry breaking constraints to impose a strict ordering on the patterns in k-pattern set mining. More recently, symmetry detection and elimination are integrated in itemset mining problems (Jabbour et al., 2012; Jabbour et al., 2013a). Two different approaches are proposed. In the first one, symmetries are eliminated by rewriting the transaction database (eliminating items), while in the second approach the authors integrate symmetry elimination in Apriori-like algorithms. For other previous studies on symmetries in data mining, we refer the reader to the related work section.

The work that we investigate in this paper, goes in this direction. It consists in detecting and eliminating symmetries in the itemset mining problem expressed as a Boolean satisfiability. We will show how *global symmetries*[1] of the given transaction database are detected and expressed in terms of symmetry breaking predicates. Such predicates are added to the boolean encoding of the itemset mining problem in a preprocessing step and a SAT solver is used as a black box to enumerate the non-symmetrical solutions (the non-symmetrical frequent itemsets). In most of the data mining tasks, we usually need to enumerate interesting patterns and this usually lead to a output of huge size. Eliminating symmetries might reduce the size of the output and lead to discover the non-symmetrical patterns which are the most important and representative of the knowledge.

The rest of the paper is organized as follows. In Section 2, we give some necessary background on the satisfiability problem, permutations and the necessary notion on itemset mining problem. We study the notion of symmetry in itemset mining represented as boolean constraints in Section 3. In Section 4 we show how symmetries can be detected by means of graph automorphism. We show in section 5 how this symmetry can be eliminated by adding symmetry breaking predicates to the Boolean encoding. Section 6 gives experiments on different data-sets to show the advantage of using symmetries in itemset mining.

[1]Symmetries that are present in the initial formulation of the problem

Section 7 investigates the related works and Section 8 concludes the work.

2 BACKGROUND

We summarize in this section some background on the satisfiability problem, permutations, and itemset mining problem.

2.1 Propositional Satisfiability (SAT)

We shall assume that the reader is familiar with propositional logic. We give here, a short description. Let V be the set of propositional variables called only variables. Variables will be distinguished from literals, which are variables with an assigned parity 1 or 0 that means *True* or *False*, respectively. This distinction will be ignored whenever it is convenient, but not confusing. For a propositional variable p, there are two literals: p the positive literal and $\neg p$ the negative one.

A clause is a disjunction of literals such that no literal appears more than once, nor a literal and its negation at the same time. This clause is denoted by $p_1 \vee p_2 \vee \ldots \vee p_n$. A formula $\mathcal{F}$ in conjunctive normal form (CNF) is a conjunction of clauses.

A truth assignment to a CNF $\mathcal{F}$ is a mapping ρ defined from the set of variables of $\mathcal{F}$ into the set $\{True, False\}$. If $\rho[p]$ is the value for the positive literal p then $\rho[\neg p] = \neg\rho[p]$. The value of a clause $p_1 \vee p_2 \vee \ldots \vee p_n$ in ρ is *True*, if the value *True* is assigned to at least one of its literals in ρ, *False* otherwise. By convention, we define the value of the empty clause ($n = 0$) to be *False*. The value $\rho[\mathcal{F}]$ is *True* if the value of each clause of $\mathcal{F}$ is *True*, *False*, otherwise. We say that a CNF formula $\mathcal{F}$ is satisfiable if there exists some truth assignments ρ that assign the value *True* to $\mathcal{F}$, it is unsatisfiable otherwise. In the first case I is called a model of $\mathcal{F}$. Let us remark that a CNF formula which contains the empty clause is unsatisfiable.

It is well-known (Tseitin, 1968) that for every propositional formula $\mathcal{F}$ there exists a formula $\mathcal{F}'$ in conjunctive normal form (CNF) such that $\mathcal{F}'$ is satisfiable iff $\mathcal{F}$ is satisfiable. In the following we will assume that the formulas are given in a CNF.

2.2 Permutations

Let $\Omega = \{1, 2, \ldots, N\}$ for some integer N, where each integer might represent a propositional variable. A permutation of Ω is a bijective mapping σ from Ω to Ω that is usually represented as a product of cycles of permutations. We denote by $Perm(\Omega)$ the set of all permutations of Ω and $\circ$ the composition of the permutation of $Perm(\Omega)$. The pair $(Perm(\Omega), \circ)$ forms the permutation group of Ω. That is, $\circ$ is closed and associative. The inverse of a permutation is a permutation and the identity permutation is a neutral element. A pair $(T, \circ)$ forms a sub-group of $(S, \circ)$ iff T is a subset of S and forms a group under the operation $\circ$. The orbit $\omega^{Perm(\Omega)}$ of an element ω of Ω on which the group $Perm(\Omega)$ acts is $\omega^{Perm(\Omega)} = \{\omega^\sigma | \omega^\sigma = \sigma(\omega), \sigma \in Perm(\Omega)\}$. A generating set of the group $Perm(\Omega)$ is a subset Gen of $Perm(\Omega)$ such that each element of $Perm(\Omega)$ can be written as a composition of elements of Gen. We write $Perm(\Omega) = < Gen >$. An element of Gen is called a generator. The orbit of $\omega \in \Omega$ can be computed by using only the set of generators Gen.

