# User's Constraints in Itemset Mining

Christian Bessiere, Nadjib Lazaar, Mehdi Maamar

# User's Constraints in Itemset Mining

Christian Bessiere[1], Nadjib Lazaar[1], Mehdi Maamar[2]

[1] LIRMM, University of Montpellier, CNRS, Montpellier, France
{bessiere,lazaar}@lirmm.fr
[2] CRIL-CNRS, University of Artois, Lens, France
maamar@cril.fr

**Abstract.** Discovering significant itemsets is one of the fundamental tasks in data mining. It has recently been shown that constraint programming is a flexible way to tackle data mining tasks. With a constraint programming approach, we can easily express and efficiently answer queries with user's constraints on itemsets. However, in many practical cases queries also involve user's constraints on the dataset itself. For instance, in a dataset of purchases, the user may want to know which itemset is frequent and the day at which it is frequent. This paper presents a general constraint programming model able to handle any kind of query on the dataset for itemset mining.

## 1 Introduction

People have always been interested in analyzing phenomena from data by looking for significant itemsets. This task became easier and accessible for big datasets thanks to computers, and thanks to the development of specialized algorithms for finding frequent/closed/... itemsets. Nevertheless, looking for itemsets with additional user's constraints remains a bottleneck nowadays. According to [11], there are three ways to handle user's constraints in an itemset mining problem. We can use a pre-processing step that restricts the dataset to transactions that satisfy the constraints. Such a technique quickly becomes infeasible when there is a large number of sub-datasets satisfying the user's constraints. We can integrate the filtering of the user's constraints into the specialized data mining process in order to extract only the itemsets satisfying the constraints. Such a technique requires the development of a new algorithm for each new itemset mining problem with user's constraints. We can sometimes use a post-processing step to filter out the itemsets violating the user's constraints. Such a brute-force technique does not apply to all kinds of constraints and is computationally infeasible when the problem without the user's constraints has too many solutions.

In a recent line of work [8,5,6,4,9], constraint programming (CP) has been used as a declarative way to solve data mining problems. Such an approach has not competed yet with state of the art data mining algorithms [12,10] for simple queries. Nevertheless, the advantage of the CP approach is to be able to add extra (user's) constraints in the model so as to generate only *interesting* itemsets at no other implementation cost.

The weakness of the CP approach is that the kind of user's constraints that can be expressed has never been clarified. It is easy to post constraints on the kind of itemsets we are interested in but the user may be interested in mining only in some particular

transactions of the dataset. For instance, the user may be interested in itemsets that are frequent in transactions corresponding to purchases of less than 100€. None of the current CP approaches is able to catch such kind of constraints. Hence, as with specialized approaches, we need to preprocess the dataset with an ad-hoc algorithm to generate a sub-dataset containing only transaction of less than 100€. It becomes more complex if the user is interested in itemsets that are frequent in transactions corresponding to purchases of a particular sequence of days (such as 'the week of Christmas', 'every Saturday', etc.). Preprocessing the dataset can lead to the generation of a huge number of sub-datasets, each corresponding to a potential sequence of days.

Another consequence of the lack of clarification of what the CP approach can or cannot do is that some CP models can be flawed by the closedness property. As shown in [1], non-monotone constraints interfere with closedness.[1] For instance, if we post the global constraint for frequent closed itemsets (FCIs) proposed in [6] in a CP model and if in addition we post the constraint specifying that the user is only interested in itemsets of size $k$, then frequent itemsets of size $k$ having a superset of same frequency will be lost.

In this paper we present a classification of user's constraints with respect to which itemsets are extracted and from where in the dataset they are extracted. We then propose a generic CP model in which we can capture all these types of user's constraints. The interaction between constraints and closedness is discussed.

The paper is organized as follows. Section 2 presents the background in data mining and constraint programming. In Section 3 we propose a taxonomy of the types of user's constraints that can be useful in itemset mining. In Section 4, we present a CP model able to capture all these user's constraints. Section 5 gives some case studies that can be expressed using our CP model. Section 6 reports experiments.

## 2 Background

### 2.1 Itemset mining

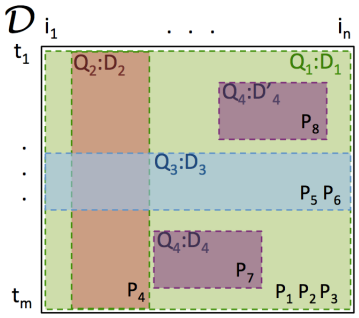
Let $\mathcal{I} = \{1, \ldots, n\}$ be a set of $n$ *item* indices and $\mathcal{T} = \{1, \ldots, m\}$ a set of $m$ *transaction* indices. An itemset $P$ is a subset of $\mathcal{I}$. The set of itemsets is $\mathcal{L}_{\mathcal{I}} = 2^{\mathcal{I}} \backslash \emptyset$. A transactional dataset is a set $\mathcal{D} \subseteq \mathcal{I} \times \mathcal{T}$. A sub-dataset is a subset of $\mathcal{D}$ obtained by removing columns (items) and/or rows (transactions). The set of possible sub-datasets is denoted by $\mathcal{L}_{\mathcal{D}}$. The cover of an itemset $P$ in a sub-dataset $D$, denoted by $cover(D, P)$, is the set of transactions in $D$ containing $P$. The frequency of an itemset $P$ in $D$ is the ratio $\frac{|cover(D,P)|}{|cover(D,\emptyset)|}$. An itemset $P$ is closed in a sub-dataset $D$ if and only if the set of items common to all transactions of $cover(D, P)$ is $P$ itself (that is, $\bigcap_{t \in cover(D,P)} t = P$).

*Example 1.* Let us consider the dataset $\mathcal{D}_1$ involving 8 items and 6 transactions and displayed in Figure 1.a. The cover $cover(\mathcal{D}_1, BEF)$ of the itemset $BEF$ in $\mathcal{D}_1$ is equal to $\{t_1, t_5, t_6\}$. The frequency of $BEF$ in $\mathcal{D}_1$ is thus 50%. The itemset $BEF$ is closed in $\mathcal{D}_1$. The itemset $BE$ is not closed in $\mathcal{D}_1$ because $F$ belongs to all transactions in $cover(\mathcal{D}_1, BE)$.

