2013 IEEE 25th International Conference on Tools with Artificial Intelligence

Symmetry-Based Pruning in Itemset Mining

Said Jabbour, Mehdi Khiari, Lakhdar Sais, Yakoub Salhi, Karim Tabia
Univ Lille Nord de France, F-59000 Lille, France.
UArtois, CRIL UMR CNRS 8188, F-62300 Lens, France.
{*jabbour, khiari, sais, salhi, tabia*}*@cril.univ-artois.fr*

***Abstract*—In this paper, we show how symmetries, a fundamental structural property, can be used to prune the search space in itemset mining problems. Our approach is based on a dynamic integration of symmetries in APRIORI-like algorithms to prune the set of possible candidate patterns. More precisely, for a given itemset, symmetry can be applied to deduce other itemsets while preserving their properties. We also show that our symmetry-based pruning approach can be extended to the general Mannila and Toivonen pattern mining framework. Experimental results highlight the usefulness and the efficiency of our symmetry-based pruning approach.**

I. INTRODUCTION

Frequent Itemset Mining is one of the most fundamental problems in data mining. Since its introduction by Agrawal in [24], the field of data mining has flourished into several research areas with fundamental applications ranging from security to bioinformatics. This basic data mining task is now a building block of various other problems, such as the ones of association rule mining, frequent pattern mining in sequence data, data clustering, episode mining, etc. As pointed out in [25], a lot of works focussed on the design of highly scalable itemset mining algorithms for large scale datasets.

The main objective of this paper is to introduce a new symmetry-based pruning framework in data mining. As in other domains such as constraint programming and Boolean satisfiability where symmetries are extensively studied, our goal is to show that symmetry-based pruning is also an interesting issue in data mining and more generally for solving enumeration problems. To this end, we consider one of the fundamental data mining tasks, the itemset mining problem and its associated popular APRIORI algorithm.

Symmetry is a fundamental concept in computer science, mathematics, physics and many other domains. Many human artifacts from classroom in a university to machines in a company exhibit symmetries. Such symmetries allow us to reason and to understand more complex entities and systems. For example, in a mathematical proof, we sometimes state that a certain assumption can be made without loss of generality. In combinatorial problem solving, symmetry has been extensively studied. For instance, in scheduling problems, as certain machines might be interchangeable, from a valid schedule, one can permute these machines and still obtain a valid schedule. Exploiting symmetries allows us to reduce the search efforts, by avoiding the exploration of symmetrical parts of the search space. This is why this kind of structures and reasoning has received a lot of attention in constraint satisfaction problems (CSP) (e.g. [9]), propositional satisfiability (SAT) (e.g. [4]) and operation research (OR) (e.g. [15]). As far as we know, symmetry reasoning has been first introduced by Krishnamurthy [14] to shorten resolution proofs in propositional logic. He shows that a resolution proof system augmented with the symmetry rule is more powerful than resolution.

In data mining, symmetries are studied mainly in frequent graph mining (e.g. [11], [6], [26]). Recently, we proposed a new framework for breaking symmetries in itemset mining problems [12]. In [12], we consider symmetries as permutations between items that leave invariant the transaction database. First, we addressed symmetry detection in transaction databases, and we have shown how such discovered symmetries can be eliminated in a preprocessing step by rewriting the transaction database i.e. eliminating some items from the original transaction database. This approach is completely different from the symmetry breaking framework widely used in propositional satisfiability and constraint satisfaction problems. Indeed, in CSP and SAT, symmetries are usually eliminated by adding to the constraints network, symmetry breaking predicates (SBP) i.e. additional constraints.

In [12], we mentioned that symmetries can also be integrated in APRIORI-like algorithms to dynamically prune the search space. In this paper, we propose to study this issue in-depth. Our main motivation comes from the fact, that symmetry-based pruning can be more easily generalized to deal with other data mining problems, while breaking symmetries in a preprocessing step [12] highly depends on the input database and on the patterns to be mined. The second problem behind symmetry breaking approach comes from the fact that symmetries between items involved in the same transaction can not be eliminated, while they can be handled by the symmetry-based pruning approach presented in this paper.

We choose APRIORI algorithm, the most popular itemset mining algorithm, as an example to show the feasibility of our symmetry-based pruning approach. Experimental results

1082-3409/13 $31.00 © 2013 IEEE
DOI 10.1109/ICTAI.2013.78

tid	itemset
001	A, B, E, F
002	A, B, C, D
003	C, D, E, F
004	A, C
005	A, E
006	C, E
007	B, D
008	B, F
009	D, F

Table I
AN EXAMPLE OF A TRANSACTION DATABASE $\mathcal{D}$

show the relevance and the efficiency of our symmetry-based pruning approach (APRIORI algorithm with symmetry). To the best of our knowledge, it is the first operative approach taking benefit from symmetries of the datasets to prune the search space in itemset mining tasks.

The rest of this paper is organized as follows. After some preliminaries on itemset mining and symmetries (Section II), we introduce our symmetry-based pruning approach in Section III. In Section V, we describe the generalization of our symmetry pruning approach to the general Mannila and Toivonen pattern mining framework [17]. Section VI, gives an overview of related works on symmetries in data mining. Our experimental evaluation is given in Section IV before concluding with several paths for future research.