2.3 Frequent, Closed and Maximal Itemset Mining Problems

Let $\mathcal{L} = \{0, \ldots, m-1\}$ be a set of m items and $\mathcal{T} = \{0, \ldots, n-1\}$ a set of n transactions (transaction identifiers). A subset $I \subseteq \mathcal{L}$ is called an itemset and a transaction $t \in \mathcal{T}$ over $\mathcal{L}$ is in fact, a pair (t_{id}, I) where t_{id} is the transaction identifier and I the corresponding itemset. In the *basket data* example, t_{id} represents the customer identification and I the set of items he put in his basket (he bought). Usually, when there is no confusing, a transaction is just expressed by its identifier. A transaction database $\mathcal{D}$ over $\mathcal{L}$ is a finite set of transactions such that no different transactions have the same identifier. Such a data set expresses in the *basket data* the different transactions made by customers. A transaction database can be seen as a binary matrix $n \times m$, where $n = \mid \mathcal{T} \mid$ and $m = \mid \mathcal{L} \mid$, with $\mathcal{D}_{t,i} \in \{0, 1\}$ forall $t \in \mathcal{T}$ and forall $i \in \mathcal{L}$. More precisely, a transaction database is expressed by the set $D = \{(t, I) \mid t \in \mathcal{T}, I \subseteq \mathcal{L}, \forall i \in I : \mathcal{D}_{t,i} = 1\}$. The *coverage* $C_\mathcal{D}(I)$ of an itemset I in a transaction database $\mathcal{D}$ is the set of all transactions in which I occurs. That is, $C_\mathcal{D}(I) = \{t \in \mathcal{T} \mid \forall i \in I, \mathcal{D}_{t,i} = 1\}$. The *support* $S_\mathcal{D}(I)$ of an itemset I in $\mathcal{D}$ is the number $|C_\mathcal{D}(I)|$ of transactions supporting I. It is just the cardinality of its coverage set. Moreover, the *frequency* $F_\mathcal{D}(I)$ of I in $\mathcal{D}$ is defined by $\frac{|C_\mathcal{D}(I)|}{|\mathcal{D}|}$.

Example 1. *Consider the transaction database $\mathcal{D}$ made over the set of drink items $\mathcal{L} = \{Beer, Wine, Whisky, Cognac, Vodka, Pastis, Ricard, Gin, Coke, Pepsi, Shweps, Juice, Water, Orangina\}$. For example, we can see in Table 1 that the itemset $I = \{Beer, Wine\}$ has $C_\mathcal{D}(I) = \{001, 002, 003, 004\}$, $S_\mathcal{D}(I) = \mid C_\mathcal{D}(I) \mid = 4$, and $F_\mathcal{D}(I) = 0,4$.*

Given a transaction database $\mathcal{D}$ over $\mathcal{L}$, and θ a minimal support threshold, an itemset I is said to be

Table 1: An instance of a transaction database.

t_{id}	*itemset*
001	Beer, Wine, Whisky, Vodka, Cognac, Water
002	Beer, Wine, Whisky, Vodka, Gin, Water
003	Beer, Wine, Whisky, Water
004	Beer, Wine, Vodka, Water
005	Ricard, Coke, Pepsi, Water
006	Pastis, Pepsi, Coke, Water
007	Shweps, Orangina, Pepsi
008	Shweps, Orangina, Coke
009	Juice, Orangina, Pepsi
010	Juice, Orangina, Coke

frequent if $S_{\mathcal{D}}(I) \geq \theta$. I is a closed frequent itemset if in addition to the frequency constraint it satisfies the following constraint: for all itemset J such that $I \subset J$, $S_{\mathcal{D}}(I) > S_{\mathcal{D}}(J)$. I is said to be a maximal frequent itemset if in addition to the frequency constraint it satisfies the following constraint: for all itemset J such that $I \subset J$, $S_{\mathcal{D}}(J) < \theta$. Both closed and maximal itemsets are two known condensed representation for frequent itemsets. The data mining tasks we are dealing with in this work are defined as follows:

Definition 1. *1. The frequent itemset mining task consists in computing the following set* $\mathcal{FIM}_{\mathcal{D}}(\theta) = \{I \subseteq \mathcal{L} | S_{\mathcal{D}}(I) \geq \theta\}$.

2. *The closed frequent itemset mining task consists in computing the following set* $\mathcal{CLO}_{\mathcal{D}}(\theta) = \{I \in \mathcal{FIM}_{\mathcal{D}}(\theta) | \forall J \subseteq \mathcal{L}, I \subset J, S_{\mathcal{D}}(I) > S_{\mathcal{D}}(J)\}$.
3. *The maximal frequent itemset mining task consists in computing the following set* $\mathcal{MAX}_{\mathcal{D}}(\theta) = \{I \in \mathcal{FIM}_{\mathcal{D}}(\theta) | \forall J \subseteq \mathcal{L}, I \subset J, S_{\mathcal{D}}(J) < \theta\}$.

The anti-monotonicity property in itemset mining expresses the fact that all the subsets of a frequent itemset are also frequent itemsets. More precisely:

Proposition 1. (Anti-monotonicity) *Let* θ *be a minimal support threshold, if the itemset* I *is such that* $S_{\mathcal{D}}(I) \geq \theta$, *then* $\forall J \subseteq I$, $S_{\mathcal{D}}(J) \geq \theta$.

3 SYMMETRY IN BOOLEAN SATISFIABILITY BASED ITEMSET MINING

Both constraint programming and Satisfiability are two known declarative programming frameworks where the user has just to specify the problem he want to solve rather than specifying how to solve it. The frequent itemset mining tasks and some of its variants (closed, maximal, etc) had been encoded for the first time in (Raedt et al., 2008; Guns et al., 2011a) as constraint programming tasks where a constraint solver could be used as a black box to solve them. Since that, other works ((Jabbour et al., 2013c), (Henriques et al., 2012), (Métivier et al., 2012), (Khiari et al., 2010), (Raedt et al., 2010), (Jabbour et al., 2013b)) expressed the data mining tasks as a satisfiability problem where the mining tasks are represented by propositional formulas that are translated into their conjunctive normal forms (CNF) which will be given as inputs to a SAT solver. In this work we use the encoding proposed in (Jabbour et al., 2013c) which we augment by the symmetry breaking predicates that are used to avoid enumerating the symmetrical models or the symmetrical no-goods of the resulting CNF encoding.

The general idea behind the CNF encoding of an itemset mining task defined on a transaction database $\mathcal{D}$ is to express each of its interpretations as a pair (I,T) where I represents an itemset and T its covering transaction subset in $\mathcal{D}$. To do that, a boolean variable I_i is associated with each item $i \in \mathcal{L}$ and a variable T_t is associated with each transaction $t \in \mathcal{T}$. The itemset I is then defined by all the variables I_i that are true. That is $I_i = 1$, if $i \in I$, and $I_i = 0$ if $i \notin I$. The set of transaction T covered by I is then defined by the set of variable T_t that are true. That is, $T_t = 1$ if $t \in C_{\mathcal{D}}(I)$ and $T_t = 0$ if $t \notin C_{\mathcal{D}}(I)$.

