# Symmetries in Itemset Mining

**Said Jabbour** and **Lakhdar Sais** and **Yakoub Salhi** and **Karim Tabia**[1]

**Abstract.** In this paper, we describe a new framework for breaking symmetries in itemset mining problems. Symmetries are permutations between items that leave invariant the transaction database. Such kind of structural knowledge induces a partition of the search space into equivalent classes of symmetrical itemsets. Our proposed framework aims to reduce the search space of possible interesting itemsets by detecting and breaking symmetries between items. Firstly, we address symmetry discovery in transaction databases. Secondly, we propose two different approaches to break symmetries in a preprocessing step by rewriting the transaction database. This approach can be seen as an original extension of the symmetry breaking framework widely used in propositional satisfiability and constraint satisfaction problems. Finally, we show that Apriori-like algorithms can be enhanced by dynamic symmetry reasoning. Our experiments clearly show that several itemset mining instances taken from the available datasets contain such symmetries. We also provide experimental evidence that breaking such symmetries reduces the size of the output on some families of instances.

## 1 Introduction

The problem of mining frequent itemsets is well-known and essential in data mining, knowledge discovery and data analysis. It has applications in various fields and becomes fundamental for data analysis as datasets and datastores are becoming very large. Since the first article of Agrawal [15] on association rules and itemset mining, the huge number of works, challenges, datasets and projects show the actual interest in this problem (see [17] for a recent survey of works addressing this problem). Note that several data mining tasks are closely related to the itemset mining problem such as the ones of association rule mining, frequent pattern mining in sequence data, data clustering, episode mining, etc. Important progress has been achieved for data mining and knowledge discovery in terms of implementations, platforms, libraries, etc. Nevertheless, the majority of works developed solutions and tools specifically tuned and designed to achieve very specific goals on specific data mining tasks. The algorithmic issues represent the most important problem for the majority of works. As pointed out in [17], lot of works deal with designing highly scalable itemset mining algorithms for large scale datasets. The existing algorithms mainly differ in the way they explore the itemset search space, how the anti-monotonicity property is used and how the dataset is handled. Another important problem of itemset mining and data mining problems in general, concerns the huge size of the output, from which it is difficult to retrieve useful information. Consequently, for practical data mining, it is important to reduce the size of the output, by exploiting the structure of the itemsets data. Computing for example, closed, maximal, condensed, discriminative itemset patterns are some of the well-known and useful techniques.

Symmetry between items is another kind of structural information that might be recovered from the transaction databases and used to further reduce the size of the output by limiting the search space. Symmetries are a fundamental concept in computer science, mathematics, physics and many other domains. Many human artifacts (e.g. classroom in a university, aircraft seats, circuit patterns) and entities in nature (e.g. plants, molecules, DNA sequences, atoms) exhibits symmetries. Such symmetries allow us to reason and to understand more complex entities and systems. For example, this kind of structures has been widely exploited in constraint satisfaction (CSP) and propositional satisfiability (SAT) problems. As far as we know, symmetry reasoning has been first introduced by Krishnamurthy [9] 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 [2], the authors showed how dynamic symmetry detection and elimination can lead to significant improvements of several automated deduction techniques. In constraint satisfaction problems, a weaker form of symmetry called value interchangeability is first introduced by E. Freuder in [6], while variable and value symmetry are studied in [14]. Symmetry breaking using additional constraints has been proposed independently by James Crawford in the context of first order [3] and propositional [4] logic theories and by Jean-François Puget in CSP [14]. More recent contributions clearly show that symmetry remains an active research area in both CSP and SAT.

To the best of our knowledge, in the context of data mining and particularly for the itemset mining problem, symmetries have not received much attention. Let us mention two related works addressing a particular case of general symmetries considered in this paper [13, 12]. 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 [13], an efficient ZBDD algorithm for detecting pairwise symmetries is proposed and the property of symmetric items in transaction database are discussed. Pairwise items symmetries, called clone items, have been proposed by Medina and Nourine [7] 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. [8] used symmetry breaking constraints to impose a strict ordering on the patterns in the pattern set.

In this paper, we propose a theoretical framework for dealing with general symmetries in itemset mining problems. We first propose an adaptation of the symmetry detection method usually used in propositional satisfiability [3, 1] to transaction databases. As this hidden structure can be very helpful for reducing the search space, we show how they can be integrated dynamically in Apriori-like al-

[1] Univ Lille Nord de France, F-59000 Lille, France. UArtois, CRIL UMR CNRS 8188, F-62300 Lens, France. {jabbour, sais, salhi, tabia}@cril.univ-artois.fr

gorithms to prune the set of possible candidate patterns. Then, we propose two symmetry breaking approaches to eliminate symmetries in transaction databases in a preprocessing step. Our proposed symmetry breaking methods eliminate items from the original transaction database. The frequent itemsets generated using the new transaction database together with the symmetry group can be used to retrieve the whole set of frequent itemsets of the original transaction database. This can be seen as a form of condensed representation of the output. A condensed representation is originally proposed by Mannila and Toivenen in [10]. For frequent itemsets, a condensed representation is a collection of itemsets that still contains the same information.

On the practical side and to show the usefulness of exploiting symmetries in itemset mining problems, we present an experimental evaluation of both symmetry detection and symmetry breaking on some benchmark instances.