II. TECHNICAL BACKGROUND AND PRELIMINARY DEFINITIONS

A. Itemset Mining problem

Let $\mathcal{I}$ be a set of *items* and $\mathcal{T}$ a set of names, called *transaction identifiers*. A set $I \subseteq \mathcal{I}$ is called an *itemset*. A *transaction* is a couple (tid, I) where tid is the transaction identifier ($tid \in \mathcal{T}$) and I is an itemset ($I \in \mathcal{I}$). A *transaction database* $\mathcal{D}$ is a finite set of transactions over $\mathcal{I}$ such that each transaction has a unique identifier. We note $\mathcal{T}_{id}(\mathcal{D}) = \{tid|\ (tid, I) \in \mathcal{D}\}$ the set of transaction identifiers associated to $\mathcal{D}$. We use $\mathcal{I}_{items}(O)$ to denote the set of all the items appearing in the syntactic object O (e.g. a transaction database, itemset, etc). We say that a transaction (tid, I) *supports* an itemset J if $J \subseteq I$.

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

Example 1: Let us consider the transaction database $\mathcal{D}$ (cf. Table I) over the set of items $\mathcal{I} = \{A, B, C, D, E, F\}$. For instance, we have $\mathcal{S}upp(\{B, D\}, \mathcal{D})$= $|\{002, 007\}|$=2 and $\mathcal{F}req(\{B, D\}, \mathcal{D})$=$\frac{2}{9}$.

In the sequel, we consider $\mathcal{D}$ as a transaction database over $\mathcal{I}$ and λ a minimal support threshold with $0 \leq \lambda \leq |\mathcal{D}|$.

Proposition 1 (Anti-Monotonicity): Let I_1 and I_2 be two itemsets such that $I_1 \subseteq I_2$. If $\mathcal{S}upp(I_2, \mathcal{D}) \geq \lambda$ then $\mathcal{S}upp(I_1, \mathcal{D}) \geq \lambda$.

Definition 1 (Frequent Itemset Mining Problem): The frequent itemset mining problem consists in computing the following set:

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

B. Symmetry in Frequent Itemset Mining

Intuitively, a symmetry of a transaction database $\mathcal{D}$ is a permutation of items that leaves $\mathcal{D}$ invariant. Such permutation over the set of items induces a permutation over the transactions. As the transaction database remains unchanged, the properties of an itemset are preserved under the application of a symmetry. The set of symmetries forms a symmetry group, where the permutation that maps each item with itself is a neutral element. In the sequel, we use notations from computational group theory.

Let us now formally define symmetry in the frequent itemset mining problem.

Definition 2 (Permutation): A permutation p over a finite set $\mathcal{S}$ is a bijective mapping from $\mathcal{S}$ to $\mathcal{S}$.
Each permutation p can be represented by a set of cycles $c_1 \dots c_n$ where each cycle $c_i = (a_1, \dots, a_k)$ is a list of elements of $\mathcal{S}$ such that $p(a_j) = a_{j+1}$ for $j = 1, \dots, k-1$, and $p(a_k) = a_1$. In the sequel, for clarity reasons, we omit cycles of the form (a, a) in the description of permutations and symmetries.

Let $\mathcal{D}$ be a transaction database. We denote by $\mathcal{P}(\mathcal{I}_{items}(\mathcal{D}))$ and $\mathcal{P}(\mathcal{T}_{id}(\mathcal{D}))$ the sets of all the permutations over respectively $\mathcal{I}_{items}(\mathcal{D})$ and $\mathcal{T}_{id}(\mathcal{D})$. Moreover, we denote by $\mathcal{P}(\mathcal{T}_{id}(\mathcal{D}), \mathcal{I}_{items}(\mathcal{D}))$ the following set of permutations $\{\sigma \circ f \mid \sigma \in \mathcal{I}_{items}(\mathcal{D})$ *and* $f \in \mathcal{T}_{id}(\mathcal{D})\}$. It is easy to see that $(\mathcal{P}(\mathcal{T}_{id}(\mathcal{D}), \mathcal{I}_{items}(\mathcal{D})), \circ)$ is a group where the identity permutation is a neutral element. A structure $(S', \circ)$ is a sub-group of the group $(\mathcal{P}(\mathcal{T}_{id}(\mathcal{D}), \mathcal{I}_{items}(\mathcal{D})), \circ)$ if and only if $(S', \circ)$ is a group and $S' \subseteq \mathcal{P}(\mathcal{T}_{id}(\mathcal{D}), \mathcal{I}_{items}(\mathcal{D}))$.

Let $\mathcal{D}$ be a transaction database. A permutation $\sigma \circ f$ is extended to a transaction database $\mathcal{D}$ as follows: $\sigma \circ f(\mathcal{D}) = \{(f(tid), \sigma(I))|(tid, I) \in \mathcal{D}\}$ where $\sigma(I) = \{\sigma(a)|a \in I\}$.

Definition 3 (Symmetry): A symmetry of $\mathcal{D}$ is a permutation $\sigma \circ f$ in $\mathcal{P}(\mathcal{T}_{id}(\mathcal{D}), \mathcal{I}_{items}(\mathcal{D}))$ such that $\sigma \in \mathcal{P}(\mathcal{I}_{items}(\mathcal{D}))$, $f \in \mathcal{P}(\mathcal{T}_{id}(\mathcal{D}))$ and $\sigma \circ f(\mathcal{D}) = \mathcal{D}$.