For instance, the $\mathcal{FIM}_{\mathcal{D}}(\theta)$ task can be seen as the search of the set of models $M = \{(I,T) \mid I \subseteq \mathcal{L}, T \subseteq \mathcal{T}, T = C_{\mathcal{D}}(I), |T| \geq \theta\}$. We have to encode both the covering constraint $T = C_{\mathcal{D}}(I)$ and the frequency constraint $|T| \geq \theta$. These constraints are expressed by the following boolean constraints:

$$\bigwedge_{t \in \mathcal{T}} (\neg T_t \leftarrow \bigvee_{i \in \mathcal{L}, \mathcal{D}_{t,i}=0} I_i)$$

$$\sum_{t \in \mathcal{T}} T_t \geq \theta$$

The frequent closed itemset task is specified by adding to the two previous constraints the following constraints:

$$\bigwedge_{t \in \mathcal{T}} (\neg T_t \rightarrow \bigvee_{i \in \mathcal{L}, \mathcal{D}_{t,i}=0} I_i)$$

$$\bigwedge_{i \in \mathcal{L}} ((\bigwedge_{t \in \mathcal{T}} T_t \rightarrow \mathcal{D}_{t,i} = 1) \rightarrow I_i)$$

The maximal frequent itemset mining is specified by adding the following constraint:

$$\bigwedge_{i \in \mathcal{L}} ((\sum_{t \in \mathcal{T}} T_t \times \mathcal{D}_{t,i} \geq \theta) \rightarrow I_i)$$

We denote by $CNF(k, \mathcal{D})$, the CNF formula encoding the data mining task k over the transaction database $\mathcal{D}$, where k refers to $\mathcal{FIM}_{\mathcal{D}}(\theta)$, $\mathcal{CLO}_{\mathcal{D}}(\theta)$

or $\mathcal{MAX}_{\mathcal{D}}(\theta)$. We also note $P^k_{\mathcal{D}}$ a predicate representing the task *k* in $\mathcal{D}$. Then an itemset $I \subseteq \mathcal{L}$ having $T \subseteq \mathcal{T}$ as a cover verifies $P^k_{\mathcal{D}}$ ($P^k_{\mathcal{D}}(I,T) = true$) if I is an itemset which is an answer to the data mining task *k* and T is its cover.

Remark 1. *We recall that a model J of $CNF(k,\mathcal{D})$ is a pair (I,T) where the part I expresses the itemset which is an answer to the considered task k and the part T encodes its cover. More precisely each literal I_i which is true in I represents the item i in the itemset I' which is an answer to the task k and each literal T_t which is true in T represents the transaction t in T' which is the corresponding cover of I'. In the sequel we denote by the pair (I',T') the itemset and its cover that are extracted from an interpretation $J = (I,T)$ of $CNF(k,\mathcal{D})$.*

Symmetry is well studied in constraint programming and propositional satisfiability. Since Krishnamurthy's (Krishnamurthy, 1985) symmetry definition and the one given in (Benhamou and Sais, 1992b; Benhamou and Sais, 1994b) in propositional logic, several other definitions are given by the CP community.

Symmetry has already been defined in itemset mining (Jabbour et al., 2012; Jabbour et al., 2013a). We give in the following a similar definition and show how to eliminate such symmetry by means of symmetry breaking predicates that we add to the Boolean encoding to solve efficiently some data mining tasks like frequent, closed or maximal itemset mining.

Definition 2. *Let $\mathcal{D}$ be a transaction database over a set of items $\mathcal{L}$. A symmetry of $\mathcal{D}$ is a permutation σ defined on $\mathcal{L}$ such that $\sigma(\mathcal{D}) = \mathcal{D}$*

Remark 2. *It is obvious to see that a permutation on the set of items $\mathcal{L}$, induces a permutation $\sigma_{\mathcal{T}}$ on the set of transactions $\mathcal{T}$ and a permutation $\sigma_{\mathcal{D}}$ on the data-set $\mathcal{D}$ itself. We denote such permutations only by σ when there is no confusion.*

A symmetry of $\mathcal{D}$ is an item permutation that leaves $\mathcal{D}$ invariant. If we denote by $Perm(\mathcal{L})$ the group of permutations of $\mathcal{L}$ and by $Sym(\mathcal{L}) \subset Perm(\mathcal{L})$ the subset of permutations of $\mathcal{L}$ that are the symmetries of $\mathcal{D}$, then $Sym(\mathcal{L})$ is trivially a sub-group of $Perm(\mathcal{L})$.

Theorem 1. *Let σ be a symmetry of a transaction database $\mathcal{D}$, $I \subseteq \mathcal{L}$ an itemset having a cover $T \subseteq \mathcal{T}$, and $P^k_{\mathcal{D}}$ the predicate expressing the data mining task k in $\mathcal{D}$, then $P^k_{\mathcal{D}}(I,T) = true$ iff $P^k_{\mathcal{D}}(\sigma(I),\sigma(T)) = true$.*

Proof. It is trivial to see that a symmetry of $\mathcal{D}$ verifies such property. Indeed, if σ is a symmetry of $\mathcal{D}$, then $\sigma(\mathcal{D}) = \mathcal{D}$, thus it results that $\mathcal{D}$ and $\sigma(\mathcal{D})$ have the same itemsets and covers satisfying the predicate $P^k_{\mathcal{D}}$. Thus σ must transform each itemset I with a cover T verifying the predicate $P^k_{\mathcal{D}}$ to an itemset $\sigma(I)$ with a cover $\sigma(T)$ verifying the predicate $P^k_{\mathcal{D}}$. □

In other words the symmetry σ of $\mathcal{D}$ transforms each itemset *I* having a cover *T* which is a solution to the data mining task *k* into a symmetrical itemset $\sigma(I)$ having a cover $\sigma(T)$ which is also a solution of the task *k*. It also transforms each itemset which is not a solution to the task *k* into a symmetrical itemset which will not be a solution to the task *k*. For instance if the task *k* concerns the frequent itemset mining problem, then by applying σ to a frequent itemset *I* we obtain a symmetrical frequent itemset $\sigma(I)$. If *I* is not frequent, then $\sigma(I)$ will not be frequent too.