[1] A constraint $c$ is monotone if any superset of an itemset $P$ satisfying $c$ also satisfies $c$.

| | | Items | | |
|---|---|---|---|---|
| Trans. | | $\mathcal{I}_1$ | $\mathcal{I}_2$ | $\mathcal{I}_3$ |
| $\mathcal{T}_1$ | $t_1$ | B | C E | F G H |
| | $t_2$ | A | D | G |
| $\mathcal{T}_2$ | $t_3$ | A | C D | H |
| | $t_4$ | A | E | F |
| $\mathcal{T}_3$ | $t_5$ | B | E | F |
| | $t_6$ | B | E | F G |

**(a)**




**(b)**

**Fig. 1.** (a) A transaction dataset $\mathcal{D}_1$. (b) Queries on dataset $\mathcal{D}$.

### 2.2 Constraint programming (CP)

A constraint program is defined by a set of variables $X = \{X_1, \ldots, X_n\}$, where $D_i$ is the set of values that can be assigned to $X_i$, and a finite set of constraints $\mathcal{C}$. Each constraint $C(Y) \in \mathcal{C}$ expresses a relation over a subset $Y$ of variables $X$. The task is to find assignments $(X_i = d_i)$ with $d_i \in D_i$ for $i = 1, \ldots, n$, such that all constraints are satisfied.

### 2.3 CP models for itemset mining

In [8,3], De Raedt *et al.* have proposed CP4IM, a first CP model for itemset mining. They showed how some constraints (e.g., frequency and closedness) can be formulated using CP. This model uses two sets of Boolean variables: (1) item variables $\{X_1, X_2, ..., X_n\}$, such that $(X_i = 1)$ if and only if the extracted itemset $P$ contains $i$; (2) transaction variables $\{T_1, T_2, ..., T_m\}$, such that $(T_t = 1)$ if and only if the extracted itemset $P$ is in the transaction $t$. The relationship between $P$ and $T$ is modeled by $m$ reified $n$-ary constraints. The minimal frequency constraint and the closedness constraint are also encoded by $n$-ary and $m$-ary reified constraints.

Recently, global constraints have been proposed to model and solve efficiently data mining problems. The CLOSEDPATTERN global constraint in [6] compactly encodes both the minimal frequency and the closedness constraints. This global constraint does not use reified constraints. It is defined only on item variables. The filtering algorithm ensures domain consistency in a polynomial time and space complexity. The COVERSIZE global constraint in [9] uses a reversible sparse bitset data structure to compute the subset of transactions that cover an itemset. The filtering algorithm computes a lower and an upper-bound on the frequency.

## 3 User's Constraints Taxonomy

For an itemset mining task we aim at extracting all itemsets $P$ of $\mathcal{L}_\mathcal{I}$ satisfying a query $Q(P)$ that is a conjunction of (user's) constraints. The set $Th(Q) = \{P \in \mathcal{L}_\mathcal{I} \mid Q(P)\}$

is called *a theory* [7]. Common examples of user's constraints on extracted itemsets are frequency, closedness, maximality, etc. However, it may be desirable for a user to ask for itemsets extracted from particular parts of the dataset. In the general case, a query predicate, denoted by $Q(D, P)$, is expressed both on the itemsets $P$ it returns and on the sub-datasets $D \in \mathcal{L}_{\mathcal{D}}$ on which it mines. The extracted elements forming a theory are now pairs:

$$Th(Q) = \{(D, P) \mid D \in \mathcal{L}_{\mathcal{D}} \wedge P \in \mathcal{L}_{\mathcal{I}} \wedge Q(D, P)\}.$$

To make the description of our user's constraints taxonomy less abstract, we suppose a categorization of items and transactions. Items are products belonging to $k$ categories (e.g., food, electronics, cleaning, etc), denoted by $\mathcal{I} = \{\mathcal{I}_1, \ldots, \mathcal{I}_k\}$. Transactions are categorized into $v$ categories of customers (e.g., categories based on age/gender criteria), denoted by $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_v\}$. It is important to bear in mind that these categories are just examples provided for illustration purposes. Example 2 presents the running example (with categories) that will be used to illustrate each of the types of user's constraints we present in this section.

*Example 2.* Let us consider again the dataset $\mathcal{D}_1$ displayed in Figure 1.a. For our running example, items belong to three categories $\{A, B\}$, $\{C, D, E\}$ and $\{F, G, H\}$, and transactions belong to three categories $\{t_1, t_2\}$, $\{t_3, t_4\}$ and $\{t_5, t_6\}$.

### 3.1 User's constraints on itemsets

When the user comes with constraints only on the nature of the itemsets to extract, the query, $Q_1$, is equivalent to a standard itemset mining task. We mine on the whole dataset. Figure 1.b graphically illustrates this. The itemsets that are solution for $Q_1$ (i.e., $P_1$, $P_2$ and $P_3$) are extracted from $D_1 = \mathcal{D}$.

An example of such a query where user's constraints are expressed only on itemsets is the query $Q_1$ asking for FCIs:

$$Q_1(D, P) \equiv frequent(D, P, \theta) \wedge closed(D, P)$$

where $frequent(D, P, \theta)$ and $closed(D, P)$ are predicates expressing user's constraints on the frequency (with a minimum frequency $\theta$) and the closedness of an itemset $P$ in $D$, where $D$ is $\mathcal{D}$ in this case. The query $Q_1$ on the dataset $\mathcal{D}_1$ of Figure 1.a with a minimum frequency $\theta \geq 50\%$ returns $A$, $BEF$, $EF$ and $G$ as FCIs.