Example 2: Let us consider again the transaction database $\mathcal{D}$ given in Table 1. Let $\sigma =$
$(A, C)(B, D)$ a permutation over $\mathcal{I}_{items}(\mathcal{D})$ and $f = (001, 003)(005, 006)(008, 009)$ a permutation over $\mathcal{T}_{id}(\mathcal{D})$. The permutation $\sigma \circ f$ is a symmetry of $\mathcal{D}$.
Let $\mathcal{S}(\mathcal{D})$ be the set of all the symmetries of $\mathcal{D}$. It is worth noting that $(\mathcal{S}(\mathcal{D}), \circ)$ is a sub-group of $(\mathcal{P}(\mathcal{T}_{id}(\mathcal{D}), \mathcal{I}_{items}(\mathcal{D})), \circ)$.

Obviously, as a symmetry leaves the transaction database invariant, all the properties of an itemset such as frequency, closeness, maximality are preserved. The following proposition, shows that symmetry preserves the frequency property.

Proposition 2: Let $\mathcal{D}$ be a transaction database, $\sigma \circ f$ a symmetry of $\mathcal{D}$, λ a minimal support threshold and I an itemset. $I \in \mathcal{FIM}(\mathcal{D}, \lambda)$ iff $\sigma(I) \in \mathcal{FIM}(\mathcal{D}, \lambda)$.

Proof: Since $\sigma \circ f$ is a symmetry, it is easy to see that the support of I is equal to the one of $\sigma(I)$. Therefore, $I \in \mathcal{FIM}(f(\mathcal{D}), \lambda)$ if and only if $\sigma(I) \in \mathcal{FIM}(\mathcal{D}, \lambda)$. ■

C. Symmetry Detection in Transaction Databases

One of the most popular means for discovering symmetries can be accomplished by encoding the problem into a graph and seek for graph automorphism i.e. permutations of its vertices that maps edges to edges. Graph automorphism is a particular case of graph isomorphism which is believed to be easier than NP-Complete and for which no polynomial algorithm is known. The good news is that graph isomorphism is rarely difficult in practice as demonstrated by the efficient available tools. Most of the symmetry detection tools mainly focus on the automorphism of a colored graph. The colors on vertices are used to constrain the mapping to vertices with the same color i.e. two vertices with different colors can not be mapped each others. The first practical implementation for graph automorphism, called `nauty`, is due to Brendan McKay [18]. In order to tackle larger graphs, different improvements in terms of data structures and sophisticated group-theoretic pruning heuristics have been proposed yielding to powerful symmetry detection systems [2], [13].

In order to search for symmetries in a transaction database $\mathcal{D}$, we only need to find how to encode $\mathcal{D}$ as a colored undirected graph $\mathcal{G}$ such that symmetries on $\mathcal{D}$ correspond to automorphisms of $\mathcal{G}$.

Definition 4: A *colored undirected graph* is a triplet $\mathcal{G} = (V, E, \eta)$ with vertex set V and edge set $E \in V \times V$ and η is a function from V to N that associates a positive integer (a color) to each vertex.

Definition 5: A permutation α of V is an automorphism of $\mathcal{G}$ iff $\sigma(\mathcal{G}) = \mathcal{G}$ or equivalently $\sigma(E) = E$.

Definition 6: An automorphism α of $\mathcal{G}$ respects a partition π of V if for each $v \in V$, v and $\sigma(v)$ belong to the same cell (or element) of π. The set of all automorphisms of $\mathcal{G}$ with respect to a partition π is called the automorphism group of $\mathcal{G}$ under π and is denoted $\mathcal{A}(\mathcal{G})_\pi$.

We now show how a transaction database can be encoded as a colored undirected graph.

Definition 7 (From $\mathcal{D}$ to $\mathcal{G}$): We define the colored undirected graph $\mathcal{G}$ associated to $\mathcal{D}$ as $\mathcal{G}(\mathcal{D}) = (V, E, \eta)$ where $V = \mathcal{I} \cup \mathcal{T}_{id}(\mathcal{D})$, $E = \{(tid, i) | \exists (tid, I) \in \mathcal{D}, i \in I\}$ and $\forall v \in V$,

$$\eta(v) = \begin{cases} 0 & \text{for } v \in \mathcal{I}_{items}(\mathcal{D}) \\ 1 & \text{for } v \in \mathcal{T}_{id}(\mathcal{D}) \end{cases}$$

Let us note that $\pi = \{\{v | v \in V, \eta(v) = 0\}, \{v | v \in V, \eta(v) = 1\}\}$ is the initial partition of V. Several refinements have been proposed to improve the partition π. The goal of these refinements is to reduce the search space by distinguishing as much as possible non symmetric vertices using vertex invariant such as vertex degree (e.g. [5]).

Figure 1 depicts the conversion of the transaction database $\mathcal{D}$ given in Example 1 to a colored undirected graph $\mathcal{G}(\mathcal{D})$. In the Figure 1, items are represented with nodes as circles (color 1) while the transaction identifiers are represented by rectangles (color 2).