Example 2. *Consider the transaction database defined in Table 1 of Example 1 and the permutation $\sigma = (Whisky, Vodka)(Cognac, Gin)(Ricard, Pastis)(Wine, Beer)(Shweps, Juice)(Pepsi, Coke)$ which is defined on the set of items $\mathcal{L}$ of $\mathcal{D}$. We can see that $\sigma(\mathcal{D})=\mathcal{D}$, then σ is a symmetry of $\mathcal{D}$.*

Now, we give an important property which establishes a relationship between the symmetries of a transaction database $\mathcal{D}$ and the Boolean encoding $CNF(k,\mathcal{D})$ of the data mining task *k* defined over $\mathcal{D}$.

Proposition 2. *Let $\mathcal{D}$ be a transaction database, $CNF(k,\mathcal{D})$ the Boolean encoding of the data mining task k, σ a symmetry of $\mathcal{D}$ and $J = (I,T)$ an interpretation of $CNF(k,\mathcal{D})$, then J is a model of $CNF(k,\mathcal{D})$ iff $\sigma(J)$ is a model of $CNF(k,\mathcal{D})$.*

Proof. Let σ be a symmetry of the transaction database $\mathcal{D}$ and $J = (I,T)$ a model of the Boolean encoding $CNF(k,\mathcal{D})$. It results that the corresponding pair itemset and cover (I',J') verify the predicate $P^k_{\mathcal{D}}$ of the data mining task *k*, that is $P^k_{\mathcal{D}}(I',J') = true$. We have to prove that $\sigma(J) = (\sigma(I),\sigma(T))$ is also a model of $CNF(k,\mathcal{D})$. The permutation σ is a symmetry of $\mathcal{D}$, thus by Theorem 1, it results that the pair $(\sigma(I'),\sigma(J'))$ verifies the predicate $P^k_{\mathcal{D}}$, that is $P^k_{\mathcal{D}}(\sigma(I'),\sigma(J')) = true$. Therefore $\sigma(J)$ is also a model of $CNF(k,\mathcal{D})$, since the pair $(\sigma(I'),\sigma(J'))$ verifying the predicate $P^k_{\mathcal{D}}$ is extracted from the model $\sigma(J) = (\sigma(I),\sigma(T))$ of $CNF(k,\mathcal{D})$. □

Remark 3. *The previous proposition allows us to use the symmetries of a transaction database $\mathcal{D}$ in its corresponding Boolean encoding $CNF(k,\mathcal{D})$ in order to detect symmetrical models and consider only one element in each symmetrical equivalent class. This gives an important alternative for symmetry exploitation in constraint-based data mining methods. Indeed, we*

can just compute the symmetries of $\mathcal{D}$ instead of computing those of its Boolean $CNF(k,\mathcal{D})$ which could be time consuming. This could accelerate the symmetry detection as the size of the transaction database $\mathcal{D}$ is generally substantially smaller than the size of its corresponding boolean encoding $CNF(k,\mathcal{D})$.

In Example 1, if we consider $\theta = 2$ and the symmetry σ of Example 2, then there will be symmetrical frequent itemsets in $\mathcal{D}$. For instance, both $I_1 = \{Beer, Wine, Wisky, Water\}$ and $I_2 = \{Shweps, Pepsi\}$ are frequent itemsets in $\mathcal{D}$. By the symmetry σ we can deduce that $\sigma(I_1) = \{Beer, Wine, Vodka, Water\}$ and $\sigma(I_2) = \{Juice, Coke\}$ are also frequent itemsets. These are what we call symmetrical frequent itemsets of $\mathcal{D}$ or symmetrical models [2] of $CNF(k,\mathcal{D})$. A symmetry σ transforms each frequent itemset (a model of the CNF encoding) into a frequent itemset and each non frequent itemset (a no-good of the CNF encoding) into a non frequent itemset. Symmetry elimination offers the advantage to enumerate only non-symmetrical patterns (like $I-1$ and I_2 here) which are considered as the most pertinent to the user for understanding the data.

4 SYMMETRY DETECTION

The most known technique to detect syntactic symmetries for CNF formulas in satisfiability is the one consisting in reducing the considered formula into a graph (Crawford et al., 1996; Aloul et al., 2002; Aloul et al., 2003b; Aloul et al., 2004) whose automorphism group is identical to the symmetry group of the original formula. We adapt the same approach here to detect the syntactic symmetries of a transaction database $\mathcal{D}$. As it is done in (Jabbour et al., 2012), we represent the database $\mathcal{D}$ by a graph $G_{\mathcal{D}}$ that we use to compute the symmetry group of $\mathcal{D}$ by means of its automorphism group. When this graph is built, we use a graph automorphism tool like Saucy (Aloul et al., 2002) to compute its automorphism group which gives the symmetry group of $\mathcal{D}$. We summarize bellow the construction of the graph which represent the transaction database $\mathcal{D}$. Given a transaction database $\mathcal{D}$, the associated colored graph $G_{\mathcal{D}}(V,E)$ is defined as follows:

- The set of colored vertices $V = \mathcal{L} \cup \mathcal{T}$ is build as follows:
  1. Each item $i \in \mathcal{L}$ is represented by a vertex $i \in V$ of the color 1 in $G_{\mathcal{D}}(V,E)$.
  2. Each item $t \in \mathcal{T}$ is represented by a vertex $t \in V$ of the color 2 in $G_{\mathcal{D}}(V,E)$.
- The set of edges E is defined by $E = \{(t,i) \mid \mathcal{D}_{t,i} = 1\}$. That is, an edge connects each transaction vertex $t \in \mathcal{T}$ to each vertex representing an item supported by t.