As a second example of such a query on itemsets, the user can ask a query $Q'_1$ where the extracted itemsets are FCIs and the items are taken from at least $lb$ and at most $ub$ categories:

$$Q'_1(D, P) \equiv Q_1(D, P) \wedge atLeast(P, lb) \wedge atMost(P, ub)$$

where $atLeast(P, lb)$ and $atMost(P, ub)$ are user's constraints ensuring that the itemset $P$ overlaps between $lb$ and $ub$ categories of items. The query $Q'_1$ on the dataset $\mathcal{D}_1$ of Figure 1.a with $lb = ub = 2$ and minimum frequency $\theta = 50\%$ only returns $EF$. It does not return $A$ and $G$ because each of these itemsets belongs to a single category. It does not return $BEF$ because it belongs to three categories.

### 3.2 User's constraints on items

In addition to constraints on itemsets, the user may want to put constraints on the items themselves. Such constraints are constraints on the dataset. They specify on which items/columns the mining will occur. In Figure 1.b, constraints on items lead the query, $Q_2$, to mine on the sub-dataset $D_2$ satisfying constraints on items, from which we extract the itemset $P_4$ satisfying the constraints on itemsets.

As an example, the user can ask a query $Q_2$, where the extracted itemsets are FCIs of sub-datasets containing at least $lb_I$ categories of items and at most $ub_I$ categories:

$$Q_2(D, P) \equiv Q_1(D, P) \wedge atLeastI(D, lb_I) \wedge atMostI(D, ub_I)$$

where $atLeastI(D, lb_I)$ and $atMostI(D, ub_I)$ are user's constraints ensuring that the dataset $D$ contains between $lb_I$ and $ub_I$ categories of items. As opposed to $Q_1$ and $Q'_1$, $Q_2$ seeks itemsets in sub-datasets satisfying a property on their items. The query $Q_2$ on the dataset $\mathcal{D}_1$ of Figure 1.a with $lb_I = ub_I = 2$ and minimum frequency $\theta = 50\%$ returns:

- $A$, $BE$ and $E$ on $\mathcal{I}_1 + \mathcal{I}_2$,
- $A$, $BF$, $F$ and $G$ on $\mathcal{I}_1 + \mathcal{I}_3$,
- $EF$ and $G$ on $\mathcal{I}_2 + \mathcal{I}_3$.

### 3.3 User's constraints on transactions

The user may also want to put constraints on transactions. Such constraints determine on which transactions/rows the mining will occur. In Figure 1.b, constraints on transactions lead the query, $Q_3$, to mine on the subset $D_3$ of transactions from which we extract the itemsets $P_5$ and $P_6$.

As an example, the user can ask a query $Q_3$, where the extracted itemsets are FCIs on at least $lb_T$ and at most $ub_T$ categories:

$$Q_3(D, P) \equiv Q_1(D, P) \wedge atLeastT(D, lb_T) \wedge atMostT(D, ub_T)$$

where $atLeastT(D, lb_T)$ and $atMostT(D, ub_T)$ are user's constraints ensuring that the dataset $D$ contains between $lb_T$ and $ub_T$ categories of transactions. The query $Q_3$ on the dataset $\mathcal{D}_1$ of Figure 1.a with $lb_T = ub_T = 2$ and minimum frequency $\theta = 50\%$ returns:

- $A$, $AD$, $CH$, $EF$ and $G$ on $\mathcal{T}_1 + \mathcal{T}_2$,
- $BEF$, $BEFG$ and $G$ on $\mathcal{T}_1 + \mathcal{T}_3$,
- $A$, $BEF$ and $EF$ on $\mathcal{T}_2 + \mathcal{T}_3$.

### 3.4 User's constraints on items and transactions

Finally, the user may want to put constraints on both items and transactions. In Figure 1.b, such constraints lead the query, $Q_4$, to mine on $D_4$ and $D'_4$ from which we extract the itemsets $P_7$ and $P_8$.

The user can ask a query $Q_4$, where the extracted itemsets are FCIs of sub-datasets containing at least $lb_I$ and at most $ub_I$ categories of items and at least $lb_T$ and at most $ub_T$ categories of transactions:

$$Q_4(D,P) \equiv Q_2(D,P) \wedge Q_3(D,P)$$

The query $Q_4$ on the dataset $\mathcal{D}_1$ of Figure 1.a with $lb_I = ub_I = lb_T = ub_T = 2$ and minimum frequency $\theta = 50\%$ will have to explore nine possible sub-datasets in which to look for frequent closed itemsets:

| | $\mathcal{I}_1 + \mathcal{I}_2$ | $\mathcal{I}_1 + \mathcal{I}_3$ | $\mathcal{I}_2 + \mathcal{I}_3$ |
|---|---|---|---|
| $\mathcal{T}_1 + \mathcal{T}_2$ | $A$, $AD$, $C$, $E$ | $A$, $F$, $G$, $H$ | $CH$, $D$, $EF$, $G$ |
| $\mathcal{T}_1 + \mathcal{T}_3$ | $BE$ | $BF$, $BFG$, $G$ | $EF$, $EFG$, $G$ |
| $\mathcal{T}_2 + \mathcal{T}_3$ | $A$, $BE$, $E$ | $A$, $BF$, $F$ | $EF$ |

$Q_4$ is merely a combination of $Q_1$ (user's constraints on itemsets), $Q_2$ (user's constraints on items), and $Q_3$ (user's constraints on transactions). We presented it to show that our model allows any kind of combinations of users constraints.

### 3.5 A simple illustration: Where Ferrari cars are frequently bought?

Consider a dataset of cars purchases in France, where each transaction/purchase also contains items representing the city, the department, and the region where the purchase was performed. (City/department/region is the way France is administratively organized.) The user may be interested in finding where (city, department or region) more than 10% of the purchases are Ferrari cars. This can be done by the query:

$$RQ(D,P) \equiv frequent(D,P,10\%) \wedge (Ferrari \in P) \wedge \\ (Reg(D) \vee Dep(D) \vee City(D))$$

where $Reg(D)$, $Dep(D)$ and $City(D)$ are user's constraints ensuring that the dataset $D$ corresponds to one of the administrative entities of France.

## 4 A General CP Model for Itemset Mining

We present ITEMSET, a CP model for itemset mining taking into account any type of user's constraints presented in Section 3.