## 2 Preliminary definitions and notations

Let $\mathcal{I}$ be a set of *items*. A set $I \subseteq \mathcal{I}$ is called an *itemset*. A *transaction* is a couple $(tid, I)$ where $tid$ is the *transaction identifier* and $I$ is an itemset. A *transaction database* is a finite set of transactions over $\mathcal{I}$ where for each two different transactions, they do not have the same transaction 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 $I$: $\mathcal{C}(I, \mathcal{D}) = \{tid \mid (tid, J) \in \mathcal{D} \text{ and } I \subseteq J\}$. The *support* of an itemset $I$ in $\mathcal{D}$ is defined by: $\mathcal{S}upp(I, \mathcal{D}) = |\mathcal{C}(I, \mathcal{D})|$. Moreover, the frequency of $I$ in $\mathcal{D}$ is defined by: $\mathcal{F}(I, \mathcal{D}) = \frac{\mathcal{S}upp(I,\mathcal{D})}{|\mathcal{D}|}$.

**Example 1** *Let us consider the following transaction database over the set of items* $\mathcal{I} = \{Spaghetti, Tomato, Parmesan, Beef, Olive\ oil, Mozzarella, Chili\ pepper, Anchovies, Eggs\}$*:*

| tid | itemset |
|---|---|
| 001 | $Spaghetti, Olive\ oil, Tomato, Mozzarella$ |
| 002 | $Spaghetti, Parmesan, Olive\ oil, Beef$ |
| 003 | $Chili\ pepper, Anchovies$ |
| 004 | $Eggs$ |

**Table 1.** An example of transaction database $\mathcal{D}$

*For instance, we have* $\mathcal{S}upp(\{Spaghetti, Olive\ \ oil\}, \mathcal{D})$= $|\{001, 002\}|$=2 *and* $\mathcal{F}(\{Spaghetti, Olive\ oil\}, \mathcal{D})$=0.5.

Let $\mathcal{D}$ be a transaction database over $\mathcal{I}$ and $\lambda$ a minimal support threshold.

**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:*

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

**Definition 2 (Transaction Renaming)** *Let $\mathcal{D}$ be a transaction database. A renaming $f$ over $\mathcal{T}_{id}(\mathcal{D})$ is a bijective mapping from $\mathcal{T}_{id}(\mathcal{D})$ to $\mathcal{T}_{id}(\mathcal{D})$.*

We can extend a renaming $f$ to $\mathcal{D}$ as follows: $f(\mathcal{D}) = \{(f(tid), I)|(tid, I) \in \mathcal{D}\}$.

## 3 Symmetry in Frequent Itemset Mining

**Definition 3 (Permutation)** *A permutation $\sigma$ over $\mathcal{I}$ is a bijective mapping from $\mathcal{I}$ to $\mathcal{I}$.*

Let $\mathcal{D}$ be a transaction database. We can extend a permutation $\sigma$ to $\mathcal{D}$ as follows: $\sigma(\mathcal{D}) = \{(tid, \sigma(I))|(tid, I) \in \mathcal{D}\}$ where $\sigma(I) = \{\sigma(a)|a \in I\}$.

We denote by $\mathcal{P}(\mathcal{I})$ the set of all the permutations over $\mathcal{I}$ and $\circ$ is the composition operation over the elements of $\mathcal{P}(\mathcal{I})$. It is easy to see that $(\mathcal{P}(\mathcal{I}), \circ)$ is a group where the identity permutation is a neutral element. $(S', \circ)$ is a sub-group of the group $(\mathcal{P}(\mathcal{I}), \circ)$ if and only if $(S', \circ)$ is a group and $S' \subseteq \mathcal{P}(\mathcal{I})$. A *generating set of the group* $(S, \circ)$ is a subset $\Sigma$ of $S$ such that every element of the group can be obtained from the combination of finitely many elements of $\Sigma$ and their inverses, and we write $S =< \Sigma >$.

Each permutation $\sigma$ 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 $\mathcal{I}$ such that $\sigma(a_j) = a_{j+1}$ for $j = 1, \dots, k-1$, and $\sigma(a_k) = a_1$. In the sequel, for clarity reasons, we omit cycles of the form $(a, a)$ in the description of permutations and symmetries,.

**Definition 4 (Orbit)** *Let $(S, \circ)$ be a sub-group of $(\mathcal{P}(\mathcal{I}), \circ)$. The orbit $a^S$ of an item $a \in \mathcal{I}$ on which $(S, \circ)$ acts is $a^S = \{\sigma(a)|\sigma \in S\}$.*

When there is no ambiguity, we note $a^S$ simply $[a]$.

**Definition 5 (Symmetry)** *Let $\mathcal{D}$ be a transaction database. A symmetry of $\mathcal{D}$ is a permutation $\sigma \in \mathcal{P}(\mathcal{I})$ such that there exists a transaction renaming $f$ over $\mathcal{T}_{id}(\mathcal{D})$ where $\sigma(\mathcal{D}) = f(\mathcal{D})$ i.e. $f^{-1}(\sigma(\mathcal{D})) = \mathcal{D}$.*

**Example 2** *Let us consider again the transaction database given in Table 1.* $\sigma = (Tomato, Parmesan)(Mozzarella, Beef)$ $(Chili\ pepper, Anchovies)$ *is a symmetry because $\sigma(\mathcal{D}) = f(\mathcal{D})$ where $f$ is a transaction renaming defined as follows:*

$$f(x) = \begin{cases} 002 & \text{if x=001} \\ 001 & \text{if x=002} \\ x & \text{otherwise} \end{cases}$$

Let $\mathcal{S}(\mathcal{D})$ be the set of all the symmetries. One can see that $(\mathcal{S}(\mathcal{D}), \circ)$ is a sub-group of $(\mathcal{P}(\mathcal{I}), \circ)$.