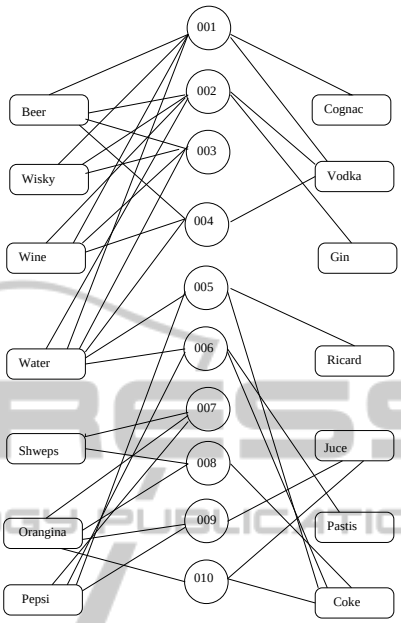

Figure 1: The graph of the transaction Database of Table 1.

Example 3. *Consider the transaction database $\mathcal{D}$ of Table 1 given in example 1. Its corresponding graph $G_{\mathcal{D}}(V,E)$ is shown in Figure 1. We can see for instance that the vertex permutation $\gamma = (Wisky, Vodka)(Cognac, Gin)(Ricard, Pastis)(Pepsi, Coke)(Shweps, Juice)(001,002)(003,004)(005,006)(007,010)(008,009)$ is one among the automorphisms of $G_{\mathcal{D}}(V,E)$. The restriction of the automorphism γ to $\mathcal{L}$ represents the symmetry $\sigma = (Wisky, Vodka)(Cognac, Gin)(Ricard, Pastis)(Pepsi, Coke)(Shweps, Juice)$ that we used in Example 2.*

An important property of the graph $G_{\mathcal{D}}(V,E)$ is that it preserves the group of symmetries of $\mathcal{D}$. That is, the symmetry group of $\mathcal{D}$ is identical to the automorphism group of its graph representation $G_{\mathcal{D}}(V,E)$, thus we could use a graph automorphism system like Saucy on $G_{\mathcal{D}}(V,E)$ to detect the symmetry group of $\mathcal{D}$. The graph automorphism system returns a set of generators *Gen* of the symmetry group from which we can deduce each symmetry of $\mathcal{D}$.

5 SYMMETRY ELIMINATION

Here we deal with the global symmetry which is present in the formulation of the given problem that

[2] Here, we omitted the part T of the model representing the cover of I.

is represented by the transaction database $\mathcal{D}$. Global symmetry can be eliminated in a static way in a preprocessing phase by just adding the symmetry breaking predicates to the Boolean encoding $CNF(k,\mathcal{D})$ and use a SAT solver as a black box on the resulting CNF formula.

We shall compute the Lex-Leader Symmetry Breaking Predicate (LL-SBP) induced by the automorphisms of $G_{\mathcal{D}}$. More precisely, the group of automorphisms $Aut(G_{\mathcal{D}})$ of the graph $G_{\mathcal{D}}$ (or the symmetry group $Sym(\mathcal{D})$ of $\mathcal{D}$) induces an equivalence relation on the set of interpretations of $CNF(k,\mathcal{D})$. That is, an interpretation I is equivalent to another interpretation J of $CNF(k,\mathcal{D})$ if there exists a symmetry σ of $\mathcal{D}$ such that $J = \sigma(I)$. The symmetry breaking predicates are chosen such that they are true for exactly one interpretation in each equivalent class (the least interpretation in the lex ordering). In general, we introduce an ordering on the the variables I_i corresponding to the items of $\mathcal{L}$ and use it to construct a lexicographical order on the set of interpretations.

The construction of the symmetry-breaking predicate is based on the lex-leader method introduced by Crawford et al (Crawford et al., 1996). Given a symmetry group $Sym(\mathcal{D}) = \{\sigma_1, \sigma_2, \ldots, \sigma_k\}$ of $\mathcal{D}$ and a total ordering $I_1 < I_2 < \cdots < I_n$ on the variables of $CNF(k,\mathcal{D})$ corresponding to the items of $\mathcal{L}$. The partial lex-leader symmetry-breaking predicate (PLL-SBP) (Aloul et al., 2003a) that we have to add to $CNF(k,\mathcal{D})$ is expressed as follows:

$$PP(\sigma_l) = \bigwedge_{1\leq i\leq n} [\bigwedge_{1\leq j\leq i-1} (I_j = I_j^{\sigma_l}) \rightarrow (I_i \leq I_i^{\sigma_l})]$$

$$PLL-SBP(Sym(\mathcal{D})) = \bigwedge_{\sigma_l \in GEN(Sym(\mathcal{D}))} PP(\sigma_l)$$

$PP(\sigma_l)$ is the permutation predicate corresponding to the symmetry generator σ_l and the expression $(I_i \leq I_i^{\sigma_l})$ denotes the clause $(I_i \rightarrow I_i^{\sigma_l})$.

The $LL-SBP$ is translated to a linear size CNF formula by introducing auxiliary variables e_j to represent the expressions $(I_j = I_j^{\sigma_l})$. For example, $e_j \leftrightarrow (I_j = I_j^{\sigma_l})$ gives rise to the following implications:

$$(\neg I_j \vee \neg I_j^{\sigma_l} \vee e_i), (I_j \vee I_j^{\sigma_l} \vee e_i)$$
$$(\neg I_j \vee \neg e_i \vee I_j^{\sigma_l}), (I_j \vee \neg e_i \vee \neg I_j^{\sigma_l})$$

Some optimizations such that ones studied in Aloul (Aloul et al., 2003a) could be done to get a more compact CNF $PLL-SBP$.

6 EXPERIMENTS

In this section, we present an experimental analysis of our symmetry breaking approach for SAT based itemset mining.

6.1 Input Data-sets

We choose for our experiments two classes of data-sets:

- **Simulated data-sets:** : In this class, we use the simulated data-sets, generated specifically to involve interesting symmetries. The data is available at http://www.cril.fr/decMining.
- **Public datasets:** The datasets used in this class are well known in the data mining community and are available at https://dtai.cs.kuleuven.be/CP4IM/datasets/

6.2 The Experimented Methods

As we aim to enumerate all the frequent/closed itemsets on the SAT based encoding, our experiments are conducted using MiniSAT-Enum dedicated to the enumeration of all models of a given CNF formula. MiniSAT-Enum is obtained from MiniSAT 2.2 [3] as follows: each time a model is found a no-good (clause) is generated and added to the formula in order to avoid enumerating the same models. MiniSAT-Enum takes as input a CNF formula and a set of items variables and returns the set of frequent/closed itemsets.