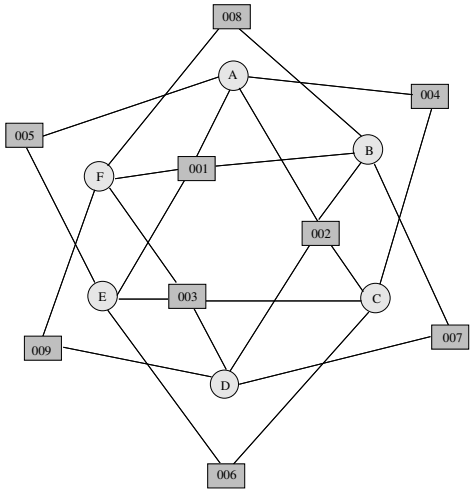

Figure 1. From transaction database $\mathcal{D}$ to colored graph $\mathcal{G}(\mathcal{D})$

Thanks to this construction, the symmetries of the transaction database correspond to the automorphisms of the colored undirected graph. Given $\mathcal{G}(\mathcal{D})$ and an initial partition π, the goal is to compute $\mathcal{A}(\mathcal{G}(\mathcal{D}))_\pi$. Using NAUTY [18], one can find the automorphisms that leave $\mathcal{G}(\mathcal{D})$ invariant. For example, $A = (A, C)(B, D)(001, 003)(005, 006)$ $(008, 009)$ is an automorphism of $\mathcal{G}(\mathcal{D})$. The corresponding symmetry is $\sigma \circ f$ where $\sigma = (A, C)(B, D)$ and $f = (001, 003)(005, 006)(008, 009)$.

III. SYMMETRY-BASED PRUNING

In this section, we show how the set of symmetries can be used to prune the search space of frequent mining problems. To this end, we consider the well known APRIORI algorithm [1], which is given in Algorithm 1. It is an algorithm for mining frequent itemsets for association rules. This algorithm is mainly based on the anti-monotonicity property: an itemset is frequent if and only if all its subsets are frequent (Proposition 1). It proceeds by a level-wise search of the frequent itemsets. Indeed, it starts by computing the frequent itemsets of size one F_1 (line 1). Then, it iteratively computes the frequent itemsets of size from 2 (F_2) to the maximal size n (F_n) such that $F_{n+1} = \emptyset$ (lines 2-8). The

basic idea consists in assuming the frequent itemsets of size $k-1$ known, and generating a set of candidates of size k by using the anti-monotonicity property (line 4).

Algorithm 1: APRIORI

Data: D: database, λ: minimal support threshold
Result: the set of all frequent itemsets

1 $F_1 \leftarrow \{$frequent itemsets of size 1$\}$;
2 **for** $(k = 2;\ F_{k-1} \neq \emptyset;\ k++)$ **do**
3 $\quad F_k \leftarrow \emptyset$;
4 $\quad C_k \leftarrow CandidatesGen(F_{k-1})$;
5 $\quad$ **for** $(c \in C_k)$ **do**
6 $\quad\quad supp(c) \leftarrow CalculSupp(c, \mathcal{D})$;
7 $\quad\quad$ **if** $(supp(c) \geq \lambda)$ **then**
8 $\quad\quad\quad F_k \leftarrow F_k \cup \{c\}$;
9 **return** $(\bigcup_k F_k)$;

We now describe the integration of our symmetry-based pruning approach into the APRIORI algorithm. It mainly consists in using symmetries to reduce the set of candidates in the same way as the anti-monotonicity property. The use of our approach of symmetry-based pruning is described in Algorithm 2.

Algorithm description:: Similarly, APRIORI$_{Sym}$ algorithm takes as input, a transaction database $\mathcal{D}$, a minimal support threshold λ, and an additional argument: the set $\mathcal{S}$ of symmetries in $\mathcal{D}$. It has as output the set of all the frequent itemsets.

Lines 1-4. This part is similar to that in APRIORI algorithm. However, we consider that the set $\mathcal{S}$ of symmetries is used to improve the function that computes the frequent itemsets of size one (line 1). For instance, we know that if $\{a\}$ is frequent (resp. not frequent) then, for all $\sigma \circ f \in \mathcal{S}$, the itemset $\{\sigma(a)\}$ is also frequent (resp. not frequent).

Lines 5-10. In this for-loop, we start by computing the support of a candidate c (line 6) and the set S_c of the candidates symmetric to c (line 7). If c is a frequent itemset (line 8), then all the elements of S_c are also frequent itemsets (line 9). Then, the elements of S_c are removed from the set of candidates (line 10).

Proposition 3: APRIORI$_{Sym}$ algorithm is correct.

Proof: It is an immediate consequence of the correctness of APRIORI algorithm and Proposition 2. ■

Algorithm 2: APRIORI$_{Sym}$

Data: D: database, λ: minimal support threshold, $\mathcal{S}$: symmetries in D
Result: the set of all frequent itemsets

1 $F_1 \leftarrow \{$frequent 1-itemsets$\}$;
2 **for** $(k = 2;\ F_{k-1} \neq \emptyset;\ k++)$ **do**
3 $\quad F_k \leftarrow \emptyset$;
4 $\quad C_k \leftarrow CandidatesGen(F_{k-1})$;
5 $\quad$ **for** $(c \in C_k)$ **do**
6 $\quad\quad supp(c) \leftarrow CalculSupp(c, \mathcal{D})$;
7 $\quad\quad S_c \leftarrow SymmGen(c, \mathcal{S})$;
8 $\quad\quad$ **if** $(supp(c) \geq \lambda)$ **then**
9 $\quad\quad\quad F_k \leftarrow F_k \cup \{c\} \cup S_c$;
10 $\quad\quad C_k = C_k \setminus S_c$;
11 **return** $(\bigcup_k F_k)$;