**Definition 6** *Let $\mathcal{D}$ be a transaction database. Two items $a, b \in \mathcal{I}_{items}(\mathcal{D})$ are symmetric if there exists a symmetry $\sigma$ of $\mathcal{D}$ such that $\sigma(a) = b$.*

The following proposition is immediate:

**Proposition 2** *Let $\mathcal{D}$ be a transaction database and $a \in \mathcal{I}_{items}(\mathcal{D})$. All the items in $a^{\mathcal{S}(\mathcal{D})}$ are symmetrical two by two.*

**Proposition 3** *Let $\mathcal{D}$ be a transaction database, $\sigma$ a symmetry of $\mathcal{D}$, $\lambda$ 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 there exists a transaction renaming $f$ such that $\sigma(\mathcal{D}) = f(\mathcal{D})$, it is easy to see that $\sigma(I) \in \mathcal{FIM}(f(\mathcal{D}), \lambda)$ and consequently $\sigma(I) \in \mathcal{FIM}(\mathcal{D}, \lambda)$. □

## 4 Symmetry Detection in Transaction Databases

In this section, we describe symmetry detection in transaction databases. One of the most popular means of discovering symmetries of a problem is to first convert the problem into a graph, and employ a general-purpose graph symmetry tool to uncover the symmetries [1]. These symmetries can then be reflected back into the original problem. The oldest and most established graph symmetry program is McKays Nauty [11]. Most of the available symmetry detection tools convert the original problem into a colored undirected graph, where vertices are labeled with colors. Such colored vertices are considered when searching for automorphism on the graph (i.e. vertices with different colors can not be permuted to each other).

**Definition 7** *A colored undirected graph is a triplet $\mathcal{G} = (V, E, \eta)$ with vertex set $V$ and edge set $E \in V^2$ and $\eta$ is a function from $V$ to $N$ that associates a positive integer (a color) to each vertex.*

**Definition 8** *A permutation $\sigma$ of $V$ is a symmetry of $\mathcal{G}$ iff $\sigma(\mathcal{G}) = \mathcal{G}$ or equivalently $\sigma(E) = E$.*

**Definition 9** *A symmetry $\sigma$ of $\mathcal{G}$ respects a partition $\pi$ of $V$ if for each $v \in V$, $v$ and $\sigma(v)$ belong to the same cell of $\pi$. The set of all symmetries of $\mathcal{G}$ with respect to a partition $\pi$ is called the automorphism group of $\mathcal{G}$ under $\pi$ and is denoted $\mathcal{A}(\mathcal{G})_\pi$.*

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

**Definition 10 (From $\mathcal{D}$ to $\mathcal{G}$)** *Let $\mathcal{I}$ be a set of items and $\mathcal{D}$ a transaction database over $\mathcal{I}$. 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 $\forall v \in V$,*

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

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 $\pi$. 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]).

The conversion of the transaction database $\mathcal{D}$ given in Example 1 to a colored undirected graph $\mathcal{G}(\mathcal{D})$ is depicted in Figure 1. For simplicity, we use the first letter of each item name in lower case (e.g. e for Eggs). In the figure we distinguish two kinds of nodes, items represented by circles (color 1) and transaction identifiers represented by rectangles (color 2).

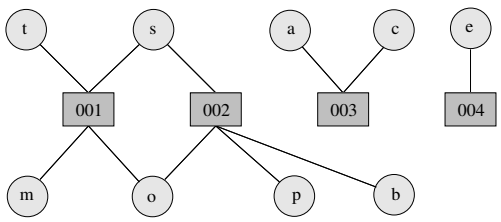


**Figure 1.** From transaction database $\mathcal{D}$ to colored graph $\mathcal{G}(\mathcal{D})$

With this construction, there is a one to one mapping between symmetries in the transaction database and the automorphisms of the colored undirected graph. Given $\mathcal{G}(\mathcal{D})$ and an initial partition $\pi$, the goal is to compute $\mathcal{A}(\mathcal{G}(\mathcal{D}))_\pi$. Using NAUTY [11], one can find the automorphisms that leave $\mathcal{G}(\mathcal{D})$ invariant. For example, $A = (t,p)(m,b)(c,a)(o)(s)(e)$ $[(001,002)(003)(004)]$ is an automorphism of $\mathcal{G}(\mathcal{D})$. The corresponding symmetry $\sigma$ is obtained from $A$ by projection on the set of items $\mathcal{I}$ of $\mathcal{D}$.

## 5 Symmetry Pruning

In this section, we show how symmetries can be used by classical enumeration algorithms such as Apriori to reduce the set of possible candidate patterns.