The methods that we experimented and compared are the following:

1. **MiniSAT-Enum:** search without symmetry breaking on the CNF encoding of the data mining task $CNF(k,\mathcal{D})$
2. **MiniSAT-Enum-SBP:** search with symmetry breaking. This method generates in a preprocessing phase the symmetry-breaking predicates, then apply MiniSAT-Enum to the resulting CNF instance $CNF(k,\mathcal{D})$+ $PLL-SBP$. The CPU time of *MiniSAT-Enum-Sym* includes the time spent to generate the $PLL-SBP$.
3. **MiniSAT-Enum-ISB:** this method (Jabbour et al., 2012), called ItemPair symmetry breaking (ISB), eliminates symmetries in a preprocessing step, by rewriting the transaction database $\mathcal{D}$ as a $\mathcal{D}'$ by eliminating symmetric items. MiniSAT-Enum is then applied on the CNF formula $CNF(k,\mathcal{D}')$ encoding the new transaction database.

[3] MiniSAT: http://minisat.se/

In our experiments, we exploit Saucy[4], a new implementation of the Nauty system. It is originally proposed in (Aloul et al., 2002) and significantly improved in (Darga et al., 2008). The latest version of Saucy outperforms all the existing tools by many orders of magnitude, in some cases improving runtime from several days to a fraction of a second.

We are interested on the CPU time and on the number of models or closed/frequent itemsets found with and without symmetry breaking. All the experimental results presented in this section have been obtained with a Quad-core Intel Xeon X5550 (2.66GHz, 32 GB RAM) cluster.

6.3 The Obtained Results

In Figure 2 and 3, we present the results obtained on a simulated data *dataset-gen-jss-5*. The experiment show the comparison of MiniSAT-Enum (CFIM), MiniSAT-Enum-SBP (CFIM-SBP) and MiniSAT-Enum-ISB (CFIM-ISB) w.r.t. CPU time in seconds (Figure 2) and number of patterns (Figure 3). As we can see, by breaking symmetries, we significantly reduce both the number of closed frequent itemsets (output) and CPU-time. Such reduction of the size of the output induces a significant reduction of the search time. Interestingly, breaking symmetries using by adding SBP on the CNF encoding of the itemset mining task (CFIM-SBP) is clearly better than eliminating symmetric items on the original transaction database (CFIM-ISB). This experiment show that our approach break more symmetries than the one proposed in (Jabbour et al., 2012).

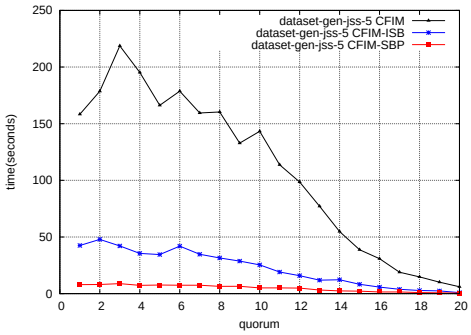

Figure 2: Results on simulated data (Closed frequent itemsets): CPU time.

The second experiment is conducted on well-know academic datasets. In this experiment, we are interested on the frequent itemsets mining problem. In Figure 4 and 5, we present the comparative results

[4]Saucy2: Fast symmetry discovery - http://vlsicad.eecs.umich.edu/BK/SAUCY/

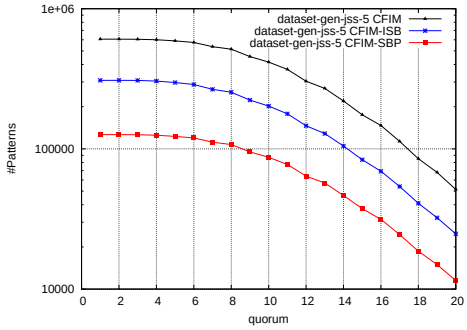

Figure 3: Results on simulated data (Closed frequent itemsets): number of patterns.

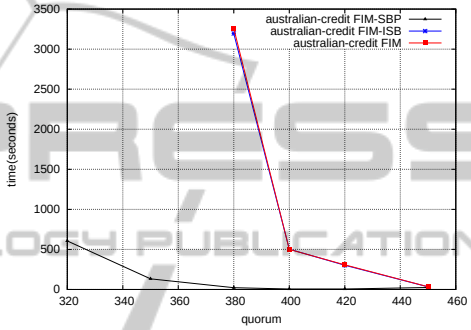

Figure 4: Results on public data - *Australian* - (frequent itemsets): CPU time.

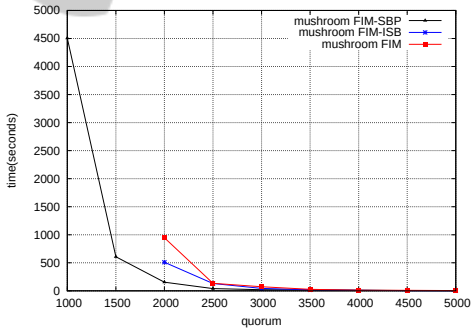

Figure 5: Results on public data - *Mushroom* - (frequent itemsets): CPU time.

w.r.t. the computation time. No reduction is observed on the number of frequent itemsets. On these datasets, most of found symmetries involves items in the same transactions. This explains why these particular symmetries does not reduce the number of closed/frequent itemsets. However, even when the size of the output is not reduced, breaking symmetries using our approach significantly reduce the search space. In general symmetry breaking reduces dramatically the search space and the corresponding CPU time for this declarative approach, but did not reach the performances of optimized dedicated algorithms like FPgrowth for example.

7 RELATED WORKS

The purpose of eliminating symmetry in data mining tasks is in general either to obtain a more compact output or to decrease the necessary CPU time for its generation or to handle new mining properties to find interesting frequent patterns. Some symmetry works are introduced in the field of Data mining following this direction.