For instance, let $\mathcal{D}$ be a transaction database such that $\mathcal{I}_{items}(\mathcal{D}) = \{A, B, C, D\}$ and $\sigma \circ f$ is a symmetry such that $\sigma = (A, D)(B, C)$. We assume that the itemsets $\{A\}$, $\{B\}$, $\{C\}$ and $\{D\}$ are frequent. We also assume that in line 8, we obtain that the itemset $\{A, B\}$ is not frequent. By using the anti-monotonicity property, we obtain that the itemsets $\{A, B, C\}$, $\{A, B, D\}$ and $\{A, B, C, D\}$ are not frequent. Moreover, by using σ, we also obtain that the itemsets $\{C, D\} = \sigma(\{A, B\})$, and by the anti-monotonicity property we deduce that $\{A, C, D\}$ and $\{B, C, D\}$ are not frequent. This example is illustrated in Figure 2 where the circle nodes correspond to the itemsets pruned by our symmetry-based pruning approach.

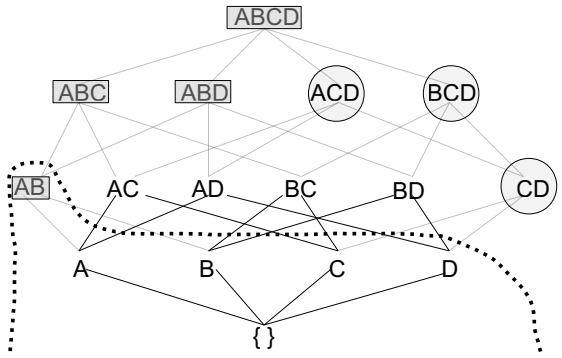

Figure 2. Symmetry Pruning

IV. EXPERIMENTAL EVALUATION

A. Experimental setup

We present in this section some experimental results highlighting the usefulness and the practical benefits of using symmetry-based pruning in itemset mining tasks.

Experiments were performed on several datasets from the

UCI repository[1] and the real dataset BMS-WebView-2 [28], this dataset contains click-stream data from e-commerce websites. Table II provides for each dataset its number of transaction (#trans), its total number of distinct items (#items) and its density: the density of a dataset is defined as the average transaction size of the dataset divided by #items. A dataset is said to be dense, when most transactions tend to be similar i.e they have about the same size and contain mostly the same items.

dataset	#trans	#items	density
Zoo	101	43	39%
Mushroom	8 142	117	18%
Australian	690	55	25%
Solar flare	323	40	32%
BMS-WebView-2	77 512	3 341	0.14%
Letter-recognition	20 000	74	23%

Table II
DESCRIPTION OF THE DATASETS

For symmetry detection in datasets, we use Saucy[2], a new implementation of the Nauty system. It was first proposed in [2] and significantly improved in [5]. In most cases, Saucy enables us to detect symmetries in common UCI databases in less than a second. Table III summarizes symmetry extraction time on the datasets we use in our experiments.

dataset	#sym	time (s)
Zoo	2	0.02
Mushroom	10	0.94
Australian	2	0.06
Solar flare	2	0.05
BMS-WebView-2	10	4.31
Letter-recognition	0	2.12

Table III
SYMMETRY EXTRACTION TIME (SECONDS)

Symmetries are present in the datasets regardless itemset mining tasks and measures thresholds, thus the symmetry-based pruning approach we propose in this paper proceeds in two steps. First, we extract symmetries using Saucy. Then, APRIORI$_{Sym}$ takes as input the set of symmetries in $\mathcal{D}$ in addition to the input of APRIORI which is composed by the dataset $\mathcal{D}$ and a minimal support threshold.

In order to quantify the runtime acceleration obtained with and without symmetry-based pruning approach, we define the mining time acceleration measure $\mathcal{M}_{\mathcal{TA}}$ as follows:

$$\mathcal{M}_{TA} = \frac{\text{runtime(APRIORI)} - \text{runtime(APRIORI}_{Sym})}{\text{runtime(APRIORI)}}$$

[1]http://www.ics.uci.edu/~mlearn/MLRepository.html

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

This measure quantifies the runtime acceleration obtained when considering symmetries during the itemset mining process. It does not consider the symmetry extraction time because symmetry extraction is done only once (offline) for a database and could be used after whatever are the itemset mining tasks and the thresholds. The symmetry detection time is less than 5 seconds on the considered datasets.

We also show how using our symmetry-based pruning approach reduces significantly the number of frequency computation operations that we call in this paper database scans. For this task, we define the following database scan reduction measure:

$$\mathcal{M}_{SR} = \frac{\text{\#db scans(APRIORI)} - \text{\#db scans(APRIORI}_{Sym})}{\text{\#db scans(APRIORI)}}$$

B. Experimental results

For our experiments, we take as an example the classical frequent itemset mining algorithm APRIORI (cf. Algorithm 1).

We mainly focus on the contribution of symmetry-based pruning on reducing the number of database scans and reducing the frequent itemsets mining algorithm runtime.

Our first observation is that, in real datasets, we find few symmetries (cf. Table III). This is an expected result as, in most cases symmetries in real datasets are rather local ones (symmetries on a portion of $\mathcal{D}$) than global (symmetries on the whole $\mathcal{D}$)

Experiments conducted in this section, show that these few symmetries are enough to obtain interesting pruning results.