The Apriori algorithm is an algorithm for mining frequent itemsets for association rules [15]. It proceeds by a level-wise search of the elements of $\mathcal{FIM}(\mathcal{D}, \lambda)$. Namely, it starts by computing the elements of $\mathcal{FIM}(\mathcal{D}, \lambda)$ of size one. Then, assuming the elements of $\mathcal{FIM}(\mathcal{D}, \lambda)$ of size $n$ known, it computes a set of candidates of size $n+1$ so that $I$ is a candidate if and only if all its subsets are in $\mathcal{FIM}(\mathcal{D}, \lambda)$. This procedure is iterated until no more candidate is found. Let $\mathcal{D}$ be a transaction database such that $\mathcal{I}_{item}(\mathcal{D}) = \{A, B, C\}$, $\sigma = (B, C)$ and $\sigma' = (A, C)(B, D)$ two symmetries of $\mathcal{D}$ and $\lambda$ a minimal support threshold. We consider that we are in an intermediary step of an Apriori like algorithm and we have $\{A\}, \{B\}, \{C\}, \{D\} \in \mathcal{FIM}(\mathcal{D}, \lambda)$ and $\{A, B\} \notin \mathcal{FIM}(\mathcal{D}, \lambda)$. Using the anti-monotonicity property (Proposition 1), we know that for all itemset $I$ containing $\{A, B\}$, $I \notin \mathcal{FIM}(\mathcal{D}, \lambda)$, then we obtain $\{A, B, C\}, \{A, B, D\}, \{A, B, C, D\} \notin \mathcal{FIM}(\mathcal{D}, \lambda)$. Moreover, using the symmetries $\sigma$ and $\sigma'$, we know that $\{A, C\} \notin \mathcal{FIM}(\mathcal{D}, \lambda)$ and $\{C, D\} \notin \mathcal{FIM}(\mathcal{D}, \lambda)$. Indeed, this is a consequence of Proposition 3, $\sigma(\{A, B\}) = \{A, C\}$ and $\sigma'(\{A, B\}) = \{C, D\}$. Finally, using the anti-monotonicity property, we deduce that also $\{A, C, D\}, \{B, C, D\} \notin \mathcal{FIM}(\mathcal{D}, \lambda)$. This is illustrated in Figure 2 where the inclined dashed circles (nodes) correspond to the itemsets pruned by symmetry.

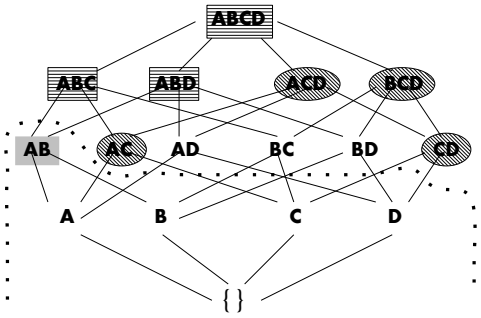


**Figure 2.** Symmetry pruning of the search space

## 6 Symmetry Breaking

In this section, we propose two approaches for breaking symmetries in a preprocessing step. Our proposed symmetry breaking methods eliminate items from the original transaction database. The frequent itemsets generated using the new transaction database together with the symmetry group can be used to retrieve the whole set of frequent itemsets of the original transaction database. This can be seen as a form of condensed representation of the output.

## 6.1 Orbit-based Symmetry Breaking

**Definition 11** *Let $D$ be a transaction database and $[a]$ an orbit following $(\mathcal{S}(\mathcal{D}), \circ)$. $\mathcal{D}^a$ is the transaction database obtained from $\mathcal{D}$ as follows: $\forall b \in [a]$ such that $b \neq a$ and for every transaction $T = (tid, I) \in \mathcal{D}$, if $b \in I$ and $a \notin I$, then $b$ is removed from $I$ in the transaction $T$.*

Let $D$ be a transaction database, $a$ an item and $\lambda$ a minimal support threshold, we denote by $\mathcal{FIM}^a(\mathcal{D}, \lambda)$ all the elements of $\mathcal{FIM}(\mathcal{D}, \lambda)$ containing $a$.

**Proposition 4** *Let $D$ be a transaction database, $a, b \in \mathcal{I}_{items}(\mathcal{D})$ such that they are in the same orbit following the group $(\mathcal{S}(\mathcal{D}), \circ)$ and $\lambda$ a minimal support threshold. There exists a symmetry $\sigma$ such that $\mathcal{FIM}^b(\mathcal{D}, \lambda) = \sigma(\mathcal{FIM}^a(\mathcal{D}, \lambda))$.*

**Proof:** Since $a$ and $b$ are in the same orbit, there exists a symmetry $\sigma$ such that $b = \sigma(a)$. Using Proposition 3, we have (1) $\mathcal{FIM}^b(\mathcal{D}, \lambda) \supseteq \sigma(\mathcal{FIM}^a(\mathcal{D}, \lambda))$ and (2) $\mathcal{FIM}^a(\mathcal{D}, \lambda) \supseteq \sigma^{-1}(\mathcal{FIM}^b(\mathcal{D}, \lambda))$. Using (2), we have $\sigma(\mathcal{FIM}^a(\mathcal{D}, \lambda)) \supseteq \mathcal{FIM}^b(\mathcal{D}, \lambda)$. Therefore, $\mathcal{FIM}^b(\mathcal{D}, \lambda) = \sigma(\mathcal{FIM}^a(\mathcal{D}, \lambda))$ holds. □

**Proposition 5** *Let $D$ be a transaction database, $[a]$ an orbit following $(\mathcal{S}(\mathcal{D}), \circ)$ of size $n$ and $\lambda$ a minimal support threshold. Then, there exist $n$ symmetries $\sigma_1, \ldots, \sigma_n$ such that $\mathcal{FIM}(\mathcal{D}, \lambda) = \mathcal{FIM}(\mathcal{D}^a, \lambda) \cup \sigma_1(\mathcal{FIM}(\mathcal{D}^a, \lambda)) \cup \ldots \cup \sigma_n(\mathcal{FIM}(\mathcal{D}^a, \lambda))$.*

**Proof:** By induction on $n$ (it is a consequence of Proposition 4). □

Let $\mathcal{D}$ be a transaction database, $\mathcal{O} = \{[a_1], \ldots, [a_k]\}$ the set of all the orbits following $(\mathcal{S}(\mathcal{D}), \circ)$ and $\lambda$ a minimal support threshold. Then, the transaction database $(\ldots(\mathcal{D}^{a_1})\ldots)^{a_k}$ and $(\mathcal{S}(\mathcal{D}), \circ)$ are sufficient to compute $\mathcal{FIM}(\mathcal{D}, \lambda)$. This is a direct consequence of Proposition 5.