### 4.1 Variables

$P$, $T$, $H$ and $V$ are Boolean vectors to encode:

- $P = \langle P_1, \ldots, P_n \rangle$: the itemset we are looking for. For each item $i$, the Boolean variable $P_i$ represents whether $i$ is in the extracted itemset.
- $T = \langle T_1, \ldots, T_m \rangle$: the transactions that are covered by the extracted itemset.
- $H = \langle H_1, \ldots, H_n \rangle$: The items in the sub-dataset where the mining will occur. $H_i = 0$ means that the item/column $i$ is ignored.
- $V = \langle V_1, \ldots, V_m \rangle$: The transactions in the sub-dataset where the mining will occur. $V_j = 0$ means that the transaction/row $j$ is ignored.

$\langle H, V \rangle$ circumscribes the sub-dataset used to extract the itemset. The CP solver searches in different sub-datasets, backtracking from a sub-dataset and branching on another. $\langle P, T \rangle$ represents the itemset we are looking for, and its coverage in terms of transactions.

### 4.2 Constraints

Our generic CP model consists of three sets of constraints:

$$\textsc{ItemSet}(P, H, T, V) = \begin{cases} \textsc{DataSet}(H, V) \\ \textsc{Channeling}(P, H, T, V) \\ \textsc{Mining}(P, H, T, V) \end{cases}$$

$\textsc{DataSet}(H, V)$ is the set of constraints that express user's constraints on items (i.e., $H$) and/or transactions (i.e., $V$). This set of constraints circumscribes the sub-datasets.

$\textsc{Channeling}(P, H, T, V)$ is the set of channeling constraints that express the relationship between the two sets of variables $\langle P, T \rangle$ and $\langle H, V \rangle$:

$$H_i = 0 \Rightarrow P_i = 0$$
$$V_j = 0 \Rightarrow T_j = 0$$

These constraints guarantee that if an item (resp. a transaction) is not part of the mining process, it will not be part of the extracted itemset (resp. the cover set).

$\textsc{Mining}(P, H, T, V)$ is the set of constraints that express the (user's) constraints on itemsets such as frequency, closedness, size, and more sophisticated user's constraints.

## 5 ItemSet Model: Cases Studies

In this section, we illustrate our CP model ItemSet on the queries detailed in Section 3. For each query, user's constraints can be written in the DataSet and/or Mining parts of the ItemSet model. Channeling remains unchanged.

#### Query $Q_1$

For query $Q_1$, we have user's constraints only on itemsets. That is, the mining process will occur on the whole set of transactions. For such a case, we have:

$$\textsc{DataSet}(H, V) = \begin{cases} \forall i \in \mathcal{I} : H_i = 1 \\ \forall j \in \mathcal{T} : V_j = 1 \end{cases}$$

The user asks for FCIs:

$$\text{MINING}(P,H,T,V) = \begin{cases} \forall j \in \mathcal{T} : T_i = 1 \Leftrightarrow \sum_{i \in \mathcal{I}} P_i(1-\mathcal{D}_{ij}) = 0 \\ \forall i \in \mathcal{I} : P_i = 1 \Rightarrow \frac{1}{|\mathcal{T}|} \sum_{j \in \mathcal{T}} T_j \mathcal{D}_{ij} \geq \theta \\ \forall i \in \mathcal{I} : P_i = 1 \Leftrightarrow \sum_{j \in \mathcal{T}} T_j(1-\mathcal{D}_{ij}) = 0 \end{cases}$$

This corresponds to the model presented in [3] and how it can be written in the MINING part of our ITEMSET model. The first constraint represents the coverage constraint, the second is the minimum frequency with respect to a given minimum frequency $\theta$, and the third one expresses the closedness constraint. Note that to obtain an optimal propagation, this part can be replaced by the global constraint CLOSEDPATTERN [6]:

$$\text{MINING}(P,H,T,V) = \text{CLOSEDPATTERN}_\theta(P,T)$$

### Query $Q'_1$

For $Q'_1$, we have $k$ item categories. The user asks for FCIs extracted from the whole dataset but the items composing the extracted FCI must belong to at least $lb$ categories and at most $ub$ categories where $lb \leq ub \leq k$. The DATASET part is the same as in the case of $Q_1$. The MINING part takes into account the new user's constraint on itemsets:

$$\text{MINING}(P,H,T,V) = \begin{cases} \text{CLOSEDPATTERN}_\theta(P,T) \\ lb \leq \sum_{j=1}^{k} \max_{i \in \mathcal{I}_j} P_i \leq ub \end{cases}$$

The first constraint is used to extract FCIs. The second constraint holds if and only if the items of the extracted itemset belong to $lb$ to $ub$ categories.

### Query $Q_2$

For $Q_2$, the user asks for FCIs not from the whole dataset as in $Q_1$ and $Q'_1$, but from a part of the dataset with $lb_I$ to $ub_I$ categories of items. Such user's constraints on items are expressed in the DATASET part of our model as:

$$\text{DATASET}(H,V) = \begin{cases} lb_I \leq \sum_{j=1}^{k} \min_{i \in \mathcal{I}_j} H_i = \sum_{j=1}^{k} \max_{i \in \mathcal{I}_j} H_i \leq ub_I \\ \forall j \in \mathcal{T} : V_j = 1 \end{cases}$$

For each category, the first constraint activates all items or none. The number of categories with their items activated is between $lb_I$ to $ub_I$. The second constraint activates the whole set of transactions. The MINING part is the almost the same as in the case of

$Q_1$. The only difference is that we need an adapted version of the CLOSEDPATTERN$_\theta$ where frequent closed itemsets are mined in the sub-dataset circumscribed by the $H$ an $V$ vectors:

$$\text{MINING}(P, H, T, V) = \text{CLOSEDPATTERN}_\theta(P, H, T, V)$$

### Query $Q_3$

For $Q_3$, we have $v$ transaction categories. With $Q_3$, the user asks for FCIs not from the whole set of transactions but from at least $lb_T$ and at most $ub_T$ transaction categories. These user's constraints on transactions are written in our model as:

$$\text{DATASET}(H, V) =$$
$$\begin{cases} \forall i \in \mathcal{I} : H_i = 1 \\ lb_T \leq \sum_{j=1}^{v} \min_{i \in \mathcal{T}_j} V_i = \sum_{j=1}^{v} \max_{i \in \mathcal{T}_j} V_i \leq ub_T \end{cases}$$

The first constraint activates the whole set of items. For each category, the second constraint activates all transactions or none. The number of categories with their transactions activated is between $lb_T$ and $ub_T$. The user asks for CFIs. That is, the MINING part is the same as in the case of $Q_2$.