Table IV shows the efficiency of integrating symmetry-based pruning into the APRIORI algorithm even if we have only few detected symmetries in the database. #freq$_{Sym}$ (resp. #infreq$_{Sym}$) denotes the number of frequent (resp. infrequent) patterns deduced using symmetries. For example, for Australian dataset containing only two symmetries, and 1% as a minimal frequency threshold ($minfr$), we observe the following: using APRIORI, the database needs to be scanned about 480 thousand times in order to check the property of candidate itemsets (whether a candidate itemset is frequent or not). Whereas, using APRIORI$_{Sym}$, the properties of about 115 thousand candidate itemsets (#freq$_{Sym}$+#infreq$_{Sym}$) are deduced due to symmetries and no more scan of the database is needed (cf. Table IV). Finally we note that, in this case, APRIORI$_{Sym}$ is 23% faster than APRIORI.

Another important result of our experiments is that the APRIORI$_{sym}$ algorithm preserves the same efficiency as APRIORI when dealing with datasets which do not contain symmetries (cf. Table IV, letter-recognition dataset). Thus, our symmetry-based pruning approach could be generalized to be used with other itemset mining algorithms allowing them to take benefit from symmetries whether they exist. We prove in section V the feasibility

Dataset	$minfr$	Sym Pruning	#db scans	$\mathcal{M}_{SR}$	#freq	#freq$_{Sym}$	#infreq$_{Sym}$	$\mathcal{M}_{\mathcal{TA}}$
Australian	1%	−−	479 402	24%	426 763	−−	−−	23%
		✓	364 822			100 961	13 613	
	5%	−−	28 535	22%	20 386	−−	−−	22%
		✓	22 288			4.461	1 886	
Solar-Flare	1%	−−	147 270	9%	145 893	−−	−−	8%
		✓	133 056			14 128	1 291	
	15%	−−	5 684	8%	5 495	−−	−−	9%
		✓	5 203			448	33	
Mushroom	5%	−−	3 764 532	0.4%	3 755 511	−−	−−	6%
		✓	3 748 084			16 384	8 957	
Zoo	5%	−−	587 782	20%	486 099	−−	−−	15%
		✓	487 224			100 480	78	
	15%	−−	103 318	23%	102 440	−−	−−	17%
		✓	79 492			23 776	50	
BMS-WebView-2	1%	−−	4 908	0%	81	−−	−−	0%
		✓	4 908			0	0	
Letter-recognition	5%	−−	32 680	0%	15 719	−−	−−	0%
		✓	32 680			0	0	

Table IV
EXPERIMENTAL RESULTS

of this generalization when dealing with interestingness predicates which are stable by symmetry.

Finally, we notice that for BMS-WebView-2 dataset, containing 10 symmetries, we do not observe any effect of these symmetries on the search space. In fact, the pruning efficiency is not proportional to the number of symmetries in the dataset: if symmetries appear in only few transactions, as is the case for BMS-WebView-2, then all concerned itemsets are not frequent and are pruned by APRIORI.

C. Discussion

Our experiments show that the symmetry pruning approach we present in this paper could be integrated into APRIORI algorithm in order to reduce the number of the costly frequency computation operations. If the dataset does not contain symmetries, APRIORI$_{sym}$ has the same performance as APRIORI. Symmetries are extracted only once for each dataset, thus the obtained mining time acceleration is always greater or equal to zero ($\mathcal{M}_{TA}$(APRIORI $_{Sym}$,APRIORI) ≥ 0).

Symmetries are then important properties that we should consider in order to improve current itemset mining algorithms on some families of datasets.

V. A GENERALIZATION OF THE USE OF SYMMETRIES

In this section, we propose a generalization of our approach to different problems of knowledge discovery in the transaction databases. The model that we consider is inspired from [17]. Let $\mathcal{D}$ be a transaction database, $\mathcal{L}$ a language for expressing considered knowledges (a set of patterns), and q an interestingness predicate for evaluating whether an element of $\mathcal{L}$ is an interesting knowledge (or pattern). In this context, we consider that a mining task consists in computing the following set:

$$\mathcal{TH}(\mathcal{D}, \mathcal{L}, q) = \{a \in \mathcal{L} \mid q(\mathcal{D}, a) \ is \ true\}$$

For instance, the problem of enumerating frequent itemsets can be obtained by defining $\mathcal{L}$ as all the non-empty subsets of the set of items in the considered transaction database, and q as the predicate expressing that an itemset has a support higher than the considered threshold.

An interestingness predicate q is said to be stable by symmetry if, for all transaction database $\mathcal{D}$ and $a \in \mathcal{L}$, and for all symmetry σ, $q(\mathcal{D}, a)$ is true if and only if $q(\mathcal{D}, \sigma(a))$ is true.

One can easily see that in the case of an interestingness predicate which is stable by symmetry, the set of symmetries can be used to prune the search space. Indeed, if an element a of the considered language is interesting (resp. not interesting), then $\sigma(a)$ is also interesting (resp. not interesting), for each symmetry σ. Thus, our approach can be integrated to different mining algorithms where the interestingness predicates are stable by symmetry.