Let us take again the transaction database given in Example 1 and the associated symmetry $\sigma$ given in section 2. Using the orbit $[Tomato]^\sigma$, we eliminate $Parmesan$ in transaction 002 as it does not contain the item $Tomato$. While using the orbit $[Mozarella]^\sigma$, we eliminate the item $Beef$ from transaction 002 as it does not contain $Mozarella$. However, using the orbit $[Chili\ pepper]^\sigma$ the item $Anchovies$ can not be eliminated from the transaction 003 as both items, $Chili\ pepper$ and $Anchovies$, occur together in the same transaction.

## 6.2 Itempair-Based Symmetry Breaking

In this section, we propose another method to reduce the search space of possible interesting itemsets by detecting and breaking symmetries between itempairs.

We only consider the particular case of the symmetries with binary cycles in our description. Moreover, we fix an order $\preceq$ on $\mathcal{I}$ and we denote by $min(I)$, with $I$ an itemset, the least item of $I$ following $\preceq$.

**Definition 12 (Ordered Symmetry)** *Let $\mathcal{D}$ be a transaction database and $\sigma = (a_1, a'_1)\cdots(a_k, a'_k)$ be a symmetry of $\mathcal{D}$. $\sigma$ is an ordered symmetry iff for all $i, j \in \{1, \ldots, k\}$ such that $i \leq j$, $a_i \preceq a'_i$ and $a_i \preceq a_j$.*

In the rest of this paper, we consider all symmetries as ordered. Let $\mathcal{D}$ be a transaction database and $\sigma = (a_1, a'_1)\cdots(a_k, a'_k)$ be a symmetry of $\mathcal{D}$, the set of items $\mathcal{L}(\sigma)$ ($\mathcal{L}$ for Left) is defined by $\mathcal{L}(\sigma) = \{a_1, \ldots, a_k\}$, $\mathcal{R}(\sigma)$ ($\mathcal{R}$ for Right) by $\mathcal{R}(\sigma) = \{a'_1, \ldots, a'_k\}$ and $\mathcal{H}(\sigma)$ by $\mathcal{H}(\sigma) = \{\{a_i, a'_j\} | a_i \in \mathcal{L}(\sigma), a'_j \in \mathcal{R}(\sigma)$ and $i \leq j\}$. Moreover, let $E$ be a set of itempairs, we denote by $E(\sigma)$ the following set of itempairs: $\sigma(E) = \{\{x, y\} \in E | x, y \notin \mathcal{R}(\sigma)$ or $\{x, y\} \in \mathcal{H}(\sigma)\}$. Intuitively, $\sigma(E)$ is obtained from $E$ by removing itempairs that can be recovered using $\sigma$.

In the following two propositions, we particularly show how some itempairs can be recovered from the others using symmetries. This will allow us to reduce transaction databases by removing items.

**Proposition 6** *Let $\mathcal{D}$ be a transaction database, $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ a set of symmetries of $\mathcal{D}$, $E_\Sigma = \{\{x, y\} | x, y \in \mathcal{I}_{items}(\Sigma)\}$ and $W_\Sigma = \sigma_1(\sigma_2(\cdots(\sigma_n(E_\Sigma))\cdots))$. Then, for all $\{x, y\} \in \mathcal{I}_{items}(\Sigma)$, there exist $\{x', y'\} \in W_\Sigma$ and a symmetry $\sigma$ such that $\sigma(\{x', y'\}) = \{x, y\}$.*

**Proof:** One can easily prove that $W_\Sigma = \sigma_1(E_\Sigma) \cap \cdots \cap \sigma_n(E_\Sigma)$ (by induction on the value of $n$). Let $x', y' \in \mathcal{I}_{items}(\Sigma)$ such that **(1)** there exists a symmetry $\sigma$ where $\sigma(\{x', y'\}) = \{x, y\}$, **(2)** and for all symmetry $\sigma'$, there are no $x'', y'' \in \mathcal{I}_{items}(\Sigma)$ such that $x'' \prec x'$ (that means in particular $x' \prec y'$) and $\sigma'(\{x'', y''\}) = \{x, y\}$. Intuitively, $\{x', y'\}$ is the smallest itempair allowing to recover $\{x, y\}$. If $\{x', y'\} \notin W_\Sigma$, then there exists $\sigma_i \in \Sigma$ such that $x', y' \in \mathcal{R}(\sigma_i)$ or $\{x', y'\} \notin \mathcal{H}(\sigma_i)$. In both cases, there exists $\{x'', y''\} \in \sigma_i(E_\Sigma)$ such that $x'' \prec x'$ and $\sigma_i(\{x'', y''\}) = \{x', y'\}$. Thus, $\sigma(\sigma_i(\{x'', y''\})) = \sigma(\{x', y'\}) = \{x, y\}$ holds and we get a contradiction. □

**Proposition 7** *Let $\mathcal{D}$ be a transaction database, $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ be a set of symmetries of $\mathcal{D}$, $E = \{\{x, y\} | x, y \in \mathcal{I}_{items}(\mathcal{D})\}$ and $W = \sigma_1(\sigma_2(\cdots(\sigma_n(E_\Sigma))\cdots))$. The three following properties are satisfied:*