Symmetries in graph mining are studied in Desrosiers et al. (Desrosiers et al., 2007), and in Vanetik (Vanetik, 2010). The area of graph mining has a great importance in many applications. In Desrosiers et al. (Desrosiers et al., 2007) symmetry is exploited to prune the search space of sub-graph mining algorithms. However, in Vanetik (Vanetik, 2010), symmetry is used to find interesting frequent sub-graphs (those having limited diameter and high symmetry). Such graphs represent the more structurally important patterns in all of the chemical, text and genetic data-sets. Their technique allows also to reduce the necessary CPU to find such graphs.

Murtagh et al in (Murtagh and Contreras, 2010) used symmetry to get a powerful means of structuring and analyzing massive, high dimensional data stores. They illustrate the power of hierarchical clustering in case studies in chemistry and finance.

Symmetry is also studied in transaction database using Zero-BDDs (Minato, 2006). These symmetries looks very particular, since they are just transpositions of two items and still identity for the remain items. They used such symmetry to study the properties of symmetrical patterns. Such symmetries are used in (Gly et al., 2005) to explain in some cases why the number of rules of a minimal cover of a relation is exponential in number of items.

Two symmetry elimination approaches for frequent itemset mining are introduced in (Jabbour et al., 2012). They consist in rewriting the transaction database in pre-processing phase by eliminating the symmetrical of some items. These approaches are specific to the data mining task considered. They could be combined with our method for the itemset mining task. Another approach integrate dynamic symmetry elimination in the Apriori-like algorithm (Jabbour et al., 2013a) in order to prune the search space of enumerating all the frequent item sets of a transaction database.

All of these methods are specific to the data mining task considered and the target method used to solve. They are different from the approach which develop here, since our approach is generic and declarative. It will work with all data mining task that is expressed in a constraint language.

8 CONCLUSION

We studied in this work the notion of symmetry for data mining tasks expressed as declarative constraints. We showed how the symmetries of the given transaction database can be detected and eliminated by adding symmetry-breaking predicate to the constraint encoding of the considered data mining task. We showed that even though such symmetries could not be syntactically the symmetries of the CNF encoding of the data mining problem, they conserve the set of its models (the set of interesting patterns). Detecting symmetry on the given transaction database rather than the CNF encoding of the considered data mining task could result in a great save of efforts in the symmetry detection. Indeed, the size of the transaction database is in general smaller then its corresponding CNF encoding. The transaction database is represented by a colored graph that is used to compute its symmetries. The symmetry group of the transaction database is identical to the automorphism group of the corresponding graph. The graph automorphism tools SAUCY is naturally used on the obtained graph to detect the group of symmetries of the transaction database. This symmetry is eliminated statically by adding in a pre-processing phase the well known lex order symmetry breaking predicates to the CNF encoding of the considered data mining task. We then applied as a black box a SAT model enumeration algorithm on this resulting encoding to solve data mining problem.

The proposed symmetry breaking method is implemented and experimented on a variety of transaction data-sets. The first experimental results confirmed that eliminating symmetry is profitable for the considered data mining tasks.

As a future work, we are looking to eliminate symmetry in other data mining problems and try to extend symmetry exploitation to the local symmetry that could exists at some nodes of the search tree. Both kind of exploitation could be complementary, then one can naturally think on the advantage of combining them.

REFERENCES

Agrawal, R., Imieliński, T., and Swami, A. (1993). 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. ACM.

Agrawal, R. and Srikant, R. (1994). Fast algorithms for mining association rules in large databases. In *Pro-*

ceedings of the 20th International Conference on Very Large Data Bases, VLDB '94, pages 487–499, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Aloul, F. A., Markov, I. L., and Sakallah, K. A. (2003a). Shatter: efficient symmetry-breaking for boolean satisfiability. In *DAC*, pages 836–839. ACM.

Aloul, F. A., Ramani, A., Markov, I. L., and Sakallah, K. A. (2002). Solving difficult SAT instances in the presence of symmetry. In *Proceedings of the 39th Design Automation Conference (DAC 2002)*, pages 731–736. ACM Press.

Aloul, F. A., Ramani, A., Markov, I. L., and Sakallah, K. A. (2003b). Solving difficult instances of boolean satisfiability in the presence of symmetry. *IEEE Trans. on CAD of Integrated Circuits and Systems*, 22(9):1117–1137.

Aloul, F. A., Ramani, A., Markov, I. L., and Sakallak, K. A. (2004). Symmetry breaking for pseudo-boolean satisfiabilty. *In ASPDAC'04*, pages 884–887.

Benhamou, B. (1994). Study of symmetry in constraint satisfaction problems. *PPCP'94*, pages 246–254.

Benhamou, B. and Sais, L. (1992a). Theoretical study of symmetries in propositional calculus and application. *In CADE'11*, pages 281–294.

Benhamou, B. and Sais, L. (1992b). Theoretical study of symmetries in propositional calculus and applications. In *CADE*, pages 281–294.

Benhamou, B. and Sais, L. (1994a). Tractability through symmetries in propositional calculus. *In JAR*, 12:89–102.

Benhamou, B. and Sais, L. (1994b). Tractability through symmetries in propositional calculus. *J. Autom. Reasoning*, 12(1):89–102.

Besson, J., Boulicaut, J.-F., Guns, T., and Nijssen, S. (2010). Generalizing itemset mining in a constraint programming setting. In *Inductive Databases and Constraint-Based Data Mining*, pages 107–126. Springer.

Bonchi, F. and Lucchese, C. (2007). Extending the state-of-the-art of constraint-based pattern discovery. *Data Knowl. Eng.*, 60(2):377–399.

Bucilă, C., Gehrke, J., Kifer, D., and White, W. (2003). Dualminer: A dual-pruning algorithm for itemsets with constraints. *Data Mining and Knowledge Discovery*, 7(3):241–272.

Burdick, D., Calimlim, M., and Gehrke, J. (2001). Mafia: A maximal frequent itemset algorithm for transactional databases. In *In ICDE*, pages 443–452.

Crawford, J., Ginsberg, M., Luks, E., and Roy, A. (1996). Symmetry-breaking predicates for search problems. In *Knowledge Representation (KR)*, pages 148–159. Morgan Kaufmann.

Darga, P. T., Sakallah, K. A., and Markov, I. L. (2008). Faster symmetry discovery using sparsity of symmetries. In *Proceedings of the 45th Annual Design Automation Conference*, DAC '08, pages 149–154, New York, NY, USA. ACM.