In order to illustrate our generalization, let us consider the problem of enumerating closed itemsets. We recall that an itemset I in a transaction database $\mathcal{D}$ is *closed* if and only if, for all itemset J such that $I \subset J$, $\mathcal{S}upp(I, \mathcal{D}) > \mathcal{S}upp(J, \mathcal{D})$ holds. Let $\mathcal{D}$ be a transaction database, I a closed itemset in $\mathcal{D}$ and σ a symmetry of $\mathcal{D}$. Since σ is a symmetry, $\mathcal{S}upp(\sigma(I), \mathcal{D}) = \mathcal{S}upp(I, \mathcal{D})$ holds. We now prove that $\sigma(I)$ is also a closed itemset in $\mathcal{D}$. Let I' be an itemset such that $\sigma(I) \subset I'$ and $\mathcal{S}upp(\sigma(I), \mathcal{D}) \leq \mathcal{S}upp(I', \mathcal{D})$. One can see that $I \subset \sigma^{-1}(I')$ and $\mathcal{S}upp(I, \mathcal{D}) \leq \mathcal{S}upp(\sigma^{-1}(I'), \mathcal{D})$. Thus, we get a contradiction and we deduce that $\sigma(I)$ is a closed itemset. Therefore, our symmetry-based pruning

approach can also be used in the problem of enumerating closed itemsets.

VI. Related works

In the context of data mining, symmetries are structural knowledge that can be efficiently exploited in reducing the size of the search space or the size of the outputs. Symmetries can also be seen as relevant knowledge and extracting symmetries can itself be a data mining task. As structural knowledge, it may for instance help a data analyst in understanding some relationships between items. Note that symmetries are hidden knowledge requiring to explore empirical data and their extraction depends on the types of symmetries one is searching for. The semantics that can be associated with each symmetry pattern in a given application field represents a high level piece of knowledge revealing some properties of the underlying data. For example, in frequent graph mining problems, symmetric items (subgraphs) are patterns satisfying some properties with respect to the frequent graph mining problem. In a transaction dataset, symmetric items may reveal for example that such items play the same role. Exploiting symmetries has received an increasing attention in artificial intelligence. In particular, symmetries are widely studied in satisfiability (e.g. [3], [4]) and constraint satisfaction problems (e.g. [23]). In data mining, symmetries are studied mainly in frequent graph mining [26], [6], [21], [10].

In the frequent sub-graph mining problem, many works use symmetry-related concepts in the mining process. For instance, in [6] the authors make use of sub-graph symmetries in order to prune the search space during the candidate enumeration steps. The concept of symmetries is used in [26] along with constraints on sub-graphs widths in order to mine relevant sub-graphs. Symmetries (in the sense of automorphisms) are used in this work as interestingness measure. In [7], the author proposes approaches for mining symmetries in social networks. Another work using the symmetry concept related with graph mining is the one of [27] where automorphism-based partition of graphs provide some structural informations. Many other works use the concept of symmetries in graph related problems but for other purposes rather than graph-mining.

In data clustering, many works use symmetry-related concepts for different purposes. For instance, the authors in [22] deal with hierarchical clustering of massive data to mine symmetries and other interesting patterns. In this work, symmetric patterns are understood as a structure which reveals invariants and intrinsic properties in data. In [21], the author provides a unifying view on symmetry in data mining and data analysis based on hierarchy. The authors in [16] use symmetry-related concepts in relational clustering.

For the itemset mining problem, symmetries have not yet received much attention. One can mention two related works addressing a particular case of general symmetries considered in this paper [20], [19]. Indeed, this restricted form of symmetry, called pairwise items symmetry, is obtained by exchanging two items, while leaving the remaining items unchanged. This is clearly a restriction of the general symmetry principle. In [20], an efficient ZBDD algorithm for detecting pairwise symmetries is proposed and the property of symmetric items in transaction database are discussed. In this work, the author deals only with a special case of symmetries, namely only pairs of symmetric items are considered. Here the concept of symmetric item refers to pairs of items that if permuted, the dataset remains unchanged and consequently, the mined frequent itemsets are the same.

Pairwise items symmetries, called clone items, have been proposed by Medina and Nourine [8] to explain why sometimes, the number of rules of a minimum cover of a relation is exponential in the number of items of the relation. More recently, in the context of constraint programming for k-pattern set mining, Guns et al. [10] used symmetry breaking constraints to impose a strict ordering on the patterns in the pattern set. The symmetry concept in this work is more related to symmetries in constraint satisfaction problems than to symmetric items in the sense of support.

VII. Conclusion and Future Works

We have proposed an efficient approach for taking advantage of symmetries in transaction databases in order to improve search space pruning in itemset mining algorithms. Symmetries are structural knowledge showing interesting properties for some enumeration and search problems. In this paper, we showed how such knowledge can be efficiently used in the pruning step of the well-known Apriori algorithm in order to limit the number of databases scans. We provided an experimental evaluation showing that on some datasets, the number of database scans has been reduced by about 24% in comparison with the standard Apriori algorithm. This approach is very interesting since computing symmetries can be done in an offline mode and only once. Moreover, we showed that symmetry-based pruning can be extended to other data-mining and enumeration tasks as far as the underlying predicates are stable by symmetry.

Future works will address the extension of symmetry-based pruning to other data mining problems such as sequences, trees and graphs mining. Other problems such as clustering can benefit from this important symmetry properties. We also plan to investigate other form of symmetries such as local or conditional symmetries i.e. symmetries on the portion of the transaction database.