1. *there exists $x \in \mathcal{I}_{items}(W)$ such that for all $y \in \mathcal{I}_{items}(\mathcal{D}) \backslash \mathcal{I}_{items}(\Sigma)$, $\{x, y\} \in W$;*
2. *for all $x \in \mathcal{I}_{items}(\mathcal{D})$, there exists $y \in \mathcal{I}_{items}(W)$ and a symmetry $\sigma$ such that $x = \sigma(y)$;*
3. *for all $\{x, y\} \in E \backslash W$, there exists $\{x', y'\} \in W$ and a symmetry $\sigma$ such that $\sigma(\{x', y'\}) = \{x, y\}$.*

**Proof:** The *first property* is proved by taking $x = min(\mathcal{I}_{items}(\Sigma))$. Indeed, by construction of $W$ we have $min(\mathcal{I}_{items}(\Sigma))$ is in $\mathcal{I}_{items}(W)$ and for all $y \in \mathcal{I}_{items}(\mathcal{D}) \backslash \mathcal{I}_{items}(\Sigma)$, $\{x, y\} \in W$.
The *second property* is proved by using the fact that the minimum of each set of the items symmetrical two by two is in $W$.
Let us now prove the *third property*. If $x \notin \mathcal{I}_{items}(\Sigma)$ (resp. $y \notin \mathcal{I}_{items}(\Sigma)$) then by using the second property we know that there exist $y' \in W$ (resp. $x' \in W$) and a symmetry $\sigma$ such that $\sigma(\{x, y'\}) = \{x, y\}$ (resp. $\sigma(\{x', y\}) = \{x, y\}$). Otherwise, $x$ and $y$ are in $\mathcal{I}_{items}(\Sigma)$ and, by using Proposition 6, the property is satisfied. □

**Definition 13** *Let $\mathcal{D}$ be a transaction database, $\Sigma = \{\sigma_1, \ldots, \sigma_n\}$ a set of symmetries of $\mathcal{D}$, $E = \{\{x, y\} | x, y \in \mathcal{I}_{items}(\mathcal{D})\}$ and $W = \sigma_1(\sigma_2(\cdots(\sigma_n(E_\Sigma))\cdots))$. $\mathcal{D}[W]$ is the transaction database obtained from $\mathcal{D}$ as follows: for all $a \in \mathcal{I}_{items}(\mathcal{D})$ and for all transaction $T = (tid, I) \in \mathcal{D}$ with $|I| \geq 2$ and $a \in I$, if for all $b \in I$ we have $\{a, b\} \notin W$, then $a$ is removed from $I$ in the transaction $T$.*

Let $\lambda$ be a minimal support threshold. From Proposition 3 and Proposition 7 (Property 3), we deduce that the transaction database $\mathcal{D}[W]$ and $\Sigma$ are sufficient to compute $\mathcal{FIM}(\mathcal{D}, \lambda)$.

Let us show an example where Itempair-Based Symmetry Breaking approach eliminates more items in the transaction database than the Orbit-based Symmetry Breaking one. Let $\mathcal{D}$ be a transaction database and $\sigma_1 = (a, b)$, $\sigma_2 = (b, c)$ and $\sigma_3 = (c, d)$ three symmetries of $\mathcal{D}$. It is easy to see that $a$, $b$, $c$ and $d$ are in the same orbit. Thus, using orbit-based symmetry breaking, $\mathcal{FIM}(\mathcal{D}, \lambda)$ can be computed from the transaction database obtained from $\mathcal{D}$ by removing $b$, $c$ and $d$ from all transactions that do not contain $a$. However, using the itempair-based symmetry breaking, we can remove more items from $\mathcal{D}$. Indeed, in $\mathcal{D}[\sigma_1(\sigma_2(\sigma_3(E)))]$, with $E = \{\{x, y\}|x, y \in \mathcal{I}_{items}(\mathcal{D})\}$, $c$ and $d$ are removed from all transactions, and $b$ is removed from all transactions that do not contain $a$.

# 7 Experimental evaluation

Our experimental studies are carried out on both real, public and simulated datasets widely used in the data-mining community. We provide several experiments in order to show the improvements that can be achieved using our symmetry breaking-based itemset mining approach. In particular, we provide experiments on challenging high dimensional data with thousands of items.

## 7.1 Experimentation setup

In our study, we carried out a series of three experimentations using different datasets:

1. Simulated datasets: In this experimentation, we use the well-known IBM itemset data generator to generate data with different features (such as dataset size, density, etc.). Note that this generator is open source and publically available[2]. We generated different datasets with different parameters regarding the number of transactions, average number of items per transaction, number of different items, etc. In Table 2, we provide the features of each simulated dataset.
2. Public datasets: The datasets used in this evaluation are from the well-known itemset FIMI repository [3]. However, the datasets from FIMI are mostly simulated and not high dimensional. Therefore, we used $SIDO1$, a dataset containing pharmacology real data form the Causality challenge[4]. This dataset contains massive data of 12678 transactions with 4932 items.
3. Real datasets: In this experimentation, we used real data regarding intrusion detection alerts. The raw data is collected from real and recent alert log files produced by Snort intrusion detection system (IDS) [5] monitoring a university campus network. These alert logs represent three months activity. The alert log files are preprocessed into alert windows. Each alert window represents 1 hour of alerts in dataset $Alert1h$ and 8 hours in dataset $Alert8h$. In the obtained datasets, each transaction represents the alerts triggered during an alert window.