### Query $Q_4$

$Q_4$ involves the different types of user's constraints presented in this paper. We have $k$ item categories and $v$ transaction categories. The user asks for FCIs on at least $lb_I$ and at most $ub_I$ categories of products and at least $lb_T$ and at most $ub_T$ categories of customers.

$$\text{DATASET}(H, V) =$$
$$\begin{cases} lb_I \leq \sum_{j=1}^{k} \min_{i \in \mathcal{I}_j} H_i = \sum_{j=1}^{k} \max_{i \in \mathcal{I}_j} H_i \leq ub_I \\ lb_T \leq \sum_{j=1}^{v} \min_{i \in \mathcal{T}_j} V_i = \sum_{j=1}^{v} \max_{i \in \mathcal{T}_j} V_i \leq ub_T \end{cases}$$

The first constraint ensures the sub-dataset satisfies the constraints on items (categories activated as a whole and between $lb_I$ and $ub_I$ of them activated). The second constraint ensures the sub-dataset satisfies the constraints on transactions (categories activated as a whole and between $lb_T$ and $ub_T$ of them activated). As we look for FCIs, the MINING part remains the same as in the case of $Q_2$ and $Q_3$.

### Query $RQ$

We illustrate our model on the query presented in Section 3.5: *Where Ferrari cars are frequently bought?*. To make it simple, suppose that transactions are categorized

into $r$ regions $\mathcal{T} = \{\mathcal{T}_1, \mathcal{T}_2, \ldots, \mathcal{T}_r\}$, each region is composed of $d$ departments $\mathcal{T}_i = \{\mathcal{T}_{i:1}, \mathcal{T}_{i:2}, \ldots, \mathcal{T}_{i:d}\}$, and each department is composed of $c$ cities $\mathcal{T}_{i:j} = \{\mathcal{T}_{i:j:1}, \mathcal{T}_{i:j:2}, \ldots, \mathcal{T}_{i:j:c}\}$. (In the real case, the number of cities per department and departments per region can vary.)

The CHANNELING part of the model is the same as in our generic CP model presented in Section 4. We need to define the DATASET and the MINING parts for the $RQ$ query. In the following, $f$ refers to the item representing the fact that the brand of the car is Ferrari.

$$\text{DATASET}(H, V) = \begin{cases} \forall i \in \mathcal{I} \setminus \{f\} : H_i = 0 \\ P_f = 1 \\ (1) \vee (2) \vee (3) \end{cases}$$

where (1), (2), and (3) are the constraints specifying that itemsets are extracted from a region, a department, or a city. That is, (1), (2), and (3) are constraints that we can express as in the second line of $\text{DATASET}(H, V)$ of query $Q_3$ with $lb_T = ub_T = 1$.[2]

$$\text{MINING}(P, H, T, V) = frequent(P, H, T, V, 10\%)$$

where frequent itemsets are mined in the sub-dataset circumscribed by the $H$ an $V$ vectors. The $V$ vector characterizes the places where Ferrari cars are frequently bought. We thus observe that the interesting part of the solutions of this data mining task is more in the value of the $V$ variables than in the $P$ variables.

**An observation on closedness**

As pointed out in the introduction and in [1], closedness can interfere with user's constraints when they are not monotone. Existing CP approaches can lead to the loss of solutions because CP approaches extract closed itemsets that in addition satisfy the user's constraints. (Some itemsets may satisfy the user's constraints whereas not being closed, and closed itemsets may violate a user's constraint.) What we usually want is to extract itemsets that are closed with respect to the user's constraints. Our model allows us to specify on which sub-datasets frequency and closedness will be computed. As a consequence, when these sub-datasets satisfy some non-intersecting properties, we are able to safely combine closedness with non-monotone constraints.

Take for instance Example 1 with a minimum frequency of $50\%$. If we want itemsets not containing $A$, $F$, or $G$, and closed with respect to these constraints, no system is able to return the only solution $BE$ because it is not closed ($BEF$ has the same frequency). In our model, if we set $H_A = H_F = H_G = 0$, $BE$ is returned as a closed itemset of the sub-dataset $\mathcal{D}_1[BCDE]$. Similarly, if we want itemsets of size 2 and closed with respect to this constraint, all systems will return $EF$ and will miss $BE$ and $BF$,

[2] A CP expert may object that disjunctions of predicates are not the most efficient way to express constraints. This operational concern can be addressed by capturing $(1) \vee (2) \vee (3)$ into a single global constraint, or by simply adding redundant constraints $V_p = 1 \rightarrow V_r = 1$ for every pair $(p, r)$ of transactions in the same city, and $(V_p = 1 \wedge V_q = 1) \rightarrow V_r = 1$ for every triplet $(p, q, r)$ of transactions in the same region (resp. department) such that $p$ and $q$ are not in the same department (resp. city).