References

[1] Rakesh Agrawal and Ramakrishnan Srikant. Fast algorithms for mining association rules in large databases. In Proceedings of 20th International Conference on Very Large Data Bases, September 12-15, 1994, Santiago de Chile, Chile (VLDB'94), pages 487–499, 1994.

[2] Fadi A. Aloul, Arathi Ramani, Igor L. Markov, and Karem A. Sakallah. Solving difficult instances of boolean satisfiability in the presence of symmetry. Transactions on Computer Aided Design, 2003.

[3] B. Benhamou and L. Sas. Tractability through symmetries in propositional calculus. Journal of Automated Reasoning, 12(1):89–102, 1994.

[4] J. Crawford, M. L. Ginsberg, E. Luck, and A. Roy. Symmetry-breaking predicates for search problems. In KR'96, pages 148–159. 1996.

[5] Paul T. Darga, Karem A. Sakallah, and Igor L. Markov. Faster symmetry discovery using sparsity of symmetries. In DAC, pages 149–154, 2008.

[6] Christian Desrosiers, Philippe Galinier, Pierre Hansen, and Alain Hertz. Improving frequent subgraph mining in the presence of symmetry. In Proceedings of Mining and Learning with Graphs (MLG 2007), 2007.

[7] Angel Garrido. Symmetry in complex networks. Symmetry, 3(1):1–15, 2011.

[8] Alain Gély, Raoul Medina, Lhouari Nourine, and Yoan Renaud. Uncovering and reducing hidden combinatorics in guigues-duquenne bases. In ICFCA'05, pages 235–248, 2005.

[9] Ian P. Gent and Barbara Smith. Symmetry breaking in constraint programming. In Proceedings of the Thirteenth European Conference on Artificial Intelligence(ECAI'00), pages 599–603, 2000.

[10] Tias Guns, Siegfried Nijssen, and Luc de Raedt. k-pattern set mining under constraints. IEEE TKDE, 99(PrePrints), 2011.

[11] Jun Huan, Wei Wang, and Jan Prins. Efficient mining of frequent subgraphs in the presence of isomorphism. In Proceedings of the 3rd IEEE International Conference on Data Mining (ICDM 2003), pages 549–552, 2003.

[12] Saïd Jabbour, Lakhdar Sais, Yakoub Salhi, and Karim Tabia. Symmetries in itemset mining. In 20th European Conference on Artificial Intelligence (ECAI'2012), pages 432–437, 2012.

[13] Tommi A. Junttila and Petteri Kaski. Engineering an efficient canonical labeling tool for large and sparse graphs. In ALENEX, 2007.

[14] Balakrishnan Krishnamurthy. Shorts proofs for tricky formulas. Acta Informatica, 22:253–275, 1985.

[15] Leo Liberti. Reformulations in mathematical programming: automatic symmetry detection and exploitation. Math. Program., 131(1-2):273–304, 2012.

[16] Bo Long, Zhongfei (Mark) Zhang, Xiaoyun Wu, and Philip S. Yu. Relational clustering by symmetric convex coding. In Proceedings of the 24th international conference on Machine learning, ICML '07, pages 569–576, New York, NY, USA, 2007. ACM.

[17] Heikki Mannila and Hannu Toivonen. Levelwise search and borders of theories in knowledge discovery. Data Min. Knowl. Discov., 1(3):241–258, 1997.

[18] B. D. McKay. Practical graph isomorphism. Congressus Numerantium, (30):47–87, 1981.

[19] Raoul Medina, Caroline Noyer, and Olivier Raynaud. Efficient algorithms for clone items detection. In CLA'05, pages 70–81, 2005.

[20] Shin-ichi Minato. Symmetric item set mining based on zero-suppressed bdds. In DS'06, pages 321–326, 2006.

[21] Fionn Murtagh. Symmetry in data mining and analysis: A unifying view based on hierarchy. Proceedings of the Steklov Institute of Mathematics, 265:177–198, 2009.

[22] Fionn Murtagh and Pedro Contreras. Hierarchical clustering for finding symmetries and other patterns in massive, high dimensional datasets. CoRR, abs/1005.2638, 2010.

[23] Jean-Francois Puget. On the satisfiability of symmetrical constrained satisfaction problems. In ISMIS, pages 350–361, 1993.

[24] T. Imielinski R. Agrawal and A. N. Swami. Mining association rules between sets of items in large databases. In ACM SIGMOD International Conference on Management of Data, pages 207–216, Baltimore, 1993. ACM Press.

[25] A. Tiwari, RK Gupta, and DP Agrawal. A survey on frequent pattern mining: Current status and challenging issues. Inform. Technol. J, 9:1278–1293, 2010.

[26] Natalia Vanetik. Mining graphs with constraints on symmetry and diameter. In International Workshops on Web-Age Information Management - WAIM 2010, pages 1–12, 2010.

[27] Yang-Hua Xiao, Wen-Tao Wu, Hui Wang, Momiao Xiong, and Wei Wang. Symmetry-based structure entropy of complex networks. Physica A: Statistical Mechanics and its Applications, 387(11):2611–2619, April 2008.

[28] Zijian Zheng, Ron Kohavi, and Llew Mason. Real world performance of association rule algorithms. In Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, KDD '01, pages 401–406, New York, NY, USA, 2001. ACM.