In our experiments, we exploit Saucy[6], a new implementation of the Nauty system. It is originally proposed in [1] and significantly improved in [5]. 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.

Because of lack of space, we only provide experiments using itempair-based symmetry breaking approach.

In the sequel, we note, $|\mathcal{D}|$ (resp. $|\mathcal{I}|$) the number of transactions (resp. different items) in the dataset $\mathcal{D}$. $|Iocc|$ denotes the total number of item occurrences in $\mathcal{D}$, while $\%\ dens$ represents the density of the input transaction dataset. $time(s)$ denotes the cumulated time in seconds required to search and break symmetries. $|Irem|$ represents the number of removed item occurrences from the initial dataset. #sym denotes the number of symmetries discovered in the dataset. Finaly, $Iset$ and $Isset$ corresponds to the number of frequent itemsets obtained on the transaction dataset before and after breaking symmetries respectively.

## 7.2 Results on simulated datasets

Table 2 provides details on the generated datasets and the results of our symmetry breaking approach.

| Dataset | $\|\mathcal{D}\|$ | $\|\mathcal{I}\|$ | $\|Iocc\|$ | $\%dens$ | time(s) | $\|Irem\|$ | #sym |
|---|---|---|---|---|---|---|---|
| Dataset1 | 100 | 1006 | 1020 | 0.099% | 1.29 | 894 | 899 |
| Dataset2 | 778 | 3506 | 4089 | 0.011% | 1.67 | 2774 | 2642 |
| Dataset3 | 6110 | 13680 | 30886 | 0.014% | 2.64 | 10577 | 8371 |
| Dataset4 | 7822 | 17266 | 40506 | 0.023% | 2.89 | 7520 | 9415 |
| Dataset5 | 10000 | 26445 | 200102 | 0.075% | 3.87 | 8330 | 9283 |
| Dataset6 | 60808 | 27141 | 246817 | 0.073% | 7.72 | 13396 | 9074 |
| Dataset7 | 100 | 27151 | 199869 | 7.37% | 8.5 | 13032 | 11607 |

**Table 2.** Details and results on simulated datasets

Table 2 clearly shows the significant improvements (in terms of the number of removed item occurrences) that can be performed as the number of symmetries grows. For instance, in experimentation on $Dataset3$ with 6110 transactions, the number of symmetries is 8371. The second point to mention is that on the generated datasets, the number of symmetries is important in datasets with low densities.

In Table 3, we provide results pointing out the correlation between the size of the preprocessed datasets (in terms of item occurrences ) and the size of outputs (in terms of frequent itemsets) with the number of discovered symmetries. Note that this experiment is carried out on datasets we simulated specifically to involve interesting symmetries. To compare the output size, we use the $LCMv3$ itemset mining implementation[7] [16] on each dataset before and after preprocessing.

| $\#sym$ | $\|\mathcal{I}\|$ | $\|Irem\|$ | $\lambda$ | $\|Iset\|$ | $\|Isset\|$ |
|---|---|---|---|---|---|
| 6 | 900 | 421 | 5 | 1015830 | 4370 |
| 10 | 5586 | 3711 | 90 | 17467891 | 12952 |
| 16 | 37350 | 32482 | 375 | 2679034 | 467 |
| 22 | 148471 | 128471 | 900 | 1462982 | 1232 |
| 26 | 311850 | 289847 | 1450 | 22156 | 33 |

**Table 3.** Comparison of the number of removed item occurences and output size in the number of discovered symmetries.

One can see for instance in Table 3 that as the number of symmetries in the dataset increases, the size of the preprocessed dataset is significantly reduced. In particular, in our simulated dataset, when the number of symmetries equals 26, the item occurrences is reduced from 311850 to 66272 (ensuring a reduction rate of 78%). Note also that the output size (see the number of frequent itemsets $|Iset|$ before preprocessing and the number frequent itemsets $|Isset|$ on the preprocessed dataset) is significantly reduced on all the datasets of Table 3. The output size is computed for each dataset using the same support $\lambda$.

[2] IBM Quest Market-Basket Synthetic Data Generator: http://sourceforge.net/projects/ibmquestdatagen/

[3] Frequent Itemset Mining Implementations Repository: http://fimi.ua.ac.be/

[4] http://www.causality.inf.ethz.ch/challenge.php

[5] http://snort.org

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

[7] http://research.nii.ac.jp/~uno/codes.htm

## 7.3 Results on public datasets

In Table 4, we provide details on the used datasets and the obtained results. The results of Table 4 show that in the tested public datasets,

| Dataset | $\|\mathcal{D}\|$ | $\|\mathcal{I}\|$ | $\|\mathcal{I}occ\|$ | $\%dens$ | $\|Irem\|$ | #sym |
|---|---|---|---|---|---|---|
| Mushroom | 8124 | 186852 | 198169866 | 13% | 1208 | 11 |
| Retail | 88162 | 908576 | 4106009 | 0.005% | 89 | 217 |
| BMS-WebView2 | 77512 | 3341 | 358278 | 0.14% | 4 | 10 |
| BMS-POS | 515597 | 1658 | 3367020 | 0.4% | 3 | 16 |
| SIDO1 | 9297 | 4926 | 4514145 | 9.85% | 568 | 190 |

**Table 4.** Details and results on public datasets

symmetries can be found. For instance, the $SIDO1$ dataset contains 190 symmetries allowing to remove 568 item occurrences from the initial dataset. However, the number of symmetries is not as important as in some simulated datasets of Table 2. Also, the number of removed item occurrences is not significant, consequently the size of the outputs in terms of the number of frequent itemsets is not significantly reduced.