**Table 1.** Properties of the used datasets

| Dataset | $\|\mathcal{T}\|$ | $\|\mathcal{I}\|$ | $\overline{\|\mathcal{T}\|}$ | $\rho$ | domain |
|---|---|---|---|---|---|
| `Zoo` | 101 | 36 | 16 | 44% | zoo database |
| `Primary` | 336 | 31 | 15 | 48% | tumor descriptions |
| `Vote` | 435 | 48 | 16 | 33% | U.S voting Records |
| `Chess` | 3196 | 75 | 37 | 49% | game steps |
| `Mushroom` | 8124 | 119 | 23 | 19% | specie's mushrooms |

`Primary`: Primary-tumor

which are not closed because $BEF$ has the same frequency. In our model, if we set $\sum_1^n H_i = 2$ (in addition to $\sum_1^n P_i = 2$), $BE$ and $BF$ are returned as closed itemsets of the sub-datasets $\mathcal{D}_1[BE]$ and $\mathcal{D}_1[BF]$ respectively. Observe that none of the constraints above are monotone. Unfortunately, not all user's constraints can be combined with closedness in our model. If we want itemsets of size *at most* 2 and closed with respect to this constraint, and if we set $\sum_1^n H_i \leq 2$, $A$, $BE$, $BF$, $EF$, and $G$ are returned, but also $B$, $E$, and $F$ because they are closed for the sub-datasets $\mathcal{D}_1[B]$, $\mathcal{D}_1[E]$ and $\mathcal{D}_1[F]$ respectively, whereas they are not closed for the constraint "itemset of size at most 2".

## 6 Experimental Evaluation

We made experiments to evaluate the queries $Q_1$, $Q_2$, $Q_3$ and $Q_4$ on our generic CP model ITEMSET for itemset mining.

### 6.1 Benchmark datasets

We selected several real-sized datasets from the FIMI repository[3] and the CP4IM repository[4]. These datasets have various characteristics representing different application domains. For each dataset, Table 1 reports the number of transactions $|\mathcal{T}|$, the number of items $|\mathcal{I}|$, the average size of transactions $\overline{|\mathcal{T}|}$, its density $\rho$ (i.e., $\overline{|\mathcal{T}|}/|\mathcal{I}|$), and its application domain. The datasets are presented by increasing size.

### 6.2 Experimental protocol

We implemented the ITEMSET model presented in Section 4. This implementation, named CP-ITEMSET, is in C++, on top of the `Gecode` solver (www.gecode.org/). The frequency and closedness constraints are performed by a new implementation of the CLOSEDPATTERN global constraint taking into account the variables $H$, $V$.[5] For LCM, the state-of-the-art specialized algorithm for CFIs, we used the publicly available version (http://research.nii.ac.jp/ uno/codes.htm). All experiments were conducted on an Intel Xeon E5-2665 @2.40 Ghz and a 48GB RAM with a timeout of 900 seconds.

[3] http://fimi.ua.ac.be/data/
[4] https://dtai.cs.kuleuven.be/CP4IM/datasets/
[5] http://www.lirmm.fr/∼lazaar/cpminer.html

**Table 2.** LCM and CP-ITEMSET on $Q_1$ queries. (Times are in seconds.)

| Instances | #FCIs | LCM | CP-ITEMSET |
|---|---|---|---|
| Zoo_50_5 | 4 | 0.01 | 0.01 |
| Primary_60_6 | 1 | 0.01 | 0.01 |
| Vote_50_2 | 1 | 0.01 | 0.01 |
| Mushroom_50_5 | 8 | 0.02 | 0.10 |
| Chess_80_10 | 4 | 0.03 | 0.29 |

In all our experiments we selected a minimum support $\theta$ and a minimum size of itemsets $k$ in order to have constrained instances with less than 10 solutions. The reason of this protocol is that a human cannot process millions of solutions. The purpose of user's constraints is to allow the user to focus on interesting solutions only. But, whatever the desired number of solutions, LCM always needs to go through a LCM +(preprocessing and/or postprocessing) that generates millions of patterns and then filters out the 'non-interesting' ones. Even if LCM is very fast to enumerate a huge number of itemsets, it cannot avoid the combinatorial explosion of all possible sub-datasets.

An instance is defined by the pair frequency/minsize $(\theta, k)$. For example, Zoo_50_5 denotes the instance of the Zoo dataset with a minimum support of $50\%$ and solutions of at least 5 items. Note that the constraint on the size of the itemset is simply added to the MINING part of our ITEMSET model as follows: $minSize_k(P) \equiv \sum_{i \in \mathcal{I}} P_i \geq k$. Note also that such a constraint is integrated in LCM without the need to a post-processing to filter out the undesirable itemsets.

### 6.3 Query $Q_1$

Our first experiment compares LCM and CP-ITEMSET on queries of type $Q_1$, where we have only user's constraints on itemsets. We take the $Q_1$ of the example in Section 3.1, where the user asks for FCIs. We added the $minSize$ constraint in the MINING part of the ITEMSET model. Table 2 reports the CPU time, in seconds, for each approach on each instance. We also report the total number of FCIs (#FCIs $\leq 10$) for each instance.

The main observation that we can draw from Table 2 is that, as expected, the specialized algorithm LCM wins on all the instances. However, CP-ITEMSET is quite competitive. LCM is only from 1 to 9 times faster.

### 6.4 Query $Q_2$

In addition to user's constraints on itemsets, in $Q_2$ the user is able to express constraints on items. We take the $Q_2$ of the example in Section 3.2, where items are in categories and the user asks for FCIs extracted from at least $lb_I$ and at most $ub_I$ categories. We again added the $minSize$ constraint.

Table 3 reports the results of the comparison between PP-LCM (LCM with a preprocessing) and CP-ITEMSET on a set of instances. For each instance, we report the number of item categories $\#\mathcal{I}_i$, the used $lb_I$ and $ub_I$, the total number $\#D$ of sub-datasets satisfying the constraints on items, the number of solutions #FCIs, and the

**Table 3.** PP-LCM and CP-ITEMSET on $Q_2$ queries. (Times are in seconds.)

| Instance | $\#\mathcal{I}_i$ | $(lb_I,ub_I)$ | $\#D$ | #FCIs | PP-LCM | CP-ITEMSET |
|---|---|---|---|---|---|---|
| Zoo_80_2 | 6 | (2,3) | 35 | 5 | 0.58 | **0.02** |
| | 6 | (3,4) | 35 | 10 | 0.62 | **0.03** |
| Primary_70_5 | 3 | (2,3) | 4 | 2 | 0.17 | **0.02** |
| Vote_50_2 | 6 | (2,3) | 35 | 5 | 0.53 | **0.02** |
| Mushroom_50_4 | 17 | (2,2) | 136 | 9 | **5.32** | 6.14 |
| Mushroom_50_4 | 17 | (2,3) | 816 | 1 | **41.31** | 51.04 |
| Chess_70_10 | 5 | (2,3) | 20 | 1 | **1.24** | 2.12 |
| Chess_80_10 | 5 | (2,5) | 26 | 5 | **1.93** | 3.32 |
| Chess_70_5 | 15 | (2,2) | 105 | 6 | 2.78 | **0.97** |
| Chess_80_6 | 15 | (2,3) | 560 | 2 | 14.57 | **7.15** |

time in seconds. Note that the categories have the same size and for a given $\#\mathcal{I}_i = n'$, and an $(lb_I, ub_I)$, we have $\#D = \sum_{i=lbI}^{ubI} \binom{n'}{i}$.