Desrosiers, C., Galinier, P., Hansen, P., and Hertz, A. (2007). Improving frequent subgraph mining in the presence of symmetry. In *MLG*.

Freuder, E. (1991). Eliminating interchangeable values in constraints satisfaction problems. *AAAI-91*, pages 227–233.

Grahne, G. and Zhu, J. (2005). Fast algorithms for frequent itemset mining using fp-trees. *IEEE Trans. on Knowl. and Data Eng.*, 17(10):1347–1362.

Guns, T., Dries, A., Tack, G., Nijssen, S., and Raedt, L. D. (2013). Miningzinc: A modeling language for constraint-based mining. In *International Joint Conference on Artificial Intelligence*, pages –, Beijing, China.

Guns, T., Nijssen, S., and De Raedt, L. (2011a). Itemset mining: A constraint programming perspective. *Artif. Intell.*, 175(12-13):1951–1983.

Guns, T., Nijssen, S., and de Raedt, L. (2011b). k-pattern set mining under constraints. *IEEE TKDE*, 99(PrePrints).

Gly, A., Medina, R., Nourine, L., and Renaud, Y. (2005). Uncovering and reducing hidden combinatorics in guigues-duquenne bases. In Ganter, B. and Godin, R., editors, *ICFCA*, Lecture Notes in Computer Science, pages 235–248. Springer.

Han, J., Pei, J., and Yin, Y. (2000). Mining frequent patterns without candidate generation. In *Proceedings of the 2000 ACM SIGMOD International Conference on Management of Data*, SIGMOD '00, pages 1–12, New York, NY, USA. ACM.

Henriques, R., Lynce, I., and Manquinho, V. M. (2012). On when and how to use sat to mine frequent itemsets. *CoRR*, abs/1207.6253.

Jabbour, S., Khiari, M., Sais, L., Salhi, Y., and Tabia, K. (2013a). Symmetry-based pruning in itemset mining. In *25th International Conference on Tools with Artificial Intelligence(ICTAI'13)*, Washington DC, USA. IEEE Computer Society.

Jabbour, S., Sais, L., and Salhi, Y. (2013b). Boolean satisfiability for sequence mining. In *CIKM*, pages 649–658.

Jabbour, S., Sais, L., and Salhi, Y. (2013c). Top-k frequent closed itemset mining using top-k sat problem. In *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML/PKDD'13)*, volume 146, pages 131–140. Springer.

Jabbour, S., Sais, L., Salhi, Y., and Tabia, K. (2012). Symmetries in itemset mining. In *20th European Conference on Artificial Intelligence (ECAI '12)*, pages 432–437. IOS Press.

Khiari, M., Boizumault, P., and Crmilleux, B. (2010). Constraint programming for mining n-ary patterns. In Cohen, D., editor, *CP*, volume 6308 of *Lecture Notes in Computer Science*, pages 552–567. Springer.

Krishnamurthy, B. (1985). Short proofs for tricky formulas. *Acta Inf.*, 22(3):253–275.

Krishnamurty, B. (1985). Short proofs for tricky formulas. *Acta Inf.*, (22):253–275.

Métivier, J.-P., Boizumault, P., Crémilleux, B., Khiari, M., and Loudni, S. (2012). A constraint language for declarative pattern discovery. In *Proceedings of the 27th Annual ACM Symposium on Applied Computing*, SAC '12, pages 119–125, New York, NY, USA. ACM.

Minato, S. I. (2006). Symmetric item set mining based on zero-suppressed bdds. In Todorovski, L., Lavrac, N.,

and Jantke, K. P., editors, *Discovery Science*, volume 4265 of *Lecture Notes in Computer Science*, pages 321–326. Springer.

Minato, S. I., Uno, T., and Arimura, H. (2007). Fast generation of very large-scale frequent itemsets using a compact graph-based representation.

Murtagh, F. and Contreras, P. (2010). Hierarchical clustering for finding symmetries and other patterns in massive, high dimensional datasets. *CoRR*, abs/1005.2638.

Pei, J., Han, J., and Lakshmanan, L. V. S. (2004). Pushing convertible constraints in frequent itemset mining. *Data Min. Knowl. Discov.*, 8(3):227–252.

Puget, J. F. (1993). On the satisfiability of symmetrical constrained satisfaction problems. In *In J. Kamorowski and Z. W. Ras,editors, Proceedings of ISMIS'93, LNAI 689*, pages 350–361.

Raedt, L. D., Guns, T., and Nijssen, S. (2008). Constraint programming for itemset mining. In *KDD*, pages 204–212.

Raedt, L. D., Guns, T., and Nijssen, S. (2010). Constraint programming for data mining and machine learning. In *AAAI*.

Tiwari, A., Gupta, R., and Agrawal, D. (2010). A survey on frequent pattern mining: Current status and challenging issues. *Inform. Technol. J*, 9:1278–1293.

Tseitin, G. S. (1968). On the complexity of derivation in propositional calculus. In *Structures in the constructive Mathematics and Mathematical logic*, pages 115–125. H.A.O Shsenko.

Uno, T., Asai, T., Uchida, Y., and Arimura, H. (2003). Lcm: An efficient algorithm for enumerating frequent closed item sets. In *In Proceedings of Workshop on Frequent itemset Mining Implementations (FIMI03*.

Uno, T., Kiyomi, M., and Arimura, H. (2004). Lcm ver. 2: Efficient mining algorithms for frequent/closed/maximal itemsets. In *FIMI*.

Vanetik, N. (2010). Mining graphs with constraints on symmetry and diameter. In Shen, H. T., Pei, J., zsu, M. T., Zou, L., Lu, J., Ling, T.-W., Yu, G., Zhuang, Y., and Shao, J., editors, *WAIM Workshops*, volume 6185 of *Lecture Notes in Computer Science*, pages 1–12. Springer.

Zaki, M. J. and Hsiao, C.-J. (2005). Efficient algorithms for mining closed itemsets and their lattice structure. *IEEE Trans. on Knowl. and Data Eng.*, 17(4):462–478.