## 7.4 Results on real datasets

In Table 5, we find details on the used alert datasets and the results of our approach. In Table 5, we see that our datasets regarding real intrusion detection alerts contain some symmetries. More importantly, on the dataset $Alert8h$, the size of the output in terms of the number of frequent itemsets (with a support $\lambda$=60) is decreased by a ratio of 50%. The two datasets have different densities and the number of removed items does not depend only on the number of discovered symmetries but also on the nature of the data itself.

| Dataset | $\|\mathcal{D}\|$ | $\|\mathcal{I}\|$ | $\|\mathcal{I}occ\|$ | $\%dens$ | $\|Irem\|$ | #sym |
|---|---|---|---|---|---|---|
| Alert1h | 22203 | 167 | 27049 | 0.73% | 86 | 23 |
| Alert8h | 123 | 167 | 3453 | 16.81% | 248 | 19 |

**Table 5.** Details and results on the alert dataset

Let us summarize the experimental evaluation of our symmetry framework for frequent itemset mining problems. From the obtained results, we can make several interesting observations:

- Several datasets contain symmetries and are discovered by our symmetry detection method in a reasonable amount of time.
- The size of the output (number of frequent itemsets) can be significantly reduced by our symmetry-based framework.
- The reduction rate in terms of the number of eliminated item occurrences in the transaction database does not depend only on the number of found symmetries but also on their form. Indeed, some of the discovered symmetries are made of permutations of items in the same transactions, leading to no reduction using our symmetry breaking approaches. Such kind of symmetries can be better exploited in a dynamic way by Apriori-like algorithms as explained in Section 5 (symmetry pruning). This issue will be a subject of future investigations. The goal is to combine both symmetry breaking and symmetry pruning in the same framework.

# 8 Conclusion and Future Works

Symmetry is an important structural property widely exploited to reduce the search space of many combinatorial problems. In this paper, we have presented the theoretical foundation for discovering and using symmetries in the context of itemset mining problems. This study is important for several reasons. First, this kind of structures can be present in structured data and can be exploited for reducing both the search space and the size of the output. Secondly, even if such output is reduced, the eliminated combinatorial structures can be recovered by symmetry. Symmetries can also be used to help the user for computing either non-symmetric or symmetric patterns. The symmetries discovered from a given set of data, might be valuable in analyzing data themselves. Our theoretical framework includes symmetry detection, symmetry breaking and symmetry pruning. Our experimental results, demonstrate that this kind of structure can be hidden in some classes of transaction databases. We also show that when symmetries are present, their exploitation leads to interesting reduction of the output size.

The symmetry framework proposed in this paper for the itemset mining problem opens interesting research directions in data mining in general. We plan to extend our symmetry framework to other data mining problems such us sequence, tree or graph mining. In the context of itemset mining, there remains many rooms for future improvements. The integration of symmetry pruning in Apriori-like algorithm is clearly an important issue as many datasets contain symmetries between items appearing in the same transactions.

# REFERENCES


[1] 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).

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

[3] J. Crawford, 'A theorical analysis of reasoning by symmetry in first order logic', in Workshop on Tractable Reasonning (AAAI'92), pp. 17–22, (1992).

[4] J. Crawford, M. L. Ginsberg, E. Luck, and A. Roy, 'Symmetry-breaking predicates for search problems', in Proceedings of the Fifth International Conference on Principles of Knowledge Representation and Reasoning (KR'96), 148–159, Morgan Kaufmann, (1996).

[5] Paul T. Darga, Karem A. Sakallah, and Igor L. Markov, 'Faster symmetry discovery using sparsity of symmetries', in Proceedings of the 45th Design Automation Conference (DAC'08), pp. 149–154, (2008).

[6] Eugene C. Freuder, 'Eliminating interchangeable values in constraint satisfaction problems', in Proceedings of the 9th National Conference on Artificial Intelligence (AAAI'91), pp. 227–233, (1991).

[7] Alain Gély, Raoul Medina, Lhouari Nourine, and Yoan Renaud, 'Uncovering and reducing hidden combinatorics in guigues-duquenne bases', in Proceedings of the third International ConferenceFormal Concept Analysis (ICFCA'05), pp. 235–248, (2005).

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

[9] Balakrishnan Krishnamurthy, 'Shorts proofs for tricky formulas', Acta Informatica, **22**, 253–275, (1985).

[10] H. Mannila and H. Toivonen, 'Multiple uses of frequent sets and condensed representations', in KDD, pp. 189–194, (1996).

[11] B. D. McKay, 'Practical graph isomorphism', Congressus Numerantium, **30**, 47–87, (1981).

[12] Raoul Medina, Caroline Noyer, and Olivier Raynaud, 'Efficient algorithms for clone items detection', in CLA'05, pp. 70–81, (2005).

[13] Shin-ichi Minato, 'Symmetric item set mining based on zero-suppressed bdds', in DS'06, pp. 321–326, (2006).

[14] Jean-Francois Puget, 'On the satisfiability of symmetrical constrained satisfaction problems', in Proceedings of the 7th International Symposium on Methodologies for Intelligent Systems (ISMIS'93), pp. 350–361, (1993).

[15] 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, pp. 207–216, Baltimore, (1993). ACM Press.

[16] H. Arimura T. Uno, M. Kiyomi, 'Lcm ver.3: Collaboration of array, bitmap and prefix tree for frequent itemset mining', in OSDM Workshop on Frequent Pattern Mining Implementations, (2005).

[17] 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).