It is important to bear in mind for such a query, PP-LCM acts in two steps: (i) preprocessing generating all possible sub-datasets with respect to the user's constraints on items; (ii) run LCM on each sub-dataset. The first step can be very expensive in terms of memory consumption because the space complexity of generating all sub-datasets is in $O(n' \times n \times m)$, where $n'$ is the number of item categories, and $n$ and $m$ the number of items and transaction.

In Table 3 we observe that CP-ITEMSET outperforms PP-LCM on 6 instances out of 10.

### 6.5 Query $Q_3$

Let us now present our experiments on queries of type $Q_3$ where we have user's constraints on itemsets and transactions. We take the $Q_3$ of the example in Section 3.3 where transactions are in categories. We added the $minSize$ constraint.

Table 4 reports the results of the comparison between PP-LCM and CP-ITEMSET. For each instance, we report the number of transaction categories $\#\mathcal{T}_i$, the lower and upper bounds $(lb_T, ub_T)$ on transaction categories, the number of sub-datasets $\#D$, the number of extracted solutions #FCIs and the time in seconds. Note that for a number of categories $\#\mathcal{T}_i = m'$ and a given $(lb_T, ub_T)$, we have $\#D = \sum_{i=lbT}^{ubT} \binom{m'}{i}$.

For $Q_3$, PP-LCM acts again in two steps. The space complexity of the preprocessing step is in $O(m' \times n \times m)$, with $m'$ transaction categories, $n$ items and $m$ transactions.

In Table 4 we observe that CP-ITEMSET is faster than PP-LCM on 6 instances out of 10. CP-ITEMSET wins on instances where $\#D$ is large. On Vote_80_3 with $\#\mathcal{T}_i = 29$ and $(lb_T, ub_T) = (2, 5)$, PP-LCM reports a timeout whereas CP-ITEMSET solves it in 12 minutes.

### 6.6 Query $Q_4$

Our last experiment is on queries of type $Q_4$ where the user can put constraints on both items and transactions in addition to the ones on the itemsets themselves. We take the

**Table 4.** PP-LCM and CP-ITEMSET on $Q_3$ queries. (Times are in seconds.)

| **Instance** | $\#\mathcal{T}_i$ | $(lb_T,ub_T)$ | $\#D$ | #FCIs | PP-LCM | CP-ITEMSET |
|---|---|---|---|---|---|---|
| Zoo_70_10 | 10 | (1,10) | 1,023 | 2 | 7.95 | **1.12** |
| Zoo_80_5 | 10 | (2,10) | 1,013 | 8 | 9.05 | **1.37** |
| Primary_85_4 | 7 | (2,7) | 120 | 1 | 1.45 | **0.25** |
| Vote_70_6 | 29 | (2,3) | 4,060 | 3 | 37.93 | **17.95** |
| Vote_80_3 | 29 | (2,4) | 27,811 | 4 | 324.53 | **135.53** |
| Vote_80_3 | 29 | (2,5) | 146,566 | 4 | TO | **739.31** |
| Mushroom_70_12 | 12 | (2,2) | 66 | 3 | **3.13** | 24.45 |
| | 12 | (3,3) | 220 | 2 | **12.63** | 87.65 |
| Chess_90_22 | 34 | (2,2) | 561 | 1 | **8.43** | 15.10 |
| Chess_90_26 | 94 | (2,2) | 4,371 | 3 | **49.73** | 68.82 |

TO**: timeout**

**Table 5.** LCM and CP-ITEMSET on $Q_4$ queries. (Times are in seconds.)

| **Instances** | $\#\mathcal{I}_i$ | $\#\mathcal{T}_i$ | $(lb_I,ub_I)$ | $(lb_T,ub_T)$ | $\#D$ | #FCIs | PP-LCM | CP-ITEMSET |
|---|---|---|---|---|---|---|---|---|
| Zoo_70_6 | 6 | 10 | (2,3) | (2,3) | 5,775 | 8 | 39.69 | **1.75** |
| Zoo_50_11 | 6 | 10 | (3,4) | (3,4) | 11,550 | 9 | 88.66 | **3.36** |
| Zoo_85_5 | 6 | 10 | (2,6) | (2,10) | 57,741 | 8 | 521.89 | **31.86** |
| Primary_82_5 | 3 | 12 | (2,3) | (2,10) | 16,280 | 8 | 199.58 | **36.13** |
| Vote_70_6 | 6 | 29 | (2,3) | (2,3) | 142,100 | 2 | TO | **118.67** |
| Vote_72_5 | 8 | 29 | (2,3) | (2,3) | 341,040 | 2 | TO | **201.79** |
| Mushroom_80_5 | 17 | 12 | (2,2) | (2,2) | 8,976 | 10 | 446.42 | **102.68** |
| Mushroom_82_5 | 17 | 12 | (2,2) | (3,3) | 29,920 | 7 | TO | **455.19** |
| Chess_90_16 | 5 | 34 | (2,3) | (2,2) | 11,220 | 3 | 286.42 | **87.22** |

TO**: timeout**

$Q_4$ of the example in Section 3.4 where items and transactions are in categories. We added the $minSize$ constraint.

Table 5 reports results of the comparison between PP-LCM and CP-ITEMSET acting on different instances. We report the number of uniform categories of items/transactions, the used $(lb_I, ub_I)$ and $(lb_T, ub_T)$, the number of sub-datasets $\#D$, the number of solutions #FCIs and the time in seconds. PP-LCM needs to generate all possible sub-datasets $\#D = \sum_{i=lbT}^{ubT} \binom{m'}{i} \times \sum_{i=lbI}^{ubI} \binom{n'}{i}$, where $n', m', n$ and $m$ are respectively the number of item categories, transaction categories, items and transactions. CP-ITEMSET is able to deal with the different queries $Q_4$ just by changing the parameters $k, lb_T, ub_T$, $lb_I, ub_I$, whereas PP-LCM needs a time/memory consuming preprocessing before each query.

We see in Table 5 that CP-ITEMSET significantly outperforms PP-LCM. On the instances where PP-LCM does not report a timeout, CP-ITEMSET is from 4 to more than 26 times faster than PP-LCM. The pre-processing step of PP-LCM can reach $90\%$ of the total time. As $\#D$ grows exponentially, it quickly leads to an infeasible preprocessing step (see the 3 timeout cases of PP-LCM).

## 7 Related Work

In [8,3], De Raedt *et al.* proposed CP4IM, a CP model to express constraints in itemset mining. CP4IM is able to express user's constraints on the itemset $P$ that is returned. Hence, CP4IM is able to deal with queries of type $Q_1$, in which user's constraints are on itemsets only. However, in CP4IM, the variables $T$ representing transactions are internal variables only used to get the cover of the itemset $P$ that is returned, that is, $T_i = 1$ if and only if the itemset $P$ is covered by transaction $i$. These $T$ variables are not decision variables that would allow constraining the transactions. Adding user's constraints directly on these variables would generate incorrect models.

MiningZinc is a programming language on top of Minizinc. Several examples of complex data mining queries using MiningZinc are discussed in [2]. However, in these examples, when closedness is required, the user's constraints are monotone, and when the mining is performed on sub-datasets, these sub-datasets are statically defined. If we need the mining process to dynamically specify on which sub-datasets the frequency, closeness, and other properties are computed, we believe that MiningZinc requires to implement a model similar to the one we propose in this paper.

## 8 Conclusion

We have presented a taxonomy of the different types of user's constraints for itemset mining. Constraints can express properties on the itemsets as well as on the items and transactions that compose the datasets on which to look. We have introduced a generic constraint programming model for itemset mining. We showed how our generic CP model can easily take into account any type of user's constraints. We empirically evaluated our CP model. We have shown that it can handle the different types of constraints on different datasets. The CP approach can find the itemsets satisfying all users constraints in an efficient way compared to the specialized algorithm LCM, which requires a memory/time consuming preprocessing step.

**Acknowledgment** Christian Bessiere was partially supported by the ANR project DEMOGRAPH (ANR-16-CE40-0028). Nadjib Lazaar is supported by the project I3A TRACT (CNRS INSMI INS2I - AMIES - 2018). Mehdi Maamar is supported by the project CPER Data from the region "Hauts-de-France". We thank Yahia Lebbah for the discussions we shared during this work.

## References

1. Bonchi, F., Lucchese, C.: On closed constrained frequent pattern mining. In: Proceedings of the 4th IEEE International Conference on Data Mining (ICDM 2004), 1-4 November 2004, Brighton, UK. pp. 35–42 (2004)
2. Guns, T., Dries, A., Nijssen, S., Tack, G., Raedt, L.D.: Miningzinc: A declarative framework for constraint-based mining. Artif. Intell. **244**, 6–29 (2017)

3. Guns, T., Nijssen, S., Raedt, L.D.: Itemset mining: A constraint programming perspective. Artif. Intell. **175**(12-13), 1951–1983 (2011)
4. Kemmar, A., Lebbah, Y., Loudni, S., Boizumault, P., Charnois, T.: Prefix-projection global constraint and top-k approach for sequential pattern mining. Constraints **22**(2), 265–306 (2017)
5. Khiari, M., Boizumault, P., Crémilleux, B.: Constraint programming for mining n-ary patterns. In: Principles and Practice of Constraint Programming - CP 2010 - 16th International Conference, CP 2010, St. Andrews, Scotland, UK, September 6-10, 2010. Proceedings. pp. 552–567 (2010)
6. Lazaar, N., Lebbah, Y., Loudni, S., Maamar, M., Lemière, V., Bessiere, C., Boizumault, P.: A global constraint for closed frequent pattern mining. In: Principles and Practice of Constraint Programming - 22nd International Conference, CP 2016, Toulouse, France, September 5-9, 2016, Proceedings. pp. 333–349 (2016)
7. Mannila, H., Toivonen, H.: Levelwise search and borders of theories in knowledge discovery. Data Min. Knowl. Discov. **1**(3), 241–258 (1997)
8. Raedt, L.D., Guns, T., Nijssen, S.: Constraint programming for itemset mining. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008. pp. 204–212 (2008)
9. Schaus, P., Aoga, J.O.R., Guns, T.: Coversize: A global constraint for frequency-based itemset mining. In: Principles and Practice of Constraint Programming - 23rd International Conference, CP 2017, Melbourne, VIC, Australia, August 28 - September 1, 2017, Proceedings. pp. 529–546 (2017)
10. Uno, T., Asai, T., Uchida, Y., Arimura, H.: An efficient algorithm for enumerating closed patterns in transaction databases. In: Discovery Science, 7th International Conference, DS 2004, Padova, Italy, October 2-5, 2004, Proceedings. pp. 16–31 (2004)
11. Wojciechowski, M., Zakrzewicz, M.: Dataset filtering techniques in constraint-based frequent pattern mining. In: Pattern Detection and Discovery, ESF Exploratory Workshop, London, UK, September 16-19, 2002, Proceedings. pp. 77–91 (2002)
12. Zaki, M.J., Hsiao, C.: CHARM: an efficient algorithm for closed itemset mining. In: Proceedings of the Second SIAM International Conference on Data Mining, Arlington, VA, USA, April 11-13, 2002. pp. 457–473 (2002)